

Recommendation

ITU-T H.274 (V3) (09/2023)

SERIES H: Audiovisual and multimedia systems

Infrastructure of audiovisual services – Coding of moving video

**Versatile supplemental enhancement
information messages for coded video
bitstreams**



ITU-T H-SERIES RECOMMENDATIONS

Audiovisual and multimedia systems

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100-H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	H.200-H.499
General	H.200-H.219
Transmission multiplexing and synchronization	H.220-H.229
Systems aspects	H.230-H.239
Communication procedures	H.240-H.259
Coding of moving video	H.260-H.279
Related systems aspects	H.280-H.299
Systems and terminal equipment for audiovisual services	H.300-H.349
Directory services architecture for audiovisual and multimedia services	H.350-H.359
Quality of service architecture for audiovisual and multimedia services	H.360-H.369
Telepresence, immersive environments, virtual and extended reality	H.420-H.439
Supplementary services for multimedia	H.450-H.499
MOBILITY AND COLLABORATION PROCEDURES	H.500-H.549
VEHICULAR GATEWAYS AND INTELLIGENT TRANSPORTATION SYSTEMS (ITS)	H.550-H.599
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	H.600-H.699
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	H.700-H.799
E-HEALTH MULTIMEDIA SYSTEMS, SERVICES AND APPLICATIONS	H.800-H.899

For further details, please refer to the list of ITU-T Recommendations.

**Versatile supplemental enhancement information messages
for coded video bitstreams**

Summary

Recommendation ITU-T H.274 specifies the syntax and semantics of video usability information (VUI) parameters and supplemental enhancement information (SEI) messages for use with coded video bitstreams. The VUI parameters and SEI messages defined in this Recommendation may be conveyed within coded video bitstreams in a manner specified in a video coding specification or may be conveyed by other means as determined by the specifications for systems that make use of such coded video bitstreams. This Recommendation is particularly intended for use with coded video bitstreams as specified by Rec. ITU-T H.266 | ISO/IEC 23090-3, although it is drafted in a manner intended to be sufficiently versatile and generic that it may also be used with other types of coded video bitstreams.

The 3rd edition adds four new versatile supplemental enhancement information (VSEI) messages for coded video bitstreams, and also includes corrections and editorial improvements to various minor defects in the prior content of this Recommendation. Changes are included as modifications to the sections of text that were previously present in H.274 V2.

This Recommendation was developed collaboratively with ISO/IEC JTC 1/SC 29 and corresponds with ISO/IEC 23002-7 as technically aligned twin text.

History*

Edition	Recommendation	Approval	Study Group	Unique ID
1.0	ITU-T H.274	2020-08-29	16	11.1002/1000/14337
2.0	ITU-T H.274 (V2)	2022-05-22	16	11.1002/1000/14949
3.0	ITU-T H.274 (V3)	2023-09-29	16	11.1002/1000/15651

* To access the Recommendation, type the URL <https://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2023

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

	<i>Page</i>
1	Scope..... 1
2	Normative references 1
2.1	Identical Recommendations International Standards 1
2.2	Paired Recommendations International Standards equivalent in technical content 1
2.3	Additional references 1
3	Definitions..... 2
4	Abbreviations 5
5	Conventions..... 6
5.1	General..... 6
5.2	Arithmetic operators 6
5.3	Logical operators 7
5.4	Relational operators 7
5.5	Bit-wise operators 7
5.6	Assignment operators..... 7
5.7	Range notation 8
5.8	Mathematical functions..... 8
5.9	Order of operation precedence..... 9
5.10	Variables, syntax elements and tables..... 9
5.11	Text description of logical operations..... 10
5.12	Processes..... 11
6	Syntax and semantics 12
6.1	General..... 12
6.2	Method of specifying syntax in tabular form 12
6.3	Specification of syntax functions and descriptors 13
7	Video usability information parameters 14
7.1	General..... 14
7.2	VUI parameters syntax..... 14
7.3	VUI parameters semantics 15
8	SEI messages..... 20
8.1	General..... 20
8.2	Filler payload SEI message..... 22
8.2.1	Filler payload SEI message syntax 22
8.2.2	Filler payload SEI message semantics 22
8.3	User data registered by Rec. ITU-T T.35 SEI message 22
8.3.1	User data registered by Rec. ITU-T T.35 SEI message syntax 22
8.3.2	User data registered by Rec. ITU-T T.35 SEI message semantics..... 22
8.4	User data unregistered SEI message 22
8.4.1	User data unregistered SEI message syntax 22
8.4.2	User data unregistered SEI message semantics..... 23
8.5	Film grain characteristics SEI message..... 23
8.5.1	Film grain characteristics SEI message syntax 23
8.5.2	Film grain characteristics SEI message semantics 24
8.6	Frame packing arrangement SEI message 30
8.6.1	Frame packing arrangement SEI message syntax 30
8.6.2	Frame packing arrangement SEI message semantics..... 30
8.7	Parameter sets inclusion indication SEI message..... 37
8.7.1	Parameter sets inclusion indication SEI message syntax 37
8.7.2	Parameter sets inclusion indication SEI message semantics 37
8.8	Decoded picture hash SEI message..... 38
8.8.1	Decoded picture hash SEI message syntax 38
8.8.2	Decoded picture hash SEI message semantics 38
8.9	Mastering display colour volume SEI message 40
8.9.1	Mastering display colour volume SEI message syntax 40
8.9.2	Mastering display colour volume SEI message semantics..... 40

8.10	Content light level information SEI message.....	41
8.10.1	Content light level information SEI message syntax	41
8.10.2	Content light level information SEI message semantics	41
8.11	Dependent random access point indication SEI message.....	42
8.11.1	Dependent random access point indication SEI message syntax	42
8.11.2	Dependent random access point indication SEI message semantics	42
8.12	Alternative transfer characteristics information SEI message.....	43
8.12.1	Alternative transfer characteristics information SEI message syntax	43
8.12.2	Alternative transfer characteristics SEI message semantics.....	43
8.13	Ambient viewing environment SEI message	43
8.13.1	Ambient viewing environment SEI message syntax	43
8.13.2	Ambient viewing environment SEI message semantics.....	43
8.14	Content colour volume SEI message	44
8.14.1	Content colour volume SEI message syntax	44
8.14.2	Content colour volume SEI message semantics.....	44
8.15	Omnidirectional video specific SEI messages	46
8.15.1	Sample location remapping process.....	46
8.15.2	Equirectangular projection SEI message.....	55
8.15.3	Generalized cubemap projection SEI message	56
8.15.4	Sphere rotation SEI message.....	62
8.15.5	Region-wise packing SEI message	63
8.15.6	Omnidirectional viewport SEI message.....	69
8.16	Frame-field information SEI message.....	70
8.16.1	Frame-field information SEI message syntax	70
8.16.2	Frame-field information SEI message semantics.....	71
8.17	Sample aspect ratio information SEI message	73
8.17.1	Sample aspect ratio information SEI message syntax.....	73
8.17.2	Sample aspect ratio information SEI message semantics.....	74
8.18	Annotated regions SEI message.....	74
8.18.1	Annotated regions SEI message syntax	74
8.18.2	Annotated regions SEI message semantics	76
8.19	Scalability dimension information SEI message.....	79
8.19.1	Scalability dimension information SEI message syntax	79
8.19.2	Scalability dimension information SEI message semantics	79
8.20	Multiview acquisition information SEI message	80
8.20.1	Multiview acquisition information SEI message syntax	80
8.20.2	Multiview acquisition information SEI message semantics.....	81
8.21	Multiview view position SEI message.....	85
8.21.1	Multiview view position SEI message syntax.....	85
8.21.2	Multiview view position SEI message semantics	85
8.22	Depth representation information SEI message	85
8.22.1	Depth representation information SEI message syntax.....	85
8.22.2	Depth representation information SEI message semantics	86
8.23	Alpha channel information SEI message	88
8.23.1	Alpha channel information SEI message syntax	88
8.23.2	Alpha channel information SEI message semantics.....	89
8.24	Extended DRAP indication SEI message.....	91
8.24.1	Extended DRAP indication SEI message syntax	91
8.24.2	Extended DRAP indication SEI message semantics.....	91
8.25	Display orientation SEI message	92
8.25.1	Display orientation SEI message syntax	92
8.25.2	Display orientation SEI message semantics.....	92
8.26	Colour transform information SEI message.....	93
8.26.1	Colour transform information SEI message syntax	93
8.26.2	Colour transform information SEI message semantics	94
8.27	Shutter interval information SEI message.....	97
8.27.1	Shutter interval information SEI message syntax	97
8.27.2	Shutter interval information SEI message semantics	97
8.28	Neural-network post-filter SEI messages.....	98
8.28.1	General post-processing filtering process using NNPFs.....	98

8.28.2	Neural-network post-filter characteristics SEI message.....	99
8.28.3	Neural-network post-filter activation SEI message.....	116
8.29	Phase indication SEI message.....	118
8.29.1	Phase indication SEI message syntax.....	118
8.29.2	Phase indication SEI message semantics	118
8.30	Reserved SEI message	119
8.30.1	Reserved SEI message syntax	119
8.30.2	Reserved SEI message semantics.....	119
9	Parsing process for k-th order Exp-Golomb codes.....	119
9.1	General.....	119
9.2	Mapping process for signed Exp-Golomb codes	120
	Bibliography	122

List of Tables

	<i>Page</i>
Table 1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)	9
Table 2 – SubWidthC and SubHeightC values derived from ChromaFormatIdc	15
Table 3 – Definition of HorizontalOffsetC and VerticalOffsetC as a function of ChromaFormatIdc and Chroma420LocType	20
Table 4 – Persistence scope of SEI messages (informative)	20
Table 5 – fg_model_id values	24
Table 6 – fg_blending_mode_id values	26
Table 7 – Definition of fp_arrangement_type	31
Table 8 – Definition of fp_content_interpretation_type	32
Table 9 – Interpretation of dph_sei_hash_type	39
Table 10 – Specification of packing type and position index based on gcmp_packing_type	58
Table 11 – Specification of counterclockwise rotation angle based on gcmp_face_rotation[i]	60
Table 12 – Specification of guard band boundary location based on gcmp_packing_type and gcmp_guard_band_boundary_exterior_flag	61
Table 13 – rwp_transform_type[i] values	66
Table 14 – Interpretation of frame-field information syntax elements	72
Table 15 – Mapping of sdi_aux_id[i] to the type of auxiliary pictures	80
Table 16 – Association between camera parameter variables and syntax elements	84
Table 17 – Definition of depth_representation_type	87
Table 18 – Association between depth parameter variables and syntax elements	87
Table 19 – display_orientation_transform_type values	93
Table 24 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)	120
Table 25 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)	120
Table 26 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)	121

List of Figures

	<i>Page</i>
Figure 1 – Location of chroma samples for top and bottom fields for ChromaFormatIdc equal to 1 (4:2:0 chroma format) as a function of vui_chroma_sample_loc_type_top_field and vui_chroma_sample_loc_type_bottom_field in the range of 0 to 5, inclusive.....	18
Figure 2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture	18
Figure 3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture	19
Figure 4 – Location of the top-left chroma sample when ChromaFormatIdc is equal to 1 (4:2:0 chroma format) and Chroma420LocType is equal to 0 to 5, inclusive, from left to right.....	19
Figure 5 – Location of the top-left chroma sample when ChromaFormatIdc is equal to 1 (4:2:0 chroma format) when Chroma420LocType is equal to 1.....	20
Figure 6 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with fp_arrangement_type equal to 3, fp_quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (4, 8) for both constituent frames.....	34
Figure 7 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with fp_arrangement_type equal to 3, fp_quincunx_sampling_flag equal to 0, (x, y) equal to (12, 8) for constituent frame 0 and (x, y) equal to (0, 0) or (4, 8) for constituent frame 1	35
Figure 8 – Flowchart for rearrangement and upconversion of top-bottom packing arrangement with fp_arrangement_type equal to 4, fp_quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (8, 4) for both constituent frames.....	35
Figure 9 – Flowchart for rearrangement and upconversion of top-bottom packing arrangement with fp_arrangement_type equal to 4, fp_quincunx_sampling_flag equal to 0, (x, y) equal to (8, 12) for constituent frame 0 and (x, y) equal to (0, 0) or (8, 4) for constituent frame 1	36
Figure 10 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling (fp_arrangement_type equal to 3 with fp_quincunx_sampling_flag equal to 1).....	36
Figure 11 – Flowchart for rearrangement of a temporal interleaving frame arrangement (fp_arrangement_type equal to 5).....	37

Introduction

Versions of this Recommendation | International Standard

Recommendation ITU-T H.274 | ISO/IEC 23002-7 version 1 refers to the first approved version of this Recommendation | International Standard. The first edition published by ITU-T as Rec. ITU-T H.274 (08/2020) and by ISO/IEC as ISO/IEC 23002-7:2021 corresponded to the first version.

Recommendation ITU-T H.274 | ISO/IEC 23002-7 version 2 refers to the integrated text additionally containing nine additional SEI messages, namely the annotated regions SEI message, the alpha channel information SEI message, the depth representation information SEI message, the multiview acquisition information SEI message, the scalability dimension information SEI message, the extended dependent random access point indication SEI message, the display orientation SEI message, the colour transform information SEI message, and the multiview view position SEI message. Besides these additional SEI messages, the version 2 integrated text also contains corrections to various minor defects in the prior content of the Specification. The second edition published by ITU-T as Rec. H.274 (05/2022) and by ISO/IEC as ISO/IEC 23002-7:2022 corresponded to the second version.

Rec. ITU-T H.274 | ISO/IEC 23002-7 version 3 (the current version) refers to the integrated text additionally containing four additional SEI messages, namely the shutter interval information SEI message, the neural-network post-filter characteristics SEI message, the neural-network post-filter activation SEI message, and the phase indication SEI message. Besides these additional SEI messages, the version 3 integrated text also contains corrections to various minor defects in the prior content of the Specification. The third edition published by ITU-T as Rec. ITU-T H.274 (09/2023) corresponds to the third version. At the time of publication of this edition by ITU-T, a corresponding third edition of ISO/IEC 23002-7 was in preparation for publication by ISO/IEC.

Versatile supplemental enhancement information messages for coded video bitstreams

1 Scope

This Recommendation | International Standard specifies the syntax and semantics of video usability information (VUI) parameters and supplemental enhancement information (SEI) messages. The VUI parameters and SEI messages defined in this Specification are designed to be conveyed within coded video bitstreams in a manner specified in a video coding specification or to be conveyed by other means determined by the specifications for systems that make use of such coded video bitstreams. This Specification is particularly intended for use with coded video bitstreams as specified by Rec. ITU-T H.266 | ISO/IEC 23090-3, although it is drafted in a manner intended to be sufficiently generic that it can also be used with other types of coded video bitstreams.

VUI parameters and SEI messages can assist in processes related to decoding, display or other purposes. However, unless otherwise specified in a referencing specification, the interpretation and use of the VUI parameters and SEI messages specified in this Specification is not a required functionality of a video decoder or receiving video system. Although semantics are specified for the VUI parameters and SEI messages, decoders and receiving video systems can simply ignore the content of the VUI parameters and SEI messages or can use them in a manner that somewhat differs from what is specified in this Specification.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- None

2.2 Paired Recommendations | International Standards equivalent in technical content

- Recommendation ITU-T H.273 (in force) | ISO/IEC 23091-2 (in force), *Coding-independent code points for video signal type identification*.

2.3 Additional references

- Recommendation ITU-T T.35 (in force), *Procedure for the allocation of ITU-T defined codes for non standard facilities*.
- IETF RFC 1321 (in force), *The MD5 Message-Digest Algorithm*.
- IETF RFC 4151 (in force), *The 'tag' URI Scheme*.
- IETF RFC 5646 (in force), *Tags for Identifying Languages*.
- IETF Internet Standard 66 (in force), *Uniform Resource Identifier (URI): Generic Syntax*.
- ISO/CIE 11664-1 (in force), *Colorimetry – Part 1: CIE standard colorimetric observers*.
- ISO/IEC 10646 (in force), *Information technology – Universal coded character set (UCS)*.
- ISO/IEC 11578:1996, *Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)*.
- ISO/IEC 15938-17 (in force), *Information technology – Multimedia content description interface – Part 17: Compression of neural networks for multimedia content description and analysis*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

- 3.1 access unit (AU):** A set of *PUs* that belong to different *layers* and contain *coded pictures* associated with the same *output time*.
- 3.2 adaptation parameter set (APS):** A *syntax structure* containing *syntax elements* that apply to zero or more *slices* as determined by zero or more *syntax elements* found in *slice headers*.
- 3.3 alpha blending:** A process not specified by this Specification, in which an auxiliary *coded picture* is used in combination with a primary *coded picture* and with other data not specified by this Specification in the display process. In an alpha blending process, the samples of an auxiliary *coded picture* are interpreted as indications of the degree of opacity (or, equivalently, the degrees of transparency) associated with the corresponding *luma* samples of the primary *coded picture*.
- 3.4 associated IRAP picture (of a particular picture):** The previous *IRAP picture* in *decoding order* (when present) in the same *layer* as the particular *picture*.
- 3.5 azimuth circle:** A circle on a sphere connecting all points with the same azimuth value.
NOTE – An azimuth circle is always a *great circle* like a longitude line on the earth.
- 3.6 byte:** A sequence of 8 bits, within which, when written or read as a sequence of bit values, the left-most and right-most bits represent the most and least significant bits, respectively, and the bits are written or read from left to right.
- 3.7 chroma:** An adjective, represented by the symbols *Cb* and *Cr*, specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours.
NOTE – The term chroma is used rather than the term chrominance in order to avoid implying the use of linear light transfer characteristics that is often associated with the term chrominance.
- 3.8 coded layer video sequence (CLVS):** A sequence of *PUs* of the same layer that consists, in *decoding order*, of a *CLVSS PU*, followed by zero or more *PUs* that are not *CLVSS PUs*, including all subsequent *PUs* up to but not including any subsequent *PU* that is a *CLVSS PU*.
- 3.9 coded layer video sequence start (CLVSS) PU:** A *PU* in which the *coded picture* is a *CLVSS picture*.
- 3.10 coded layer video sequence start (CLVSS) picture:** A *coded picture* that starts a new *CLVS* as specified in a video coding specification.
NOTE – In Rec. ITU-T H.266 | ISO/IEC 23090-3, a *CLVSS picture* is an *IRAP picture* with *NoIncorrectPicOutputFlag* equal to 1 or a gradual decoding refresh picture with *NoIncorrectPicOutputFlag* equal to 1. In Rec. ITU-T H.265 | ISO/IEC 23008-2, a *CLVSS picture* is an *IRAP picture* with *NoRaslOutputFlag* equal to 1.
- 3.11 coded picture:** A *coded representation* of a *picture* containing all *CTUs* of the *picture*.
- 3.12 coded slice NAL unit:** A *NAL unit* that contains a *coded slice*.
- 3.13 coded video bitstream:** A sequence of bits that forms the representation of a sequence of *AUs* forming one or more coded video sequences (*CVSs*).
- 3.14 coded video sequence (CVS):** A sequence of *AUs* that consists, in *decoding order*, of a *CVSS AU*, followed by zero or more *AUs* that are not *CVSS AUs*, including all subsequent *AUs* up to but not including any subsequent *AU* that is a *CVSS AU*.
- 3.15 coded video sequence start (CVSS) AU:** An *AU* that has a *PU* for each *layer* present in the *CVS* and the *coded picture* in each *PU* is a *CLVSS picture*.
- 3.16 component:** An array or single sample from one of the three arrays (*luma* and two *chroma*) that compose a *picture* in 4:2:0, 4:2:2, or 4:4:4 colour format or the array or a single sample of the array that compose a *picture* in monochrome format.
- 3.17 constituent picture:** A part of a spatially *frame*-packed stereoscopic *picture* that corresponds to one view, or a *picture* itself when *frame* packing is not in use or the temporal interleaving *frame* packing arrangement is in use.
- 3.18 cropped decoded picture:** The result of cropping a *decoded picture* based on the conformance cropping window for the corresponding *coded picture*.
- 3.19 decoded picture:** A *decoded picture* is derived by decoding a *coded picture*.
- 3.20 decoder:** An embodiment of a *decoding process*.

- 3.21 decoding order:** The order in which *syntax elements* are conveyed in the *coded video bitstream* and are processed by a *decoding process*.
- 3.22 decoding process:** The process that reads a *coded video bitstream* and derives *decoded pictures* from it.
- 3.23 elevation circle:** A circle on a sphere connecting all points with the same elevation value.
NOTE – An elevation circle is similar to a latitude line on the earth. Except when the elevation value is zero, an elevation circle is not a *great circle* like a longitude circle on the earth.
- 3.24 encoder:** An embodiment of an *encoding process*.
- 3.25 encoding process:** A process that produces a *coded video bitstream*.
- 3.26 field:** An assembly of alternative rows of samples of a *frame*.
- 3.27 flag:** A variable or single-bit *syntax element* that can take one of the two possible values: 0 and 1.
- 3.28 frame:** The composition of a top *field* and a bottom *field*, where sample rows 0, 2, 4, ... originate from the top *field* and sample rows 1, 3, 5, ... originate from the bottom *field*.
- 3.29 global coordinate axes:** The coordinate axes associated with *omnidirectional video* that are associated with an externally referenceable position and orientation.
NOTE – The global coordinate axes could correspond to the position and orientation of a device or rig used for omnidirectional audio/video acquisition as well as the position of an observer's head in the three-dimensional space of the *omnidirectional video* rendering environment.
- 3.30 great circle:** The intersection of a sphere and a plane that passes through the centre point of the sphere.
NOTE – A great circle is also known as an orthodrome or Riemannian circle.
- 3.31 inter prediction:** An aspect of the *decoding process* for a *coded picture* that makes use of data derived from the *decoding process* of one or more previously decoded *reference pictures*.
- 3.32 intra random access point (IRAP) AU:** An *AU* in which each *coded picture* is an *IRAP picture*.
- 3.33 intra random access point (IRAP) picture:** A *coded picture* starting from which all *pictures* in the same *layer* in both *decoding order* and *output order* can be decoded without first decoding any *picture* in the same *layer* earlier in *decoding order* in the *coded video bitstream*.
- 3.34 layer:** A set of *VCL NAL units* that all have a particular value of layer identifier and the associated non-VCL *NAL units*, wherein the layer identifier is a variable for which the value is specified by a video coding specification.
NOTE – In the contexts of Rec. ITU-T H.266 | ISO/IEC 23090-3 and Rec. ITU-T H.265 | ISO/IEC 23008-2, the layer identifier is the value of the *nuh_layer_id* syntax element in the *NAL unit header*.
- 3.35 leading picture (of an IRAP picture):** A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.
- 3.36 local coordinate axes:** The coordinate axes having a specified rotation relationship relative to the *global coordinate axes*.
- 3.37 luma:** An adjective, represented by the symbol or subscript Y or L, specifying that a sample array or single sample is representing the monochrome signal related to the primary colours.
NOTE – The term *luma* is used rather than the term *luminance* in order to avoid implying the use of linear light transfer characteristics that is often associated with the term *luminance*. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol *y* as used for vertical location.
- 3.38 network abstraction layer (NAL) unit:** A *syntax structure* containing an indication of the type of data that follows and *bytes* containing that data in a manner that enables the extraction of a string of data bits from the *syntax structure*.
- 3.39 network abstraction layer (NAL) unit stream:** A sequence of *NAL units*.
- 3.40 non-VCL NAL unit:** A *NAL unit* that is not a *VCL NAL unit*.
- 3.41 omnidirectional video:** A video content in a format that enables rendering according to the user's viewing orientation, e.g., if viewed using a head-mounted device, or according to a user's desired *viewport*, reflecting a potentially rotated viewing position.
- 3.42 output order:** The order in which the *decoded pictures* are output from the *decoder* (for the *decoded pictures* that are to be output from the *decoder*).

- 3.43 output time:** A time when a *decoded picture* is to be output from the *decoder* (for the *decoded pictures* that are to be output from the *decoder*).
- 3.44 packed region:** A region in a *region-wise packed picture* that is mapped to a *projected region* according to a *region-wise packing*.
- 3.45 picture:** An array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format.
NOTE – A picture could be either a frame or a field. However, in one CLVS, either all pictures are frames, or all pictures are fields.
- 3.46 picture parameter set (PPS):** A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by a *syntax element* that is the same for all *slices* of a picture and found in the picture header or *slice headers* of each *picture*.
- 3.47 picture unit (PU):** A set of *NAL units* that contain all *VCL NAL units* of a *coded picture* and their associated non-VCL *NAL units*.
- 3.48 projected picture:** A picture that uses a *projection* format for *omnidirectional video*.
- 3.49 projected region:** A region in a *projected picture* that is mapped to a *packed region* according to a *region-wise packing*.
- 3.50 projection:** A specified correspondence between the colour samples of a *projected picture* and azimuth and elevation positions on a sphere.
- 3.51 random access:** The act of starting the decoding process for a *coded video bitstream* at a point other than the beginning of the bitstream.
- 3.52 random access skipped leading (RASL) picture:** A *leading picture* that cannot be correctly decoded when the decoding process starts from the *associated IRAP picture*.
- 3.53 reference picture:** A *picture* that contains samples that could be used for *inter prediction* in the decoding process of subsequent pictures in decoding order.
- 3.54 reference picture list:** A list of *reference pictures* that is used for *inter prediction* of a *slice*.
- 3.55 region-wise packed picture:** A decoded picture that contains one or more *packed regions*.
NOTE – A region-wise packed picture could contain a *region-wise packing* of a *projected picture*.
- 3.56 region-wise packing:** A transformation, resizing, and relocation of *packed regions* of a *region-wise packed picture* to remap the *packed regions* to *projected regions* of a *projected picture*.
- 3.57 sample aspect ratio (SAR):** The indicated width-to-height aspect ratio of the *luma* samples of the associated *decoded pictures*.
- 3.58 slice:** A region of a *picture* that can be decoded separately from other regions of the same *coded picture* (although in some cases the *decoding process* for the *picture* might use *inter prediction* that makes reference to other previously decoded *reference pictures*).
- 3.59 source:** A term used to describe the video material or some of its attributes before encoding.
- 3.60 sphere coordinates:** The azimuth and elevation angles identifying a location of a point on a sphere.
- 3.61 sphere region:** A region on a sphere, specified either by four *great circles* or by two *azimuth circles* and two *elevation circles*, or such a region on a rotated sphere after applying yaw, pitch, and roll rotations.
- 3.62 step-wise temporal sublayer access (STSA) picture:** A *coded picture* that enables up-switching, at the *coded picture*, to the *temporal sublayer* containing the *coded picture*, from the immediately lower *temporal sublayer* of the same *layer* when the *coded picture* does not belong to the lowest *temporal sublayer*.
NOTE – An STSA picture does not use pictures in the same layer and with the same temporal sublayer identifier as the STSA picture for inter prediction reference. Pictures following an STSA picture in decoding order in the same layer and with the same temporal sublayer identifier as the STSA picture do not use pictures prior to the STSA picture in decoding order in the same layer and with the same temporal sublayer identifier as the STSA picture for inter prediction reference. STSA pictures in an independent layer (i.e., a layer that does not depend on other layers in its decoding) always have a temporal sublayer identifier greater than 0.
- 3.63 supplemental enhancement information (SEI) message:** A *syntax structure* that provides a particular type of information that assists in processes related to decoding, display or other purposes but is not needed by the *decoding process* in order to determine the values of the samples in *decoded pictures*.
- 3.64 syntax element:** An element of data represented in a *syntax structure*.

- 3.65 syntax structure:** Zero or more *syntax elements* that are present together in a specified order in a string of data bits, where the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.
- 3.66 temporal sublayer:** A subset of a temporal scalable *bitstream*, consisting of *VCL NAL units* with a particular value of *temporal sublayer identifier* and the associated *non-VCL NAL units*.
- 3.67 temporal sublayer identifier:** A number greater than or equal to 0 defined by a variable for which the value is specified by a video coding specification such that pictures of all *temporal sublayers* have a specified temporal output order relative to each other and pictures with a lower temporal sublayer identifier can be decoded without reference to pictures with a higher temporal sublayer identifier.
- 3.68 tilt angle:** The angle indicating the amount of tilt of a *sphere region*, measured as the amount of rotation of a *sphere region* along the axis originating from the sphere origin passing through the centre point of the *sphere region*, where the angle value increases clockwise when looking from the origin towards the positive end of the axis.
- 3.69 trailing picture:** A *coded picture* that follows an *IRAP* picture in both decoding order and output order.
- 3.70 video coding layer (VCL) NAL unit:** A collective term for *coded slice NAL units* and the subset of other *NAL units* that have *reserved* values of *NAL unit* type identifiers that are classified as VCL NAL units in a referencing specification.
- 3.71 video usability information (VUI) parameters:** A syntax structure that identifies properties of interpretation of decoded pictures for display purposes, particularly including colour representation information.
- 3.72 viewport:** A region of *omnidirectional video* content suitable for display and viewing by the user.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

ACI	Alpha Channel Information
APS	Adaptation Parameter Set
AU	Access Unit
CLVS	Coded Layer Video Sequence
CLVSS	Coded Layer Video Sequence Start
CRC	Cyclic Redundancy Check
CTI	Colour Transform Information
CVS	Coded Video Sequence
DRAP	Dependent Random Access Point
DRI	Depth Representation Information
EDRAP	Extended Dependent Random Access Point
FIR	Finite Impulse Response
IRAP	Intra Random Access Point
MAI	Multiview Acquisition Information
NAL	Network Abstraction Layer
NNPF	Neural-Network Post-processing Filter
NNPFA	Neural-Network Post-Filter Activation
NNPFC	Neural-Network Post-Filter Characteristics
PPS	Picture Parameter Set
PU	Picture Unit
RASL	Random Access Skipped Leading
RWP	Region-Wise Packing
SAR	Sample Aspect Ratio
SARI	Sample Aspect Ratio Information
SDI	Scalability Dimension Information

SEI	Supplemental Enhancement Information
STSA	Step-wise Temporal Sublayer Access
URI	Uniform Resource Identifier
VCL	Video Coding Layer
VUI	Video Usability Information

5 Conventions

5.1 General

The term "this Specification" is used to refer to this Recommendation | International Standard.

The word "shall" is used to express mandatory requirements for conformance to this Specification. When used to express a mandatory constraint on the values of syntax elements or the values of variables derived from these syntax elements, it is the responsibility of the encoder to ensure that the constraint is fulfilled.

The word "may" is used to refer to behaviour that is allowed, but not necessarily required.

The word "should" is used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Specification.

Content of this Specification that is identified as "informative" does not establish any mandatory requirements for conformance to this Specification and is thus not considered an integral part of this Specification. Informative remarks in the text are, in some cases, set apart and prefixed with the word "note" or "NOTE".

The word "reserved" is used to specify that some values of a particular syntax element are for future use by ITU-T | ISO/IEC and shall not be used in syntax structures conforming to this version of this Specification, but could potentially be used in syntax structures conforming to future versions of this Specification by ITU-T | ISO/IEC.

The word "unspecified" is used to describe some values of a particular syntax element to indicate that the values have no specified meaning in this Specification and are not expected to have a specified meaning in the future as an integral part of future versions of this Specification.

The mathematical operators used in this Specification are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division.

Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

5.2 Arithmetic operators

+	addition
−	subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	multiplication, including matrix multiplication
	exponentiation
x^y	Specifies x to the power of y . In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	integer division with truncation of the result toward zero For example, $7 / 4$ and $-7 / -4$ are truncated to 1 and $-7 / 4$ and $7 / -4$ are truncated to -1 .
÷	division in mathematical equations where no truncation or rounding is intended
$\frac{x}{y}$	division in mathematical equations where no truncation or rounding is intended
$\sum_{i=x}^y f(i)$	summation of $f(i)$ with i taking all integer values from x up to and including y
$x \% y$	modulus Remainder of x divided by y , defined only for integers x and y with $x \geq 0$ and $y > 0$

5.3 Logical operators

`x && y` Boolean logical "and" of `x` and `y`

`x || y` Boolean logical "or" of `x` and `y`

`!` Boolean logical "not"

`x ? y : z` if `x` is TRUE or not equal to 0, evaluates to the value of `y`; otherwise, evaluates to the value of `z`.

5.4 Relational operators

`>` greater than

`>=` greater than or equal to

`<` less than

`<=` less than or equal to

`==` equal to

`!=` not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.5 Bit-wise operators

`&` bit-wise "and"

When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

`|` bit-wise "or"

When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

`^` bit-wise "exclusive or"

When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

`x >> y` arithmetic right shift of a two's complement integer representation of `x` by `y` binary digits

This function is defined only for non-negative integer values of `y`. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of `x` prior to the shift operation.

`x << y` arithmetic left shift of a two's complement integer representation of `x` by `y` binary digits

This function is defined only for non-negative integer values of `y`. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.

5.6 Assignment operators

`=` assignment operator

`++` increment, i.e., `x++` is equivalent to `x = x + 1`; when used in an array index, evaluates to the value of the variable prior to the increment operation

`--` decrement, i.e., `x--` is equivalent to `x = x - 1`; when used in an array index, evaluates to the value of the variable prior to the decrement operation

`+=` increment by amount specified, i.e., `x += 3` is equivalent to `x = x + 3`, and `x += (-3)` is equivalent to `x = x + (-3)`

`-=` decrement by amount specified, i.e., `x -= 3` is equivalent to `x = x - 3`, and `x -= (-3)` is equivalent to `x = x - (-3)`.

5.7 Range notation

$x = y..z$ x takes on integer values starting from y to z , inclusive, with x , y , and z being integer numbers and z being greater than y .

5.8 Mathematical functions

$$\text{Abs}(x) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases} \quad (1)$$

$\text{Asin}(x)$ trigonometric inverse sine function, operating on an argument x that is in the range of -1.0 to 1.0 , inclusive, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians (2)

$\text{Atan}(x)$ trigonometric inverse tangent function, operating on an argument x , with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians (3)

$$\text{Atan2}(y, x) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \quad x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \quad x < 0 \ \&\& \ y \geq 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \quad x < 0 \ \&\& \ y < 0 \\ +\frac{\pi}{2} & ; \quad x = 0 \ \&\& \ y \geq 0 \\ -\frac{\pi}{2} & ; \quad \text{otherwise} \end{cases} \quad (4)$$

$\text{Ceil}(x)$ smallest integer greater than or equal to x . (5)

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \quad (6)$$

$\text{Cos}(x)$ trigonometric cosine function operating on an argument x in units of radians. (7)

$\text{Floor}(x)$ largest integer less than or equal to x . (8)

$\text{Ln}(x)$ natural logarithm of x (the base- e logarithm, where e is the natural logarithm base constant 2.718 281 828...). (9)

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases} \quad (10)$$

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases} \quad (11)$$

$$\text{Reflect}(x, y) = \begin{cases} \text{Min}(-y, x) & ; \quad y < 0 \\ \text{Max}(x - (y - x), 0) & ; \quad y > x \\ y & ; \quad \text{otherwise} \end{cases} \quad (12)$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5) \quad (13)$$

$$\text{Sign}(x) = \begin{cases} 1 & ; x > 0 \\ 0 & ; x = 0 \\ -1 & ; x < 0 \end{cases} \quad (14)$$

$\text{Sin}(x)$ trigonometric sine function operating on an argument x in units of radians (15)

$\text{Sqrt}(x)$ square root of x (16)

$\text{Tan}(x)$ trigonometric tangent function operating on an argument x in units of radians (17)

$$\text{Wrap}(x, y) = \begin{cases} \text{Max}(0, x + y + 1) & ; y < 0 \\ \text{Min}(x, y - x - 1) & ; y > x \\ y & ; \text{otherwise} \end{cases} \quad (18)$$

5.9 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- Operations of a higher precedence are evaluated before any operation of a lower precedence.
- Operations of the same precedence are evaluated sequentially from left to right.

Table 1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table 1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

Operations (with operands $x, y,$ and z)
"x++", "x--"
"!x", "-x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", " $\frac{x}{y}$ ", "x % y"
"x + y", "x - y" (as a two-argument operator), " $\sum_{i=x}^y f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x == y", "x != y"
"x & y"
"x y"
"x && y"
"x y"
"x ? y : z"
"x..y"
"x = y", "x += y", "x -= y"

5.10 Variables, syntax elements and tables

Syntax elements in the syntax tables are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), and one descriptor for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases, the syntax tables and semantics use the values of other variables derived from the values of syntax elements. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax

structure and all depending syntax structures. Variables starting with an upper case letter could, in some cases, be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the clause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and could contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the SEI message payload data are referred to as syntax functions. These functions are specified in clause 6.3 and assume the existence of a pointer with an indication of the position of the next bit to be read by the decoding process from the payload data. Syntax functions are described by their names, which are constructed as syntax element names and end with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in clause 5.8) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

A one-dimensional array is referred to as a list. A two-dimensional array is referred to as a matrix. Arrays can either be syntax elements or variables. Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing. Thus, an element of a matrix s at horizontal position x and vertical position y could be denoted either as $s[x][y]$ or as s_{yx} . A single column of a matrix could be referred to as a list and denoted by omission of the row index. Thus, the column of a matrix s at horizontal position x could be referred to as the list $s[x]$.

A specification of values of the entries in rows and columns of an array could be denoted by $\{ \{ \dots \} \{ \dots \} \}$, where each inner pair of brackets specifies the values of the elements within a row in increasing column order and the rows are ordered in increasing row order. Thus, setting a matrix s equal to $\{ \{ 1 \ 6 \} \{ 4 \ 9 \} \}$ specifies that $s[0][0]$ is set equal to 1, $s[1][0]$ is set equal to 6, $s[0][1]$ is set equal to 4, and $s[1][1]$ is set equal to 9.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", is used in some cases instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.11 Text description of logical operations

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
    statement 0
else if( condition 1 )
    statement 1
...
else /* informative remark on remaining condition */
    statement n
```

is typically described in the following manner:

... as follows / ... the following applies:

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- ...

- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described mathematically in the following form:

```

if( condition 0a  &&  condition 0b )
    statement 0
else if( condition 1a  ||  condition 1b )
    statement 1
...
else
    statement n

```

is typically described in the following manner:

... as follows / ... the following applies:

- If all of the following conditions are true, statement 0:
 - condition 0a
 - condition 0b
- Otherwise, if one or more of the following conditions are true, statement 1:
 - condition 1a
 - condition 1b
- ...
- Otherwise, statement n

In the text, a statement of logical operations as would be described mathematically in the following form:

```

if( condition 0 )
    statement 0
if( condition 1 )
    statement 1

```

is typically described in the following manner:

```

When condition 0, statement 0
When condition 1, statement 1

```

5.12 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification might also have a lower case variable explicitly specified as input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

6 Syntax and semantics

6.1 General

This Specification is written in a manner such that it is intended to be referenced by other technical specifications. Such other technical specifications are to be written in a manner to specify certain necessary elements to enable the use of the specified VUI parameters and SEI messages.

Technical specifications that reference this Specification for carrying VUI parameters syntax structure shall specify a container to carry the data of the VUI parameters syntax structure and to identify the length in bits of the VUI parameters syntax structure, e.g., the `vui_payload()` syntax structure specified in Rec. ITU-T H.266 | ISO/IEC 23090-3. The design of the container should provide the ability to detect the number of bits in the `vui_parameters()` syntax structure and to allow the number of bits to be increased in future versions of this Specification, thus enabling this Specification to provide extensibility by directly appending additional syntax elements to the end of the `vui_parameters()` syntax structure in future versions of this Specification. The syntax of the container of the `vui_parameters()` syntax structure is outside the scope of this Specification.

Technical specifications that reference this Specification for carrying SEI messages shall specify a way to carry the payload syntax of each specified SEI message, to identify which SEI message is conveyed, and to identify the length in bits of the SEI message syntax structure, e.g., the `sei_payload()` syntax structure specified in Rec. ITU-T H.266 | ISO/IEC 23090-3 and Rec. ITU-T H.265 | ISO/IEC 23008-2. The design of the container should provide the ability to detect the number of bits in an SEI message and to allow the number of bits to be increased in future versions of this Specification, thus enabling this Specification to provide extensibility by directly appending additional syntax elements to the end of the SEI message syntax structure in future versions of this Specification. The syntax of the container of the SEI messages as well as the method of identifying which SEI message is outside the scope of this Specification.

The length of the VUI parameters syntax structure or an SEI message syntax structure in bits is referred to herein by the variable `PayloadBits`, which is provided by an external means not specified in this Specification. The number of bytes that contains the payload data is referred to herein by the variable `payloadSize`, where `payloadSize` is equal to $\text{Ceil}(\text{PayloadBits} \div 8)$.

For the VUI parameters and most of the SEI messages specified in this version of this Specification (other than the filler payload, user data registered, user data unregistered, and reserved SEI messages), the values of `PayloadBits` and `payloadSize` are not used for the parsing of the syntax. However, in some future version of this Specification, the value of `PayloadBits` or `payloadSize` could be used as part of the syntax specification for these syntax structures, for example to identify whether payload extension data is present in the VUI parameters or in an SEI message syntax structure that was not specified in an earlier version of this Specification.

The syntax specification in Rec. ITU-T H.266 | ISO/IEC 23090-3 and Rec. ITU-T H.265 | ISO/IEC 23008-2 establishes, under some circumstances, a certain pattern of bits that is used for detecting the value of `PayloadBits`. It is expected that future versions of this Specification will be written to ensure that such future versions will be compatible with the pattern for extension data that is specified in those other specifications. This pattern is such that when extension data is present and the last bit of such extension data is the last (least significant) bit of a byte, the extension data ends with a byte that contains a bit equal to 1 followed by 7 bits that are equal to 0.

It is a requirement of bitstream conformance to this version of this Specification that the value of `PayloadBits`, as determined by this external means, shall be equal to the number of bits in the VUI parameters syntax structure or the SEI message syntax structure, as applicable.

It is a requirement of decoder conformance to this version of this Specification that when `PayloadBits` is greater than the number of bits in the VUI parameters syntax structure or an SEI message syntax structure, the extra data at the end of the VUI or SEI payload data shall be ignored. The semantics for such extra data could potentially be specified in some future version of this Specification.

For example, each SEI message could be carried as a string of data bits that is prefixed with an SEI message payload type indication derived as a `payloadType` variable within a NAL unit that could contain emulation prevention bytes as specified in Rec. ITU-T H.266 | ISO/IEC 23090-3. When such emulation prevention bytes are present, the emulation prevention bytes are not counted when determining the values of `PayloadBits` and `payloadSize`.

6.2 Method of specifying syntax in tabular form

The syntax tables in this Specification specify a superset of the syntax of the VUI parameters and all allowed SEI messages. Additional constraints on the syntax are specified, either directly or indirectly, in other clauses.

The following table lists examples of the syntax specification format. When **`syntax_element`** appears, it specifies that a syntax element is parsed from the VUI parameters syntax or an SEI message syntax and the data pointer is advanced to the next position beyond the syntax element in the syntax parsing process.

	Descriptor
/* A statement can be a syntax element with an associated descriptor or can be an expression used to specify conditions for the existence, type and quantity of syntax elements, as in the following two examples */	
syntax_element	ue(v)
conditioning statement	
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */	
{	
statement	
statement	
...	
}	
/* A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */	
while(condition)	
statement	
/* A "do ... while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */	
do	
statement	
while(condition)	
/* An "if ... else" structure specifies a test of whether a condition is true and, if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */	
if(condition)	
primary statement	
else	
alternative statement	
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */	
for(initial statement; condition; subsequent statement)	
primary statement	

6.3 Specification of syntax functions and descriptors

The functions presented in this clause are used in the syntactical description. These functions are expressed in terms of the value of the VUI parameters syntax or an SEI message syntax data pointer that indicates the position of the next bit to be read by the decoding process from the syntax structure.

more_data_in_payload() is specified as follows:

- If byte_aligned() is equal to TRUE and the current position in an SEI message syntax structure or vui_parameters() syntax structure is 8 * payloadSize bits from the beginning of the syntax structure, the return value of more_data_in_payload() is equal to FALSE.

- Otherwise, the return value of `more_data_in_payload()` is equal to `TRUE`.

`read_bits(n)` reads the next `n` bits from the syntax structure and advances the data pointer by `n` bit positions. When `n` is equal to 0, `read_bits(n)` is specified to return a value equal to 0 and to not advance the data pointer.

The following descriptors specify the parsing process of each syntax element:

- `b(8)`: byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function `read_bits(8)`.
- `f(n)`: fixed-pattern bit string using `n` bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)`.
- `i(n)`: signed integer using `n` bits. When `n` is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)` interpreted as a two's complement integer representation with most significant bit written first.
- `se(v)`: signed integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9 with the order `k` equal to 0.
- `st(v)`: null-terminated string encoded as universal coded character set (UCS) transmission format-8 (UTF-8) characters as specified in ISO/IEC 10646. The parsing process is specified as follows: `st(v)` begins at a byte-aligned position in the bitstream and reads and returns a series of bytes from the bitstream, beginning at the current position and continuing up to but not including the next byte-aligned byte that is equal to `0x00`, and advances the bitstream pointer by $(\text{stringLength} + 1) * 8$ bit positions, where `stringLength` is equal to the number of bytes returned.
NOTE – The `st(v)` syntax descriptor is only used in this Specification when the current position in the bitstream is a byte-aligned position.
- `u(n)`: unsigned integer using `n` bits. When `n` is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)` interpreted as a binary representation of an unsigned integer with most significant bit written first.
- `ue(v)`: unsigned integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9 with the order `k` equal to 0.

7 Video usability information parameters

7.1 General

Clause 7 specifies the syntax and semantics for VUI parameters.

When any information regarding the interpretation of the pictures is not present in the `vui_parameters()` syntax structure, or the `vui_parameters()` syntax structure is not present, there may be some external means that controls the interpretation.

7.2 VUI parameters syntax

	Descriptor
<code>vui_parameters(payloadSize) {</code>	
vui_progressive_source_flag	u(1)
vui_interlaced_source_flag	u(1)
vui_non_packed_constraint_flag	u(1)
vui_non_projected_constraint_flag	u(1)
vui_aspect_ratio_info_present_flag	u(1)
if(<code>vui_aspect_ratio_info_present_flag</code>) {	
vui_aspect_ratio_constant_flag	u(1)
vui_aspect_ratio_idc	u(8)
if(<code>vui_aspect_ratio_idc == 255</code>) {	
vui_sar_width	u(16)
vui_sar_height	u(16)
}	
}	
}	

vui_overscan_info_present_flag	u(1)
if(vui_overscan_info_present_flag)	
vui_overscan_appropriate_flag	u(1)
vui_colour_description_present_flag	u(1)
if(vui_colour_description_present_flag) {	
vui_colour_primaries	u(8)
vui_transfer_characteristics	u(8)
vui_matrix_coeffs	u(8)
vui_full_range_flag	u(1)
}	
vui_chroma_loc_info_present_flag	u(1)
if(vui_chroma_loc_info_present_flag) {	
if(vui_progressive_source_flag && !vui_interlaced_source_flag)	
vui_chroma_sample_loc_type_frame	ue(v)
else {	
vui_chroma_sample_loc_type_top_field	ue(v)
vui_chroma_sample_loc_type_bottom_field	ue(v)
}	
}	
}	

7.3 VUI parameters semantics

VUI parameters apply to one or more CLVSs.

NOTE 1 – The interpretation of several syntax elements of the VUI parameters are specified by reference to coding-independent code points specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. Further information about the usage of such code points is found in Supplement ITU-T H-Suppl. 19 | ISO/IEC TR 23091-4.

Use of the VUI parameters requires the definition of the following variables:

- A chroma format indicator, denoted herein by $ChromaFormatIdc$, such that the value 0 indicates that the picture has only a luma component and other values indicate that the picture has three colour components that consist of a luma component and two associated chroma components, such that the width and height of each chroma component are the width and height of the luma component divided by $SubWidthC$ and $SubHeightC$, respectively, where $SubWidthC$ and $SubHeightC$ are determined from $ChromaFormatIdc$ as specified by Table 2.
- A bit depth for the samples of the luma component, denoted herein by $BitDepth_Y$, and when $ChromaFormatIdc$ is not equal to 0, a bit depth for the samples of the two associated chroma components, denoted herein by $BitDepth_C$.

Table 2 – SubWidthC and SubHeightC values derived from ChromaFormatIdc

ChromaFormatIdc	Chroma format	SubWidthC	SubHeightC
0	Monochrome	1	1
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

vui_progressive_source_flag and **vui_interlaced_source_flag** are interpreted as follows:

- If **vui_progressive_source_flag** is equal to 1 and **vui_interlaced_source_flag** is equal to 0, the source scan type of the pictures should be interpreted as progressive only.
- Otherwise, if **vui_progressive_source_flag** is equal to 0 and **vui_interlaced_source_flag** is equal to 1, the source scan type of the pictures should be interpreted as interlaced only.

- Otherwise, if `vui_progressive_source_flag` is equal to 0 and `vui_interlaced_source_flag` is equal to 0, the source scan type of the pictures should be interpreted as unknown or unspecified or specified by external means not specified in this Specification.
- Otherwise (`vui_progressive_source_flag` is equal to 1 and `vui_interlaced_source_flag` is equal to 1), the source scan type of each picture is indicated at the picture level using the syntax element `ffi_source_scan_type` in a frame-field information SEI message.

vui_non_packed_constraint_flag equal to 1 specifies that there shall not be any frame packing arrangement SEI messages present in the bitstream that apply to the CLVS. `vui_non_packed_constraint_flag` equal to 0 does not impose such a constraint.

vui_non_projected_constraint_flag equal to 1 specifies that there shall not be any equirectangular projection SEI messages or generalized cubemap projection SEI messages present in the bitstream that apply to the CLVS. `vui_non_projected_constraint_flag` equal to 0 does not impose such a constraint.

vui_aspect_ratio_info_present_flag equal to 1 specifies that `vui_aspect_ratio_idc` is present. `vui_aspect_ratio_info_present_flag` equal to 0 specifies that `vui_aspect_ratio_idc` is not present.

vui_aspect_ratio_constant_flag equal to 1 specifies that the values of `vui_aspect_ratio_idc`, `SarWidth`, and `SarHeight` apply to all pictures in the CLVS and there is no SARI SEI message present in the CLVS. `vui_aspect_ratio_constant_flag` equal to 0 specifies that the values of `vui_aspect_ratio_idc`, `SarWidth`, and `SarHeight` might or might not apply to all pictures in the CLVS and that SARI SEI messages could be present in the CLVS indicating a different sample aspect ratio applicable to the pictures associated with SARI SEI messages. When the `vui_aspect_ratio_constant_flag` syntax element is not present, the value of `vui_aspect_ratio_constant_flag` is inferred to be equal to 0.

vui_aspect_ratio_idc, when not equal to 255, indicates the SAR of the luma samples of decoded pictures in the CLVS, unless indicated otherwise by associated SARI SEI messages when `vui_aspect_ratio_constant_flag` is equal to 0. Its semantics are as specified for the `SampleAspectRatio` parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When the `vui_aspect_ratio_idc` syntax element is not present, the value of `vui_aspect_ratio_idc` is inferred to be equal to 0. Values of `vui_aspect_ratio_idc` that are specified as reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret values of `vui_aspect_ratio_idc` that are reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 as equivalent to the value 0.

vui_sar_width, when present, indicates the horizontal size of the SAR (in arbitrary units) of the luma samples of decoded pictures in the CLVS, unless indicated otherwise by associated SARI SEI messages when `vui_aspect_ratio_constant_flag` is equal to 0.

vui_sar_height, when present, indicates the vertical size of the SAR (in the same arbitrary units as `vui_sar_width`) of the luma samples of decoded pictures in the CLVS, unless indicated otherwise by associated SARI SEI messages when `vui_aspect_ratio_constant_flag` is equal to 0.

When present, `vui_sar_width` and `vui_sar_height` shall be relatively prime or equal to 0. When `vui_aspect_ratio_idc` is equal to 0 or `vui_sar_width` is equal to 0 or `vui_sar_height` is equal to 0, the SAR is unknown or unspecified in this Specification or may be determined by other means, such as the SARI SEI message.

vui_overscan_info_present_flag equal to 1 specifies that the `vui_overscan_appropriate_flag` is present. When `vui_overscan_info_present_flag` is equal to 0 or is not present, the preferred display method for the video signal is unknown or unspecified or specified by external means.

vui_overscan_appropriate_flag equal to 1 indicates that the cropped decoded pictures output are suitable for display using overscan. `vui_overscan_appropriate_flag` equal to 0 indicates that the cropped decoded pictures output contain visually important information in the entire region out to the edges of the conformance cropping window of the picture, such that the cropped decoded pictures output should not be displayed using overscan. Instead, they should be displayed using either an exact match between the display area and the conformance cropping window, or using underscan. As used in this paragraph, the term "overscan" refers to display processes in which some parts near the borders of the cropped decoded pictures are not visible in the display area. The term "underscan" describes display processes in which the entire cropped decoded pictures are visible in the display area, but they do not cover the entire display area. For display processes that neither use overscan nor underscan, the display area exactly matches the area of the cropped decoded pictures.

NOTE 2 – For example, `vui_overscan_appropriate_flag` equal to 1 might be used for entertainment television programming or for a live view of people in a videoconference, and `vui_overscan_appropriate_flag` equal to 0 might be used for computer screen capture or security camera content.

vui_colour_description_present_flag equal to 1 specifies that `vui_colour primaries`, `vui_transfer_characteristics`, and `vui_matrix_coeffs` are present. `vui_colour_description_present_flag` equal to 0 specifies that `vui_colour primaries`, `vui_transfer_characteristics`, and `vui_matrix_coeffs` are not present.

vui_colour primaries indicates the chromaticity coordinates of the source colour primaries. Its semantics are as specified for the ColourPrimaries parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When the `vui_colour_primaries` syntax element is not present, the value of `vui_colour_primaries` is inferred to be equal to 2 (the chromaticity is unknown or unspecified or determined by other means not specified in this Specification). Values of `vui_colour_primaries` that are identified as reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of `vui_colour_primaries` as equivalent to the value 2.

vui_transfer_characteristics indicates the transfer characteristics function of the colour representation. Its semantics are as specified for the TransferCharacteristics parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When the `vui_transfer_characteristics` syntax element is not present, the value of `vui_transfer_characteristics` is inferred to be equal to 2 (the transfer characteristics are unknown or unspecified or determined by other means not specified in this Specification). Values of `vui_transfer_characteristics` that are identified as reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of `vui_transfer_characteristics` as equivalent to the value 2.

vui_matrix_coeffs describes the equations used in deriving luma and chroma signals from the green, blue, and red, or Y, Z, and X primaries. Its semantics are as specified for MatrixCoefficients in Rec. ITU-T H.273 | ISO/IEC 23091-2.

`vui_matrix_coeffs` shall not be equal to 0 unless both of the following conditions are true:

- `BitDepthC` is equal to `BitDepthY`.
- `ChromaFormatIdc` is equal to 3 (the 4:4:4 chroma format).

The specification of the use of `vui_matrix_coeffs` equal to 0 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

`vui_matrix_coeffs` shall not be equal to 8 unless one of the following conditions is true:

- `BitDepthC` is equal to `BitDepthY`,
- `BitDepthC` is equal to `BitDepthY + 1` and `ChromaFormatIdc` is equal to 3 (the 4:4:4 chroma format).

The specification of the use of `vui_matrix_coeffs` equal to 8 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

When the `vui_matrix_coeffs` syntax element is not present, the value of `vui_matrix_coeffs` is inferred to be equal to 2 (unknown or unspecified or determined by other means not specified in this Specification).

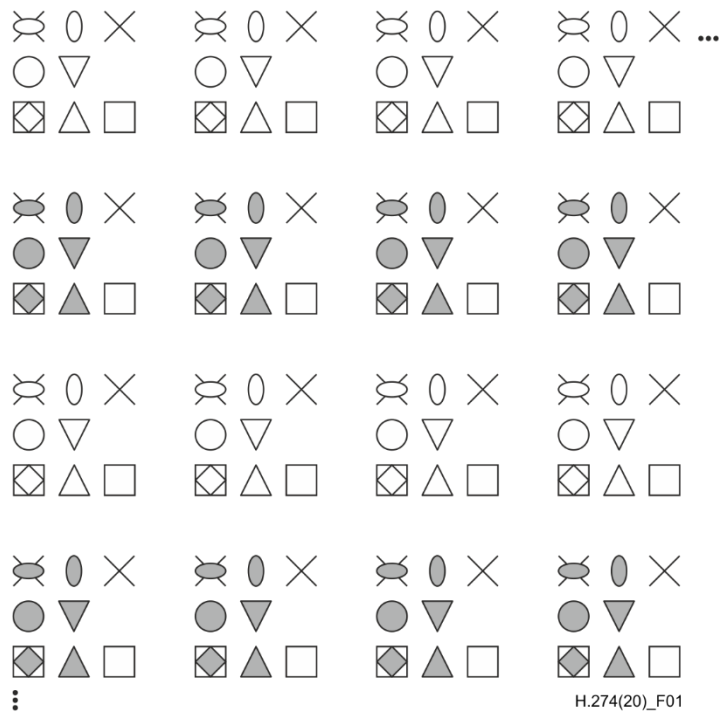
vui_full_range_flag indicates the scaling and offset values applied in association with the matrix coefficients. Its semantics are as specified for the VideoFullRangeFlag parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When not present, the value of `vui_full_range_flag` is inferred to be equal to 0.

vui_chroma_loc_info_present_flag equal to 1 specifies that either `vui_chroma_sample_loc_type_frame` or both `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` are present. `vui_chroma_loc_info_present_flag` equal to 0 specifies that `vui_chroma_sample_loc_type_frame`, `vui_chroma_sample_loc_type_top_field`, and `vui_chroma_sample_loc_type_bottom_field` are not present.

When `ChromaFormatIdc` is not equal to 1, `vui_chroma_loc_info_present_flag` should be equal to 0.

vui_chroma_sample_loc_type_frame, **vui_chroma_sample_loc_type_top_field**, and **vui_chroma_sample_loc_type_bottom_field**, when present, specify the location of chroma samples as follows:

- If `GeneralProgressiveSourceFlag` is equal to 1, `GeneralInterlacedSourceFlag` is equal to 0, and `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format), `vui_chroma_sample_loc_type_frame` specifies the location of chroma samples for both fields of each frame of the CLVS as shown in Figure 1.
- Otherwise, if `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format), `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` specify the location of chroma samples for each top field and bottom field of the CLVS, respectively, as shown in Figure 1.
- Otherwise (`ChromaFormatIdc` is not equal to 1), the values of the syntax elements `chroma_sample_loc_type`, `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` shall be ignored.



Interpretation of symbols

Luma sample position indications:

⊗ Luma sample top field □ Luma sample bottom field

Chroma sample position indications, where gray fill indicates a bottom field sample type and no fill indicates a top field sample type:

○ Chroma sample type 2 ○ Chroma sample type 3
 ○ Chroma sample type 0 ▽ Chroma sample type 1
 ◇ Chroma sample type 4 △ Chroma sample type 5

Figure 1 – Location of chroma samples for top and bottom fields for ChromaFormatIdc equal to 1 (4:2:0 chroma format) as a function of vui_chroma_sample_loc_type_top_field and vui_chroma_sample_loc_type_bottom_field in the range of 0 to 5, inclusive

When ChromaFormatIdc is equal to 2 (4:2:2 chroma format), the nominal positions of the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a picture are as shown in Figure 2.

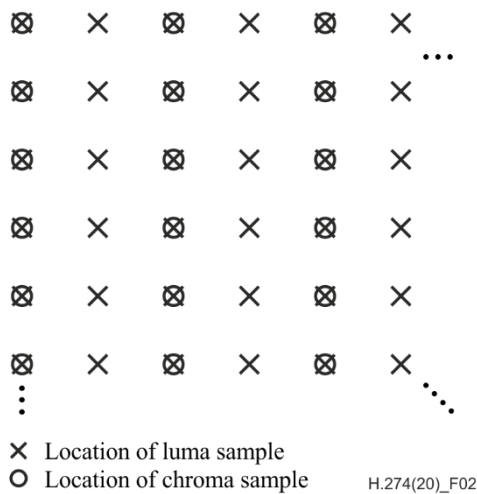


Figure 2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture

When ChromaFormatIdc is equal to 3 (4:4:4 chroma format), the nominal positions of the chroma samples are such that all array samples are co-sited for all cases of pictures and the nominal locations in a picture are as shown in Figure 3.

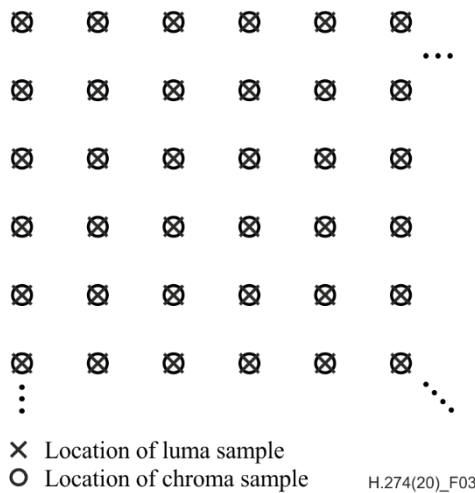


Figure 3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture

When ChromaFormatIdc is equal to 0, there is no chroma sample array.

When present, the values of `vui_chroma_sample_loc_type_frame`, `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` shall be in the range of 0 to 6, inclusive.

When ChromaFormatIdc is equal to 1 and `vui_chroma_loc_info_present_flag` is equal to 0, `vui_chroma_sample_loc_type_frame` is not present and is inferred to be equal to 6, which indicates that the location of the chroma samples is unknown or unspecified or specified by other means not specified in this Specification. When `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` are not present, the values of `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` are inferred to be equal to `vui_chroma_sample_loc_type_frame`.

NOTE 3 – In Rec. ITU-T H.266 | ISO/IEC 23090-3 and Rec. ITU-T H.265 | ISO/IEC 23008-2, a nominal chroma sampling type is identified for ChromaFormatIdc equal to 1 that corresponds to `vui_chroma_sample_loc_type_frame`, `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` equal to 0.

Figure 4 illustrates the indicated relative position of the top-left chroma sample when ChromaFormatIdc is equal to 1 (i.e., the 4:2:0 chroma format), and `vui_chroma_sample_loc_type_top_field` or `vui_chroma_sample_loc_type_bottom_field` is equal to the value of a variable Chroma420LocType. The region represented by the top-left 4:2:0 chroma sample (depicted as a large grey, solid-line square with a large grey dot at its centre) is shown relative to the region represented by the top-left luma sample (depicted as a small black square with a small black dot at its centre). The regions represented by neighbouring luma samples are depicted as small grey, dotted-line squares with small grey dots at their centres.

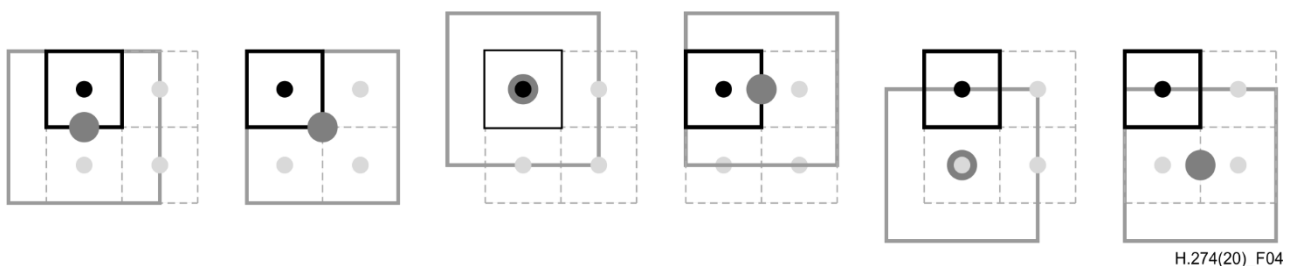


Figure 4 – Location of the top-left chroma sample when ChromaFormatIdc is equal to 1 (4:2:0 chroma format) and Chroma420LocType is equal to 0 to 5, inclusive, from left to right

The relative spatial positioning of the chroma samples, as illustrated in Figure 5, can be expressed by defining two variables `HorizontalOffsetC` and `VerticalOffsetC` as a function of `ChromaFormatIdc` and the variable `Chroma420LocType` as given by Table 3, where `HorizontalOffsetC` is the horizontal (x) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples and `VerticalOffsetC` is the vertical (y) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples.

In a typical FIR filter design, when ChromaFormatIdc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), HorizontalOffsetC and VerticalOffsetC would serve as the phase offsets for the horizontal and vertical filter operations, respectively, for separable downsampling from 4:4:4 chroma format to the chroma format indicated by ChromaFormatIdc.

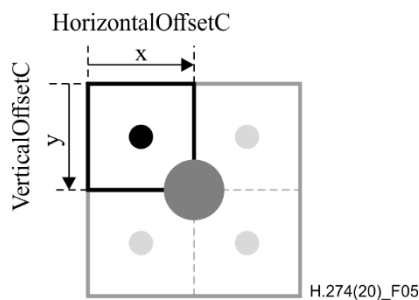


Figure 5 – Location of the top-left chroma sample when ChromaFormatIdc is equal to 1 (4:2:0 chroma format) when Chroma420LocType is equal to 1

Table 3 – Definition of HorizontalOffsetC and VerticalOffsetC as a function of ChromaFormatIdc and Chroma420LocType

ChromaFormatIdc	Chroma420LocType	HorizontalOffsetC	VerticalOffsetC
1 (4:2:0)	0	0	0.5
1 (4:2:0)	1	0.5	0.5
1 (4:2:0)	2	0	0
1 (4:2:0)	3	0.5	0
1 (4:2:0)	4	0	1
1 (4:2:0)	5	0.5	1
2 (4:2:2)	–	0	0
3 (4:4:4)	–	0	0

When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) and the decoded video content is intended for interpretation according to Rec. ITU-R BT.2020 or Rec. ITU-R BT.2100, vui_chroma_loc_info_present_flag should be equal to 1, and vui_chroma_sample_loc_type_frame, vui_chroma_sample_loc_type_top_field, and vui_chroma_sample_loc_type_bottom_field (as applicable) should be equal to 2.

8 SEI messages

8.1 General

Clause 8 specifies the syntax and semantics for SEI messages.

For SEI messages for which the specified syntax structure is empty, such as the dependent random access point SEI message, the mere indication that the SEI message is present (e.g., as indicated by a payload type indicator) is sufficient to convey the associated information (e.g., by indicating that a set of specified constraints are fulfilled).

The semantics and persistence scope for each SEI message are specified in the semantics specification for each particular SEI message.

NOTE – Persistence information for SEI messages is summarized in Table 4.

Table 4 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Filler payload	The PU containing the SEI message
User data registered by Rec. ITU-T T.35	Unspecified
User data unregistered	Unspecified
Film grain characteristics	Specified by the syntax of the SEI message
Frame packing arrangement	Specified by the syntax of the SEI message
Parameter sets inclusion indication	The CLVS containing the SEI message

Table 4 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Decoded picture hash	The PU containing the SEI message
Mastering display colour volume	The CLVS containing the SEI message
Content light level information	The CLVS containing the SEI message
DRAP indication	The picture associated with the SEI message
Alternative transfer characteristics	The CLVS containing the SEI message
Ambient viewing environment	The CLVS containing the SEI message
Content colour volume	Specified by the syntax of the SEI message
Equiangular projection	Specified by the syntax of the SEI message
Generalized cubemap projection	Specified by the syntax of the SEI message
Sphere rotation	Specified by the syntax of the SEI message
Region-wise packing	Specified by the syntax of the SEI message
Omnidirectional viewport	Specified by the syntax of the SEI message
Frame-field information	The PU containing the SEI message
Sample aspect ratio information	Specified by the syntax of the SEI message
Annotated regions	Specified by the syntax of the SEI message
Scalability dimension information	The CVS containing the SEI message
Multiview acquisition information	The CVS containing the SEI message
Multiview view position	The CVS containing the SEI message
Depth representation information	Specified by the semantics of the SEI message
Alpha channel information	Specified by the syntax of the SEI message
Extended DRAP indication	The picture associated with the SEI message
Display orientation	Specified by the syntax of the SEI message
Colour transform information	Specified by the syntax of the SEI message
Shutter interval information	The CLVS containing the SEI message
Neural-network post-filter characteristics	The CLVS containing the SEI message
Neural-network post-filter activation	Specified by the syntax of the SEI message
Phase indication	Specified by the semantics of the SEI message

In the semantics of a particular SEI message, the phrase "the current layer" in the semantics refer to the layer that the particular SEI message is associated with, the phrase "the current picture" refer to the picture that the particular SEI message is associated with, and the phrase "the current CLVS" or "the CLVS" refers to the CLVS containing the current picture. The association of an SEI message to a layer or a picture is specified in a video coding specification that specifies a coded video bitstream with which the SEI messages are used.

The values of some SEI message syntax elements, including `fp_arrangement_id` and `omni_viewport_id`, are split into two sets of value ranges, where the first set is specified as "may be used as determined by the application", and the second set is specified as "reserved for future use by ITU-T | ISO/IEC". Applications should be cautious of potential "collisions" of the interpretation for values of these syntax elements belonging to the first set of value ranges. Since different applications might use these IDs having values in the first set of value ranges for different purposes, particular care should be exercised in the design of encoders that generate SEI messages with these IDs having values in the first set of value ranges, and in the design of decoders that interpret SEI messages with these IDs. This Specification does not define any management for these values. These IDs having values in the first set of value ranges might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the syntax and semantics of an SEI message with one of these IDs having the same value in the first set of value ranges) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

Some SEI messages include a persistency cancel flag in the syntax. For example, the film grain characteristics SEI message includes the `fg_characteristics_cancel_flag`. Regardless of whether such an SEI message is included in an NAL unit that precedes or succeeds the VCL NAL units of the current picture (i.e., the picture associated with the SEI message), the current picture associated with the SEI message is not included in the persistence established by any previous SEI message of that type in decoding order.

8.2 Filler payload SEI message

8.2.1 Filler payload SEI message syntax

	Descriptor
filler_payload(payloadSize) {	
for(k = 0; k < payloadSize; k++)	
ff_byte /* equal to 0xFF */	f(8)
}	

8.2.2 Filler payload SEI message semantics

This SEI message contains a series of payloadSize bytes of value 0xFF, which can be discarded.

ff_byte shall be a byte having the value 0xFF.

8.3 User data registered by Rec. ITU-T T.35 SEI message

8.3.1 User data registered by Rec. ITU-T T.35 SEI message syntax

	Descriptor
user_data_registered_itu_t_t35(payloadSize) {	
itu_t_t35_country_code	b(8)
if(itu_t_t35_country_code != 0xFF)	
i = 1	
else {	
itu_t_t35_country_code_extension_byte	b(8)
i = 2	
}	
do {	
itu_t_t35_payload_byte	b(8)
i++	
} while(i < payloadSize)	
}	

8.3.2 User data registered by Rec. ITU-T T.35 SEI message semantics

This SEI message contains user data registered as specified in Rec. ITU-T T.35, the contents of which are not specified in this Specification.

itu_t_t35_country_code shall be a byte having a value specified as a country code by Rec. ITU-T T.35:2000, Annex A.

itu_t_t35_country_code_extension_byte shall be a byte having a value specified as a country code by Rec. ITU-T T.35:2000, Annex B.

itu_t_t35_payload_byte shall be a byte containing data registered as specified in Rec. ITU-T T.35.

The ITU-T T.35 terminal provider code and terminal provider oriented code shall be contained in the first one or more bytes of the **itu_t_t35_payload_byte**, in the format specified by the Administration that issued the terminal provider code. Any remaining **itu_t_t35_payload_byte** data shall be data having syntax and semantics as specified by the entity identified by the ITU-T T.35 country code and terminal provider code.

8.4 User data unregistered SEI message

8.4.1 User data unregistered SEI message syntax

	Descriptor
user_data_unregistered(payloadSize) {	
uuid_iso_iec_11578	u(128)

for(i = 16; i < payloadSize; i++)	
user_data_payload_byte	b(8)
}	

8.4.2 User data unregistered SEI message semantics

This SEI message contains unregistered user data identified by a universal unique identifier (UUID), the contents of which are not specified in this Specification.

uuid_iso_iec_11578 shall have a value specified as a UUID according to the procedures of ISO/IEC 11578:1996, Annex A.

user_data_payload_byte shall be a byte containing data having syntax and semantics as specified by the UUID generator.

8.5 Film grain characteristics SEI message

8.5.1 Film grain characteristics SEI message syntax

	Descriptor
film_grain_characteristics(payloadSize) {	
fg_characteristics_cancel_flag	u(1)
if(!fg_characteristics_cancel_flag) {	
fg_model_id	u(2)
fg_separate_colour_description_present_flag	u(1)
if(fg_separate_colour_description_present_flag) {	
fg_bit_depth_luma_minus8	u(3)
fg_bit_depth_chroma_minus8	u(3)
fg_full_range_flag	u(1)
fg_colour_primaries	u(8)
fg_transfer_characteristics	u(8)
fg_matrix_coeffs	u(8)
}	
fg_blending_mode_id	u(2)
fg_log2_scale_factor	u(4)
for(c = 0; c < 3; c++)	
fg_comp_model_present_flag[c]	u(1)
for(c = 0; c < 3; c++)	
if(fg_comp_model_present_flag[c]) {	
fg_num_intensity_intervals_minus1[c]	u(8)
fg_num_model_values_minus1[c]	u(3)
for(i = 0; i <= fg_num_intensity_intervals_minus1[c]; i++) {	
fg_intensity_interval_lower_bound[c][i]	u(8)
fg_intensity_interval_upper_bound[c][i]	u(8)
for(j = 0; j <= fg_num_model_values_minus1[c]; j++)	
fg_comp_model_value[c][i][j]	se(v)
}	
}	
fg_characteristics_persistence_flag	u(1)
}	
}	

8.5.2 Film grain 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.

NOTE 1 – For example, an encoder could use the film grain characteristics SEI message to characterize film grain that was present in the original source video material and was removed by pre-processing filtering techniques. Synthesis of simulated film grain on the input images, which could be the decoded pictures or converted from the decoded pictures, for the display process is optional and does not necessarily exactly follow the specified semantics of the film grain characteristics SEI message. When synthesis of simulated film grain on the input images for the display process is performed, there is no requirement that the method by which the synthesis is performed be the same as the parameterized model for the film grain as provided in the film grain characteristics SEI message.

NOTE 2 – The display process is not specified in this Specification.

NOTE 3 – SMPTE RDD 5 (2006) specifies a film grain simulator based on the information provided in the film grain characteristics 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.
- When the syntax element `fg_separate_colour_description_present_flag` of the film grain 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 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.

`fg_characteristics_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous film grain characteristics SEI message in output order that applies to the current layer. `fg_characteristics_cancel_flag` equal to 0 indicates that film grain modelling information follows.

`fg_model_id` identifies the film grain simulation model as specified in Table 5. The value of `fg_model_id` shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for `fg_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 characteristic SEI messages with `fg_model_id` equal to 2 or 3.

Table 5 – fg_model_id values

Value	Description
0	Frequency filtering
1	Auto-regression

`fg_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 characteristics SEI message syntax. `fg_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 `fg_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, `fg_separate_colour_description_present_flag` shall be equal to 1.

The input image \hat{I} , 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 \hat{I} 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 `fg_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 \hat{I} used in the equations in this clause is also greater than `BitDepthY`. In such a case, the input image \hat{I} 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.

fg_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 `fg_bit_depth_luma_minus8` is not present in the film grain characteristics SEI message, the value of `fg_bit_depth_luma_minus8` is inferred to be equal to `BitDepthY - 8`.

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

$$\text{fgBitDepth}[0] = \text{fg_bit_depth_luma_minus8} + 8 \quad (19)$$

fg_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 `fg_bit_depth_chroma_minus8` is not present in the film grain characteristics SEI message, the value of `fg_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:

$$\text{fgBitDepth}[c] = \text{fg_bit_depth_chroma_minus8} + 8, \text{ with } c = 1, 2 \quad (20)$$

fg_full_range_flag has the same semantics as specified in clause 7.3 for the `vui_full_range_flag` syntax element, except as follows:

- `fg_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 `fg_full_range_flag` is not present in the film grain characteristics SEI message, the value of `fg_full_range_flag` is inferred to be equal to `vui_full_range_flag`.

fg_colour_primaries has the same semantics as specified in clause 7.3 for the `vui_colour_primaries` syntax element, except as follows:

- `fg_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 `fg_colour_primaries` is not present in the film grain characteristics SEI message, the value of `fg_colour_primaries` is inferred to be equal to `vui_colour_primaries`.

fg_transfer_characteristics has the same semantics as specified in clause 7.3 for the `vui_transfer_characteristics` syntax element, except as follows:

- `fg_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 `fg_transfer_characteristics` is not present in the film grain characteristics SEI message, the value of `fg_transfer_characteristics` is inferred to be equal to `vui_transfer_characteristics`.

fg_matrix_coeffs has the same semantics as specified in clause 7.3 for the `vui_matrix_coeffs` syntax element, except as follows:

- `fg_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 `fg_matrix_coeffs` is not present in the film grain characteristics SEI message, the value of `fg_matrix_coeffs` is inferred to be equal to `vui_matrix_coeffs`.
- The values allowed for `fg_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.

fg_blending_mode_id identifies the blending mode used to blend the simulated film grain with the input images as specified in Table 6. `fg_blending_mode_id` shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for

fg_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 fg_blending_mode_id equal to 2 or 3.

Table 6 – fg_blending_mode_id values

Value	Description
0	Additive
1	Multiplicative

Depending on the value of fg_blending_mode_id, the blending mode is specified as follows:

- If fg_blending_mode_id is equal to 0, the blending mode is additive as specified by:

$$I_{\text{grain}}[c][x][y] = \text{Clip3}(0, (1 \ll \text{fgBitDepth}[c]) - 1, \hat{I}[c][x][y] + G[c][x][y]) \quad (21)$$

- Otherwise (fg_blending_mode_id is equal to 1), the blending mode is multiplicative as specified by:

$$I_{\text{grain}}[c][x][y] = \text{Clip3}(0, (1 \ll \text{fgBitDepth}[c]) - 1, \hat{I}[c][x][y] + \text{Round}((\hat{I}[c][x][y] * G[c][x][y]) \div ((1 \ll \text{fgBitDepth}[c]) - 1))) \quad (22)$$

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

fg_log2_scale_factor specifies a scale factor used in the film grain characterization equations.

fg_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. **fg_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 **fg_separate_colour_description_present_flag** is equal to 0 and **ChromaFormatIdc** is equal to 0, the value of **fg_comp_model_present_flag[1]** and **fg_comp_model_present_flag[2]** shall be equal to 0.

fg_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.

fg_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 **fg_num_model_values_minus1[c]** shall be in the range of 0 to 5, inclusive.

fg_intensity_interval_lower_bound[c][i] specifies the lower bound of the i -th intensity interval for which the set of model values applies.

fg_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 $\hat{I}[c][x][y]$ for $c = 0..2$, $x = 0..\text{PicWidthInLumaSamples} - 1$, $y = 0..\text{PicHeightInLumaSamples} - 1$, and $j = 0..\text{numApplicableIntensityIntervals}[c][x][y] - 1$, where **numApplicableIntensityIntervals[c][x][y]** is derived below.

Depending on the value of **fg_model_id**, the selection of the one or more intensity intervals for the sample value $\hat{I}[c][x][y]$ is specified as follows:

- The variable **numApplicableIntensityIntervals[c][x][y]** is initially set equal to 0.
- If **fg_model_id** is equal to 0, the following applies:
 - The top-left sample location (x_B, y_B) of the current 8×8 block b that contains the sample value $\hat{I}[c][x][y]$ is derived as $(x_B, y_B) = (x / 8, y / 8)$.

- The average value b_{avg} of the current 8x8 block b is derived as follows:

$$\begin{aligned}
 & \text{sum8x8} = 0 \\
 & \text{for}(i = 0; i < 8; i++) \\
 & \quad \text{for}(j = 0; j < 8; j++) \\
 & \quad \quad \text{sum8x8} += \hat{I}[c][xB * 8 + i][yB * 8 + j] \\
 & b_{avg} = \text{Clip3}(0, 255, (\text{sum8x8} + (1 \ll (\text{fgBitDepth}[c] - 3))) \gg (\text{fgBitDepth}[c] - 2))
 \end{aligned} \tag{23}$$

- The value of $\text{intensityIntervalIdx}[c][x][y][j]$ is derived as follows:

$$\begin{aligned}
 & \text{for}(i = 0, j = 0; i \leq \text{fg_num_intensity_intervals_minus1}[c]; i++) \\
 & \quad \text{if}(b_{avg} \geq \text{fg_intensity_interval_lower_bound}[c][i] \ \&\& \\
 & \quad \quad b_{avg} \leq \text{fg_intensity_interval_upper_bound}[c][i]) \{ \\
 & \quad \quad \quad \text{intensityIntervalIdx}[c][x][y][j] = i \\
 & \quad \quad \quad j++ \\
 & \quad \quad \} \\
 & \text{numApplicableIntensityIntervals}[c][x][y] = j
 \end{aligned} \tag{24}$$

- Otherwise (fg_model_id is equal to 1), the value of $\text{intensityIntervalIdx}[c][x][y][j]$ is derived as follows:

$$\begin{aligned}
 & I_8[c][x][y] = (\text{fgBitDepth}[c] == 8) ? (\hat{I}[c][x][y] : \\
 & \quad \text{Clip3}(0, 255, (\hat{I}[c][x][y] + (1 \ll (\text{fgBitDepth}[c] - 9))) \gg (\text{fgBitDepth}[c] - 8))) \\
 & \text{for}(i = 0, j = 0; i \leq \text{fg_num_intensity_intervals_minus1}[c]; i++) \\
 & \quad \text{if}(I_8[c][x][y] \geq \text{fg_intensity_interval_lower_bound}[c][i] \ \&\& \\
 & \quad \quad I_8[c][x][y] \leq \text{fg_intensity_interval_upper_bound}[c][i]) \{ \\
 & \quad \quad \quad \text{intensityIntervalIdx}[c][x][y][j] = i \\
 & \quad \quad \quad j++ \\
 & \quad \quad \} \\
 & \text{numApplicableIntensityIntervals}[c][x][y] = j
 \end{aligned} \tag{25}$$

Samples that do not fall into any of the defined intervals (i.e., those samples for which the value of $\text{numApplicableIntensityIntervals}[c][x][y]$ 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 $\text{numApplicableIntensityIntervals}[c][x][y]$ 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 s_j in each instance of the list $\text{fg_comp_model_value}[c][s_j]$ is the value of $\text{intensityIntervalIdx}[c][x][y][j]$ derived for the sample value $\hat{I}[c][x][y]$.

fg_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 fg_model_id .

The value of $\text{fg_comp_model_value}[c][i][j]$ is constrained as follows, and could be additionally constrained as specified elsewhere in this clause:

- If fg_model_id is equal to 0, $\text{fg_comp_model_value}[c][i][j]$ shall be in the range of 0 to $2^{\text{fgBitDepth}[c]} - 1$, inclusive.
- Otherwise (fg_model_id is equal to 1), $\text{fg_comp_model_value}[c][i][j]$ shall be in the range of $-2^{(\text{fgBitDepth}[c] - 1)}$ to $2^{(\text{fgBitDepth}[c] - 1)} - 1$, inclusive.

Depending on the value of fg_model_id , the synthesis of the film grain is modelled as follows:

- If fg_model_id is equal to 0, a frequency filtering model enables simulating the original film grain for $c = 0..2$, $x = 0..\text{PicWidthInLumaSamples} - 1$, and $y = 0..\text{PicHeightInLumaSamples} - 1$ as specified by:

$$\begin{aligned}
 & G[c][x][y] = (\text{fg_comp_model_value}[c][s_j][0] * Q[c][x][y] + \text{fg_comp_model_value}[c][s_j][5] * \\
 & \quad G[c - 1][x][y]) \gg \text{fg_log2_scale_factor}
 \end{aligned} \tag{26}$$

where $Q[c]$ is a two-dimensional random process generated by filtering 16x16 blocks gaussRv with random-valued elements gaussRv_{ij} 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 uRv_0 and uRv_1 , using the Box-Muller transformation specified by:

$$\text{gaussRv}_{i,j} = \text{Sqrt}(-2 * \text{Ln}(uRv_0)) * \text{Cos}(2 * \pi * uRv_1) \quad (27)$$

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 < fg_comp_model_value[ c ][ s_j ][ 3 ] && y < fg_comp_model_value[ c ][ s_j ][ 4 ] ) ||
        x > fg_comp_model_value[ c ][ s_j ][ 1 ] || y > fg_comp_model_value[ c ][ s_j ][ 2 ] )
      gaussRv[ x ][ y ] = 0
filteredRv = IDCT16x16( gaussRv )
```

(28)

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

$$\text{IDCT16x16}(z) = r * z * r^T \quad (29)$$

where the superscript T indicates a matrix transposition and r is the 16x16 matrix with elements r_{ij} specified by:

$$r_{i,j} = \frac{((i == 0) ? 1 : \text{Sqrt}(2))}{4} * \text{Cos}\left(\frac{i * (2 * j + 1) * \pi}{32}\right) \quad (30)$$

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 `fg_comp_model_value[c][s_j][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 (`fg_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 ] = ( fg_comp_model_value[ c ][ s_j ][ 0 ] * n[ c ][ x ][ y ] +
  fg_comp_model_value[ c ][ s_j ][ 1 ] * ( G[ c ][ x - 1 ][ y ] +
  ( ( fg_comp_model_value[ c ][ s_j ][ 4 ] * G[ c ][ x ][ y - 1 ] ) >>
  fg_log2_scale_factor ) + fg_comp_model_value[ c ][ s_j ][ 3 ] *
  ( ( fg_comp_model_value[ c ][ s_j ][ 4 ] * ( G[ c ][ x - 1 ][ y - 1 ] + G[ c ][ x + 1 ][ y - 1 ] ) ) >>
  fg_log2_scale_factor ) +
  fg_comp_model_value[ c ][ s_j ][ 5 ] * ( G[ c ][ x - 2 ][ y ] +
  ( ( fg_comp_model_value[ c ][ s_j ][ 4 ] * fg_comp_model_value[ c ][ s_j ][ 4 ] * G[ c ][ x ][ y - 2 ] ) >>
  ( 2 * fg_log2_scale_factor ) ) ) +
  fg_comp_model_value[ c ][ s_j ][ 2 ] * G[ c - 1 ][ x ][ y ] ) >> fg_log2_scale_factor
```

(31)

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.

`fg_comp_model_value[c][i][0]` provides the first model value for the model as specified by `fg_model_id`. `fg_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.

`fg_comp_model_value[c][i][1]` provides the second model value for the model as specified by `fg_model_id`. When `fg_model_id` is equal to 0, `fg_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 characteristics SEI message, `fg_comp_model_value[c][i][1]` is inferred as follows:

- If `fg_model_id` is equal to 0, `fg_comp_model_value[c][i][1]` is inferred to be equal to 8.

- Otherwise (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][1]` is inferred to be equal to 0.

`fg_comp_model_value[c][i][1]` is interpreted as follows:

- If `fg_model_id` is equal to 0, `fg_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 (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][1]` indicates the first order spatial correlation for neighbouring samples $(x - 1, y)$ and $(x, y - 1)$.

`fg_comp_model_value[c][i][2]` provides the third model value for the model as specified by `fg_model_id`. When `fg_model_id` is equal to 0, `fg_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 characteristics SEI message, `fg_comp_model_value[c][i][2]` is inferred as follows:

- If `fg_model_id` is equal to 0, `fg_comp_model_value[c][i][2]` is inferred to be equal to `fg_comp_model_value[c][i][1]`
- Otherwise (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][2]` is inferred to be equal to 0.

`fg_comp_model_value[c][i][2]` is interpreted as follows:

- If `fg_model_id` is equal to 0, `fg_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 (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][2]` indicates the colour correlation between consecutive colour components.

`fg_comp_model_value[c][i][3]` provides the fourth model value for the model as specified by `fg_model_id`. When `fg_model_id` is equal to 0, `fg_comp_model_value[c][i][3]` shall be greater than or equal to 0 and less than or equal to `fg_comp_model_value[c][i][1]`.

When not present in the film grain characteristics SEI message, `fg_comp_model_value[c][i][3]` is inferred to be equal to 0.

`fg_comp_model_value[c][i][3]` is interpreted as follows:

- If `fg_model_id` is equal to 0, `fg_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 (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][3]` indicates the first order spatial correlation for neighbouring samples $(x - 1, y - 1)$ and $(x + 1, y - 1)$.

`fg_comp_model_value[c][i][4]` provides the fifth model value for the model as specified by `fg_model_id`. When `fg_model_id` is equal to 0, `fg_comp_model_value[c][i][4]` shall be greater than or equal to 0 and less than or equal to `fg_comp_model_value[c][i][2]`.

When not present in the film grain characteristics SEI message, `fg_comp_model_value[c][i][4]` is inferred to be equal to `fg_model_id`.

`fg_comp_model_value[c][i][4]` is interpreted as follows:

- If `fg_model_id` is equal to 0, `fg_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 (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][4]` indicates the aspect ratio of the modelled grain.

`fg_comp_model_value[c][i][5]` provides the sixth model value for the model as specified by `fg_model_id`.

When not present in the film grain characteristics SEI message, `fg_comp_model_value[c][i][5]` is inferred to be equal to 0.

`fg_comp_model_value[c][i][5]` is interpreted as follows:

- If `fg_model_id` is equal to 0, `fg_comp_model_value[c][i][5]` indicates the colour correlation between consecutive colour components.
- Otherwise (`fg_model_id` is equal to 1), `fg_comp_model_value[c][i][5]` indicates the second order spatial correlation for neighbouring samples $(x, y - 2)$ and $(x - 2, y)$.

`fg_characteristics_persistence_flag` specifies the persistence of the film grain characteristics SEI message for the current layer.

`fg_characteristics_persistence_flag` equal to 0 specifies that the film grain characteristics SEI message applies to the current decoded picture only.

fg_characteristics_persistence_flag equal to 1 specifies that the film grain 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 characteristics SEI message is output that follows the current picture in output order.

8.6 Frame packing arrangement SEI message

8.6.1 Frame packing arrangement SEI message syntax

frame_packing_arrangement(payloadSize) {	Descriptor
fp_arrangement_id	ue(v)
fp_arrangement_cancel_flag	u(1)
if(!fp_arrangement_cancel_flag) {	
fp_arrangement_type	u(7)
fp_quincunx_sampling_flag	u(1)
fp_content_interpretation_type	u(6)
fp_spatial_flipping_flag	u(1)
fp_frame0_flipped_flag	u(1)
fp_field_views_flag	u(1)
fp_current_frame_is_frame0_flag	u(1)
fp_frame0_self_contained_flag	u(1)
fp_frame1_self_contained_flag	u(1)
if(!fp_quincunx_sampling_flag && fp_arrangement_type != 5) {	
fp_frame0_grid_position_x	u(4)
fp_frame0_grid_position_y	u(4)
fp_frame1_grid_position_x	u(4)
fp_frame1_grid_position_y	u(4)
}	
fp_arrangement_reserved_byte	u(8)
fp_arrangement_persistence_flag	u(1)
}	
fp_upsampled_aspect_ratio_flag	u(1)
}	

8.6.2 Frame packing arrangement SEI message semantics

This SEI message informs the decoder that the cropped decoded picture contains samples of multiple distinct spatially packed constituent frames that are packed into one frame, or that the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames, using an indicated frame packing arrangement scheme. This information can be used by the decoder to appropriately rearrange the samples and process the samples of the constituent frames appropriately for display or other purposes (which are outside the scope of this Specification).

This SEI message may be associated with pictures that are either frames or fields (as determined outside the scope of this Specification). The frame packing arrangement of the samples is specified in terms of the sampling structure of a frame in order to define a frame packing arrangement structure that is invariant with respect to whether a picture is a single field of such a packed frame or is a complete packed frame.

NOTE 1 – The interpretation of frame_packing_arrangement_type is in alignment with the code point specifications in Rec. ITU-T H.273 | ISO/IEC 23091-2. However, more values of frame_packing_arrangement_type are specified in Rec. ITU-T H.273 | ISO/IEC 23091-2 than are specified for use herein.

fp_arrangement_id contains an identifying number that may be used to identify the usage of the frame packing arrangement SEI message. The value of **fp_arrangement_id** shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of **fp_arrangement_id** from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of **fp_arrangement_id** from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **fp_arrangement_id** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

fp_arrangement_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous frame packing arrangement SEI message in output order that applies to the current layer. **fp_arrangement_cancel_flag** equal to 0 indicates that frame packing arrangement information follows.

fp_arrangement_type identifies the indicated interpretation of the sample arrays of the output cropped decoded picture as specified in Table 7.

When **fp_arrangement_type** is equal to 3 or 4, each component plane of the output cropped decoded picture contains all samples (when **ffi_field_pic_flag** is equal to 0) or the samples corresponding to the top or bottom field (when **ffi_field_pic_flag** is equal to 1) of the samples of a frame packing arrangement structure.

Table 7 – Definition of fp_arrangement_type

Value	Interpretation
3	The frame packing arrangement structure contains a side-by-side packing arrangement of corresponding planes of two constituent frames as illustrated in Figure 6, Figure 7 and Figure 10.
4	The frame packing arrangement structure contains a top-bottom packing arrangement of corresponding planes of two constituent frames as illustrated in Figure 8 and Figure 9.
5	The component planes of the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames as illustrated in Figure 11.

NOTE 2 – Figure 6 to Figure 10 provide typical examples of rearrangement and upconversion processing for various packing arrangement schemes. Actual characteristics of the constituent frames are signalled in detail by the subsequent syntax elements of the frame packing arrangement SEI message. In Figure 6 to Figure 10, an upconversion processing is performed on each constituent frame to produce frames having the same resolution as that of the decoded frame. An example of the upsampling method to be applied to a quincunx sampled frame as shown in Figure 10 is to fill in missing positions with an average of the available spatially neighbouring samples (the average of the values of the available samples above, below, to the left and to the right of each sample to be generated). The actual upconversion process to be performed, if any, is outside the scope of this Specification.

NOTE 3 – When the output time of the samples of constituent frame 0 differs from the output time of the samples of constituent frame 1 (i.e., when **fp_field_views_flag** is equal to 1 or **fp_arrangement_type** is equal to 5) and the display system in use presents two views simultaneously, the display time for constituent frame 0 could be delayed to coincide with the display time for constituent frame 1. (The display process is not specified in this Specification.)

NOTE 4 – When **fp_field_views_flag** is equal to 1 or **fp_arrangement_type** is equal to 5, the value 0 for **fixed_pic_rate_within_cvs_flag** is not expected to be prevalent in industry use of this SEI message.

NOTE 5 – **fp_arrangement_type** equal to 5 describes a temporal interleaving process of different views.

All other values of **fp_arrangement_type** are reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that bitstreams conforming to this version of this Specification shall not contain such other values of **fp_arrangement_type**. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of **fp_arrangement_type**.

fp_quincunx_sampling_flag equal to 1 indicates that each colour component plane of each constituent frame is quincunx sampled as illustrated in Figure 10 and **fp_quincunx_sampling_flag** equal to 0 indicates that the colour component planes of each constituent frame are not quincunx sampled.

When **fp_arrangement_type** is equal to 5, it is a requirement of bitstream conformance that **fp_quincunx_sampling_flag** shall be equal to 0.

NOTE 6 – For any chroma format (monochrome, 4:2:0, 4:2:2 or 4:4:4), the luma plane and each chroma plane (as applicable) is quincunx sampled as illustrated in Figure 10 when **fp_quincunx_sampling_flag** is equal to 1.

fp_content_interpretation_type indicates the intended interpretation of the constituent frames as specified in Table 8. Values of **fp_content_interpretation_type** that do not appear in Table 8 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of **fp_content_interpretation_type**.

For each specified frame packing arrangement scheme, there are two constituent frames that are referred to as frame 0 and frame 1.

Table 8 – Definition of fp_content_interpretation_type

Value	Interpretation
0	Unknown or unspecified relationship between the frame packed constituent frames
1	Indicates that the two constituent frames form the left and right views of a stereo view scene, with frame 0 being associated with the left view and frame 1 being associated with the right view
2	Indicates that the two constituent frames form the right and left views of a stereo view scene, with frame 0 being associated with the right view and frame 1 being associated with the left view

NOTE 7 – The value 2 for fp_content_interpretation_type is not expected to be prevalent in industry use of this SEI message. However, the value was specified herein for purposes of completeness.

fp_spatial_flipping_flag equal to 1, when fp_arrangement_type is equal to 3 or 4, indicates that one of the two constituent frames is spatially flipped relative to its intended orientation for display or other such purposes.

When fp_arrangement_type is equal to 3 or 4 and fp_spatial_flipping_flag is equal to 1, the type of spatial flipping that is indicated is as follows:

- If fp_arrangement_type is equal to 3, the indicated spatial flipping is horizontal flipping.
- Otherwise (fp_arrangement_type is equal to 4), the indicated spatial flipping is vertical flipping.

When fp_arrangement_type is not equal to 3 or 4, it is a requirement of bitstream conformance that fp_spatial_flipping_flag shall be equal to 0. When fp_arrangement_type is not equal to 3 or 4, the value 1 for fp_spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When fp_arrangement_type is not equal to 3 or 4, decoders shall ignore the value 1 for fp_spatial_flipping_flag.

fp_frame0_flipped_flag, when fp_spatial_flipping_flag is equal to 1, indicates which one of the two constituent frames is flipped.

When fp_spatial_flipping_flag is equal to 1, fp_frame0_flipped_flag equal to 0 indicates that frame 0 is not spatially flipped and frame 1 is spatially flipped and fp_frame0_flipped_flag equal to 1 indicates that frame 0 is spatially flipped and frame 1 is not spatially flipped.

When fp_spatial_flipping_flag is equal to 0, it is a requirement of bitstream conformance that fp_frame0_flipped_flag shall be equal to 0. When fp_spatial_flipping_flag is equal to 0, the value 1 for fp_spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When fp_spatial_flipping_flag is equal to 0, decoders shall ignore the value of fp_frame0_flipped_flag.

fp_field_views_flag equal to 1 indicates that all pictures in the current CLVS are coded as fields, all fields of a particular parity are considered a first constituent frame and all fields of the opposite parity are considered a second constituent frame. It is a requirement of bitstream conformance that the fp_field_views_flag shall be equal to 0, the value 1 for fp_field_views_flag is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of fp_field_views_flag.

fp_current_frame_is_frame0_flag equal to 1, when fp_arrangement is equal to 5, indicates that the current decoded frame is constituent frame 0 and the next decoded frame in output order is constituent frame 1 and the display time of the constituent frame 0 should be delayed to coincide with the display time of constituent frame 1. fp_current_frame_is_frame0_flag equal to 0, when fp_arrangement is equal to 5, indicates that the current decoded frame is constituent frame 1 and the previous decoded frame in output order is constituent frame 0 and the display time of the constituent frame 1 should not be delayed for purposes of stereo-view pairing.

When fp_arrangement_type is not equal to 5, the constituent frame associated with the upper-left sample of the decoded frame is considered to be constituent frame 0 and the other constituent frame is considered to be constituent frame 1. When fp_arrangement_type is not equal to 5, it is a requirement of bitstream conformance that fp_current_frame_is_frame0_flag shall be equal to 0. When fp_arrangement_type is not equal to 5, the value 1 for fp_current_frame_is_frame0_flag is reserved for future use by ITU-T | ISO/IEC. When fp_arrangement_type is not equal to 5, decoders shall ignore the value of fp_current_frame_is_frame0_flag.

fp_frame0_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 0 of the CLVS refer to samples of any constituent frame 1. fp_frame0_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 0 of the CLVS might or might not refer to samples of some constituent frame 1. Within a CLVS, the value of fp_frame0_self_contained_flag in all frame packing arrangement SEI messages shall be the same.

fp_frame1_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 1 of the CLVS refer to samples of any constituent frame 0.

`fp_frame1_self_contained_flag` equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 1 of the CLVS might or might not refer to samples of some constituent frame 0. Within a CLVS, the value of `fp_frame1_self_contained_flag` in all frame packing arrangement SEI messages shall be the same.

When `fp_quincunx_sampling_flag` is equal to 0 and `fp_arrangement_type` is not equal to 5, two (*x*, *y*) coordinate pairs are specified to determine the indicated luma sampling grid alignment for constituent frame 0 and constituent frame 1, relative to the upper left corner of the rectangular area represented by the samples of the corresponding constituent frame.

NOTE 8 – The location of chroma samples relative to luma samples could be indicated by the `vui_chroma_sample_loc_type_frame` or `vui_chroma_sample_loc_type_top_field` and `vui_chroma_sample_loc_type_bottom_field` syntax elements in the VUI parameters, when present.

`fp_frame0_grid_position_x` (when present) specifies the *x* component of the (*x*, *y*) coordinate pair for constituent frame 0.

`fp_frame0_grid_position_y` (when present) specifies the *y* component of the (*x*, *y*) coordinate pair for constituent frame 0.

`fp_frame1_grid_position_x` (when present) specifies the *x* component of the (*x*, *y*) coordinate pair for constituent frame 1.

`fp_frame1_grid_position_y` (when present) specifies the *y* component of the (*x*, *y*) coordinate pair for constituent frame 1.

When `fp_quincunx_sampling_flag` is equal to 0 and `fp_arrangement_type` is not equal to 5 the (*x*, *y*) coordinate pair for each constituent frame is interpreted as follows:

- If the (*x*, *y*) coordinate pair for a constituent frame is equal to (0, 0), this indicates a default sampling grid alignment specified as follows:
 - If `fp_arrangement_type` is equal to 3, the indicated position is the same as for the (*x*, *y*) coordinate pair value (4, 8), as illustrated in Figure 6.
 - Otherwise (`fp_arrangement_type` is equal to 4), the indicated position is the same as for the (*x*, *y*) coordinate pair value (8, 4), as illustrated in Figure 8.
- Otherwise, if the (*x*, *y*) coordinate pair for a constituent frame is equal to (15, 15), this indicates that the sampling grid alignment is unknown or unspecified or specified by other means not specified in this Specification.
- Otherwise, the *x* and *y* elements of the (*x*, *y*) coordinate pair specify the indicated horizontal and vertical sampling grid alignment positioning to the right of and below the upper left corner of the rectangular area represented by the corresponding constituent frame, respectively, in units of one sixteenth of the luma sample grid spacing between the samples of the columns and rows of the constituent frame that are present in the decoded frame (prior to any upsampling for display or other purposes).

NOTE 9 – The spatial location reference information `fp_frame0_grid_position_x`, `fp_frame0_grid_position_y`, `fp_frame1_grid_position_x`, and `fp_frame1_grid_position_y` is not provided when `fp_quincunx_sampling_flag` is equal to 1 because the spatial alignment in this case is assumed to be such that constituent frame 0 and constituent frame 1 cover corresponding spatial areas with interleaved quincunx sampling patterns as illustrated in Figure 10.

`fp_arrangement_reserved_byte` is reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that the value of `fp_arrangement_reserved_byte` shall be equal to 0. All other values of `fp_arrangement_reserved_byte` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `fp_arrangement_reserved_byte`.

`fp_arrangement_persistence_flag` specifies the persistence of the frame packing arrangement SEI message for the current layer.

`fp_arrangement_persistence_flag` equal to 0 specifies that the frame packing arrangement SEI message applies to the current decoded frame only.

`fp_arrangement_persistence_flag` equal to 1 specifies that the frame packing arrangement SEI message applies to the current decoded picture and persists 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 frame packing arrangement SEI message is output that follows the current picture in output order.

fp_upsampled_aspect_ratio_flag equal to 1 indicates that the SAR indicated by the VUI parameters or the SARI SEI message identifies the SAR of the samples after the application of an upconversion process to produce a higher resolution frame from each constituent frame as illustrated in Figure 6 to Figure 10. **fp_upsampled_aspect_ratio_flag** equal to 0 indicates that the SAR indicated by the VUI parameters or the SARI SEI message identifies the SAR of the samples before the application of any such upconversion process.

NOTE 10 – The SAR indicated in the VUI parameters or the SARI SEI message could indicate the preferred display picture shape for the packed decoded frame output by a decoder that does not interpret the frame packing arrangement SEI message. When **fp_upsampled_aspect_ratio_flag** is equal to 1, the SAR produced in each up-converted colour plane is indicated to be the same as the SAR indicated in the VUI parameters or the SARI SEI message in the examples shown in Figure 6 to Figure 10. When **fp_upsampled_aspect_ratio_flag** is equal to 0, the SAR produced in each colour plane prior to upconversion is indicated to be the same as the SAR indicated in the VUI parameters or the SARI SEI message in the examples shown in Figure 6 to Figure 10.

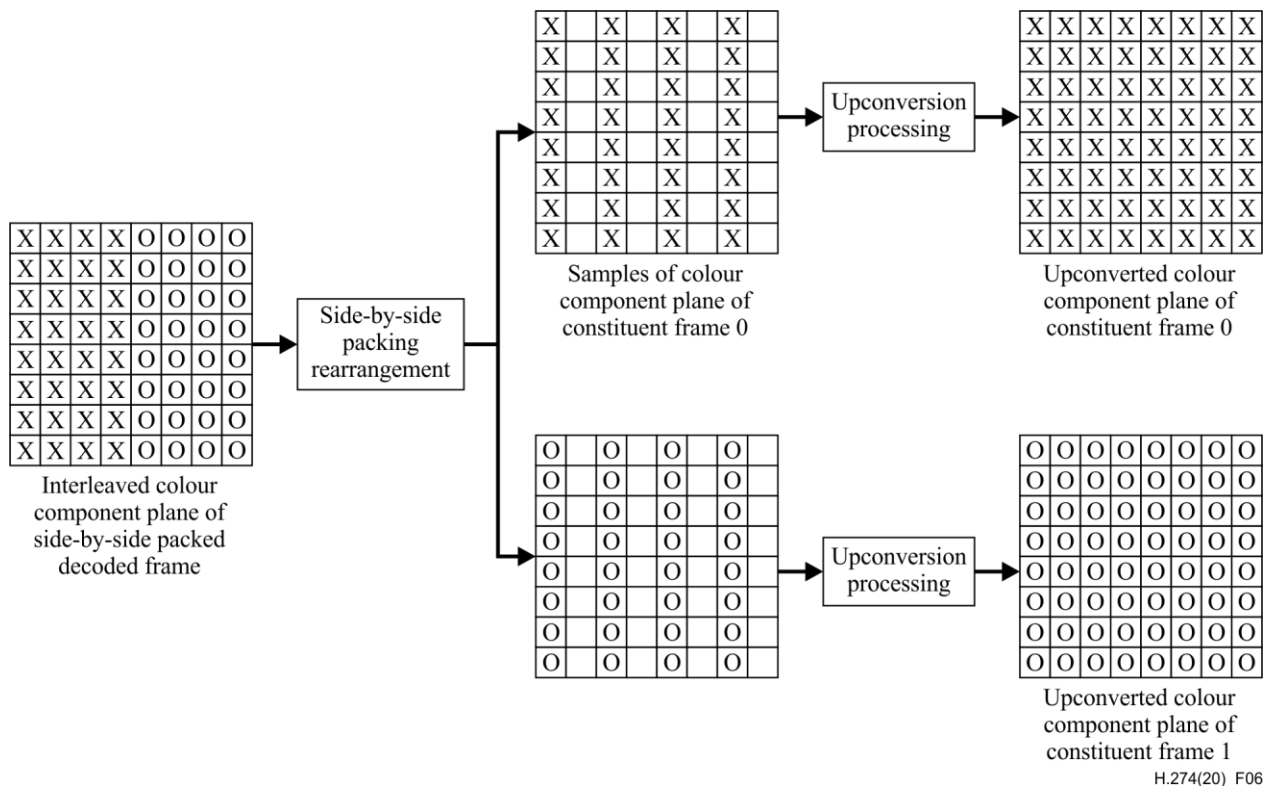


Figure 6 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with **fp_arrangement_type equal to 3, **fp_quincunx_sampling_flag** equal to 0 and (x, y) equal to (0, 0) or (4, 8) for both constituent frames**

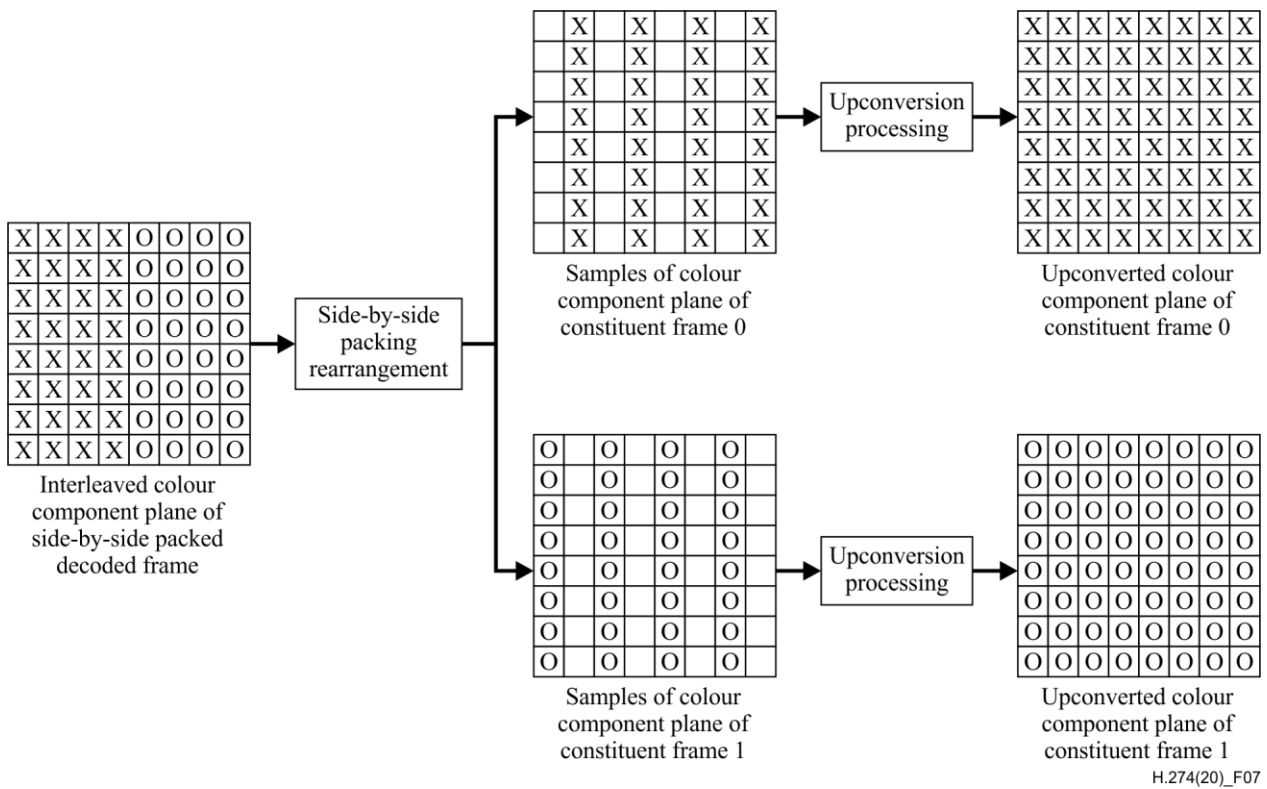


Figure 7 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with $fp_arrangement_type$ equal to 3, $fp_quincunx_sampling_flag$ equal to 0, (x, y) equal to $(12, 8)$ for constituent frame 0 and (x, y) equal to $(0, 0)$ or $(4, 8)$ for constituent frame 1

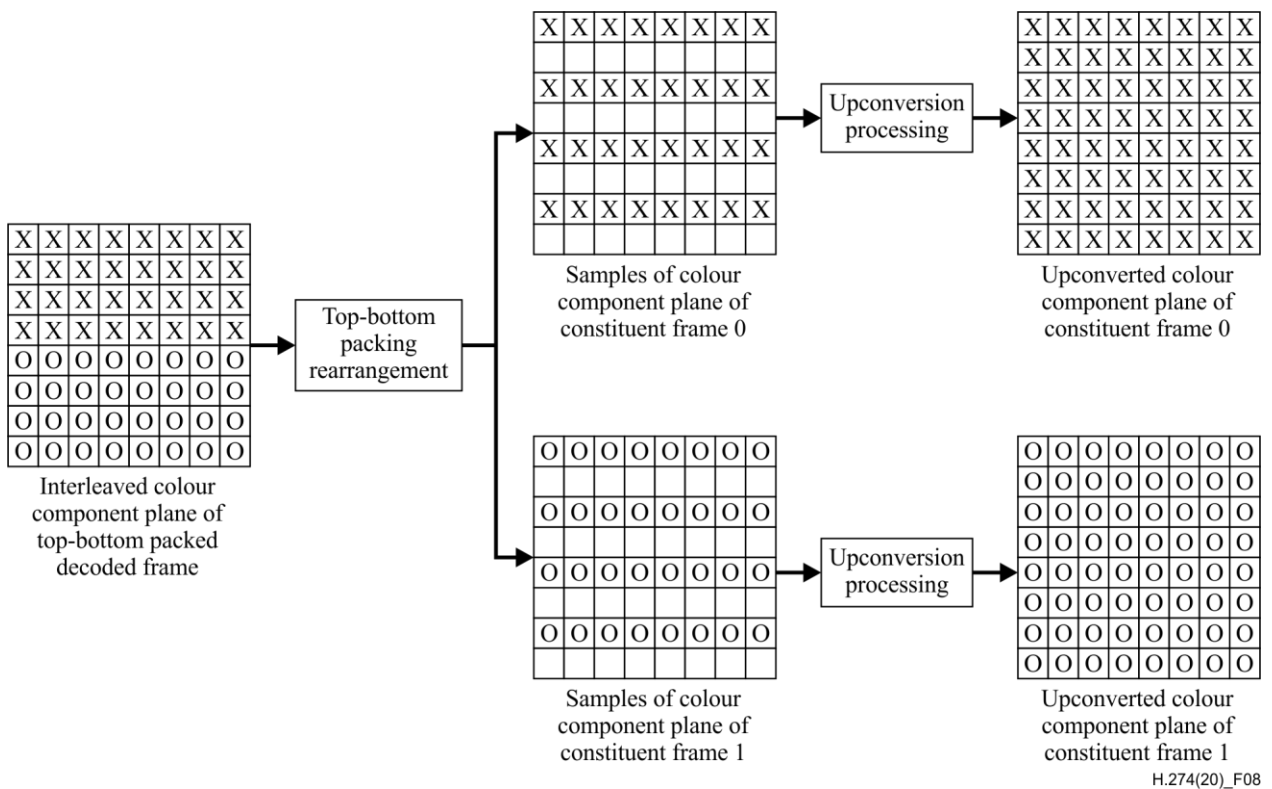
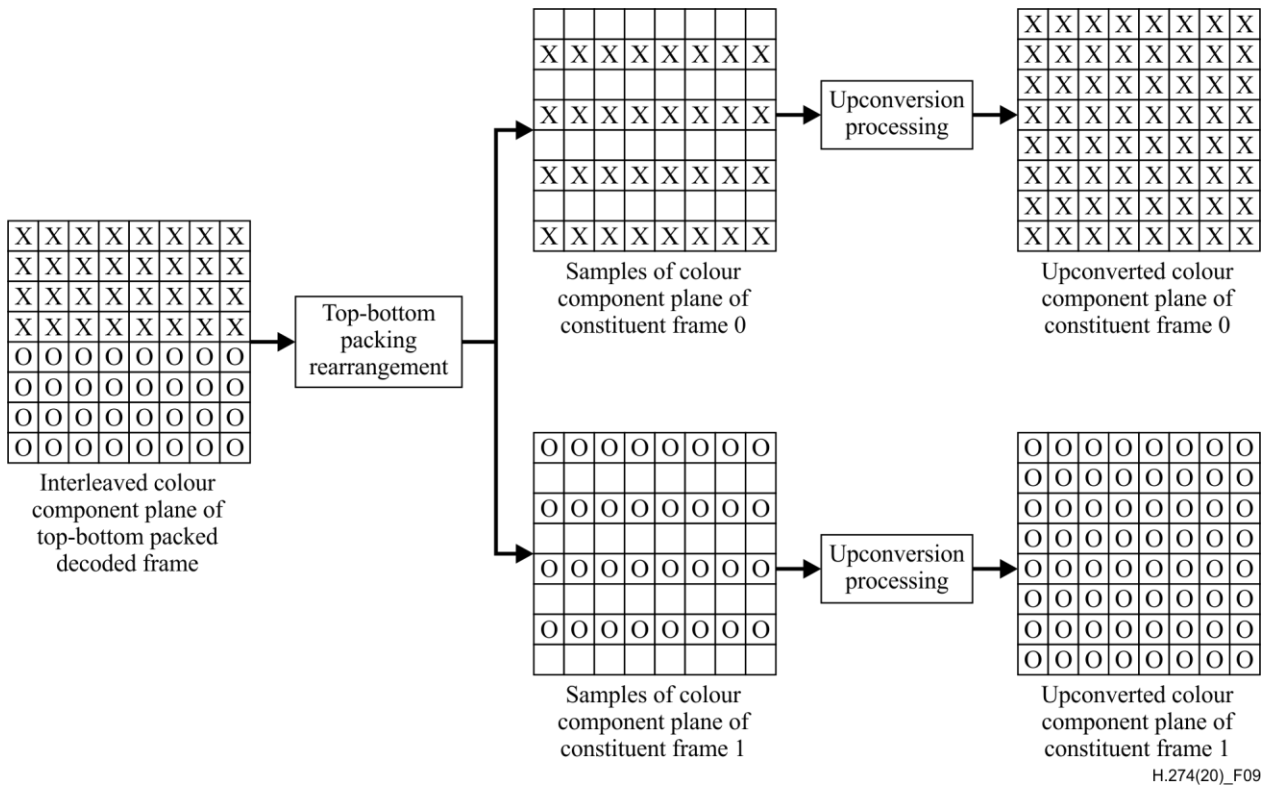
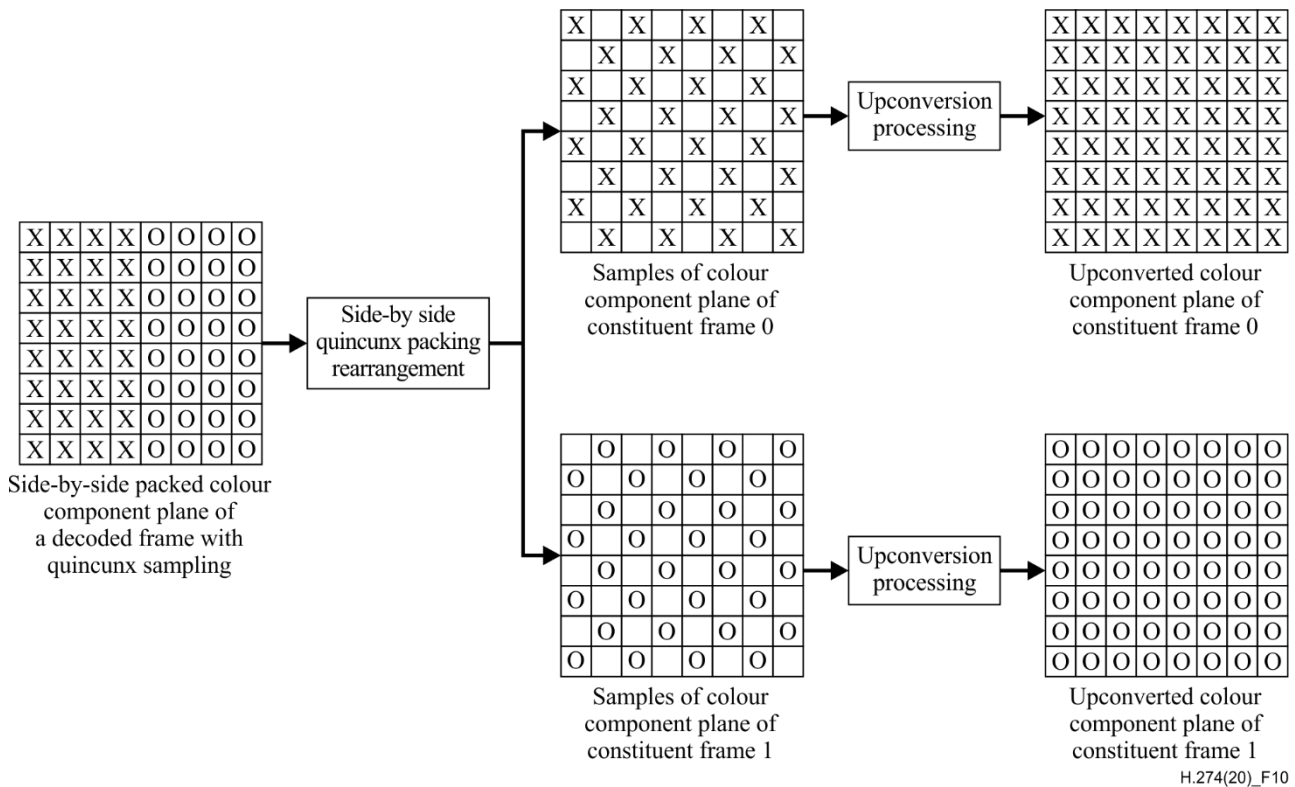


Figure 8 – Flowchart for rearrangement and upconversion of top-bottom packing arrangement with $fp_arrangement_type$ equal to 4, $fp_quincunx_sampling_flag$ equal to 0 and (x, y) equal to $(0, 0)$ or $(8, 4)$ for both constituent frames



H.274(20)_F09

Figure 9 – Flowchart for rearrangement and upconversion of top-bottom packing arrangement with $fp_arrangement_type$ equal to 4, $fp_quincunx_sampling_flag$ equal to 0, (x, y) equal to $(8, 12)$ for constituent frame 0 and (x, y) equal to $(0, 0)$ or $(8, 4)$ for constituent frame 1



H.274(20)_F10

Figure 10 – Flowchart for rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling ($fp_arrangement_type$ equal to 3 with $fp_quincunx_sampling_flag$ equal to 1)

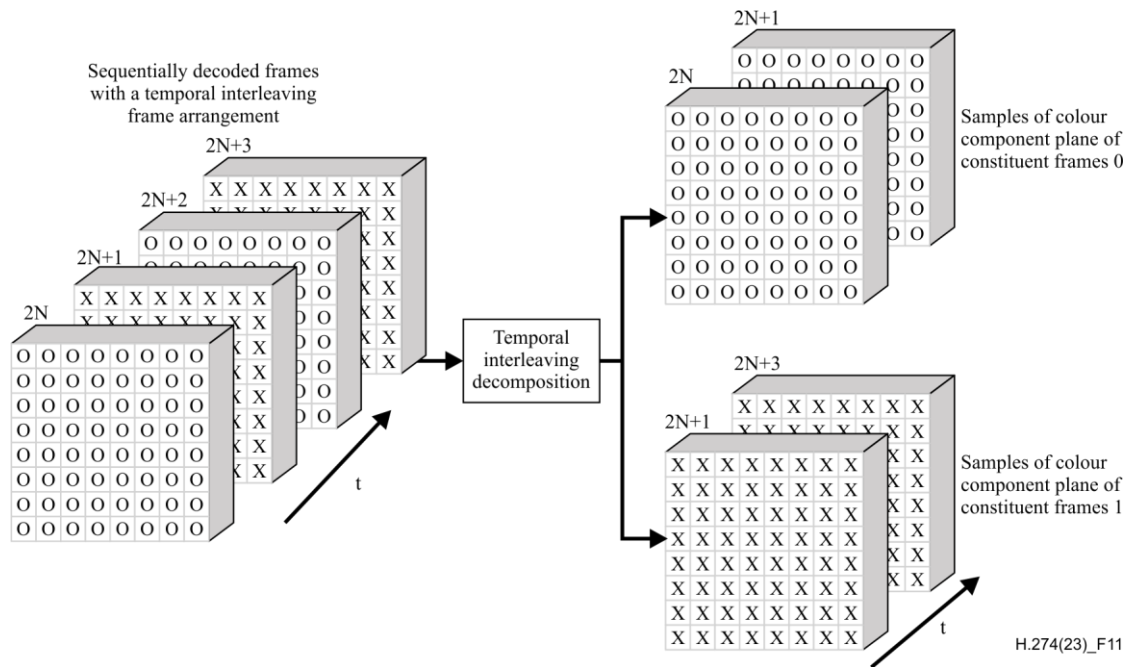


Figure 11 – Flowchart for rearrangement of a temporal interleaving frame arrangement (fp_arrangement_type equal to 5)

8.7 Parameter sets inclusion indication SEI message

8.7.1 Parameter sets inclusion indication SEI message syntax

parameter_sets_inclusion_indication(payloadSize) {	Descriptor
psii_self_contained_clvs_flag	u(1)
}	

8.7.2 Parameter sets inclusion indication SEI message semantics

This message provides an indication of whether the CLVS contains all the required NAL units for decoding the CLVS that is associated with the SEI message and whether temporal sublayer up-switching within the CLVS works without a need of fetching parameter sets from PUs earlier in decoding order than the PU containing the picture at which the temporal sublayer up-switching occurs. When the CLVS does not contain all the required NAL units, the NAL units that are not present in the CLVS may be provided externally.

psii_self_contained_clvs_flag equal to 1 indicates that the following restrictions apply:

- Each parameter set that is (directly or indirectly) referenced by any VCL NAL unit of the CLVS that is not a VCL NAL unit of a RASL picture (when present) associated with the first AU of the CLVS is present within the CLVS at a position that precedes, in decoding order, any NAL unit that (directly or indirectly) references the parameter set.
- For any STSA picture stsaPicA with temporal sublayer identifier equal to tIdA in the CLVS, the following applies:
 - stsaPicA does not refer to a PPS or an APS that precedes the first NAL unit of the PU containing stsaPicA in decoding order and has temporal sublayer identifier equal to tIdA.
 - For any picture picB with temporal sublayer identifier equal to tIdA and following stsaPicA in decoding order, picB does not refer to a PPS or an APS that has temporal sublayer identifier equal to tIdA that precedes the first NAL unit of the PU containing stsaPicA in decoding order.

psii_self_contained_clvs_flag equal to 0 indicates that this property might or might not apply.

8.8 Decoded picture hash SEI message

8.8.1 Decoded picture hash SEI message syntax

decoded_picture_hash(payloadSize) {	Descriptor
dph_sei_hash_type	u(8)
dph_sei_single_component_flag	u(1)
dph_sei_reserved_zero_7bits	u(7)
for(cIdx = 0; cIdx < (dph_sei_single_component_flag ? 1 : 3); cIdx++)	
if(dph_sei_hash_type == 0)	
for(i = 0; i < 16; i++)	
dph_sei_picture_md5[cIdx][i]	b(8)
else if(dph_sei_hash_type == 1)	
dph_sei_picture_crc[cIdx]	u(16)
else if(dph_sei_hash_type == 2)	
dph_sei_picture_checksum[cIdx]	u(32)
}	

8.8.2 Decoded picture hash SEI message semantics

This message provides a hash for each colour component of the current decoded picture.

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 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`.
- For each colour component `cIdx`, an array of samples `ComponentSample[cIdx][x][y]`.

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

Prior to computing the hash, the decoded picture data are arranged into one or three strings of bytes called `pictureData[cIdx]` of lengths `dataLen[cIdx]` as follows:

```

for( cIdx = 0; cIdx < dph_sei_single_component_flag ? 1 : 3; cIdx++ ) {
  if( cIdx == 0 ) {
    compWidth[ cIdx ] = PicWidthInLumaSamples
    compHeight[ cIdx ] = PicHeightInLumaSamples
    compDepth[ cIdx ] = BitDepthY
  } else {
    compWidth[ cIdx ] = PicWidthInLumaSamples / SubWidthC
    compHeight[ cIdx ] = PicHeightInLumaSamples / SubHeightC
    compDepth[ cIdx ] = BitDepthC
  }
  iLen = 0
  for( y = 0; y < compHeight[ cIdx ]; y++ ) /* raster scan order */
    for( x = 0; x < compWidth[ cIdx ]; x++ ) {
      pictureData[ cIdx ][ iLen++ ] = ComponentSample[ cIdx ][ x ][ y ] & 0xFF
      if( compDepth[ cIdx ] > 8 )
        pictureData[ cIdx ][ iLen++ ] = ComponentSample[ cIdx ][ x ][ y ] >> 8
    }
  dataLen[ cIdx ] = iLen
}

```

(32)

where `ComponentSample[cIdx]` is a two-dimensional array of the decoded sample values of a component of a decoded picture.

dph_sei_hash_type indicates the method used to calculate the checksum as specified in Table 9 . Values of **dph_sei_hash_type** that are not listed in in Table 9 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 decoded picture hash SEI messages that contain reserved values of **dph_sei_hash_type**.

Table 9 – Interpretation of dph_sei_hash_type

dph_sei_hash_type	Method
0	MD5 (IETF RFC 1321)
1	CRC
2	Checksum

dph_sei_single_component_flag equal to 1 specifies that the picture associated with the decoded picture hash SEI message contains a single colour component. **dph_sei_single_component_flag** equal to 0 specifies that the picture associated with the decoded picture hash SEI message contains three colour components. The value of **dph_sei_single_component_flag** shall be equal to (**ChromaFormatIdc** == 0).

dph_sei_reserved_zero_7bits shall be equal to 0. Values greater than 0 for **dph_sei_reserved_zero_7bits** 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 conforming to this version of this Specification shall ignore the value of **dph_sei_reserved_zero_7bits**.

dph_sei_picture_md5[cIdx][i] is the 16-byte MD5 hash of the **cIdx**-th colour component of the decoded picture. The value of **dph_sei_picture_md5[cIdx][i]** shall be equal to the value of **digestVal[cIdx]** obtained as follows, using the MD5 functions defined in IETF RFC 1321:

```
MD5Init( context )
MD5Update( context, pictureData[ cIdx ], dataLen[ cIdx ] )
MD5Final( digestVal[ cIdx ], context )
```

(33)

dph_sei_picture_crc[cIdx] is the cyclic redundancy check (CRC) of the colour component **cIdx** of the decoded picture. The value of **dph_sei_picture_crc[cIdx]** shall be equal to the value of **crcVal[cIdx]** obtained as follows:

```
crc = 0xFFFF
pictureData[ cIdx ][ dataLen[ cIdx ] ] = 0
pictureData[ cIdx ][ dataLen[ cIdx ] + 1 ] = 0
for( bitIdx = 0; bitIdx < ( dataLen[ cIdx ] + 2 ) * 8; bitIdx++ ) {
    dataByte = pictureData[ cIdx ][ bitIdx >> 3 ]
    crcMsb = ( crc >> 15 ) & 1
    bitVal = ( dataByte >> ( 7 - ( bitIdx & 7 ) ) ) & 1
    crc = ( ( ( crc << 1 ) + bitVal ) & 0xFFFF ) ^ ( crcMsb * 0x1021 )
}
crcVal[ cIdx ] = crc
```

(34)

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

dph_sei_picture_checksum[cIdx] is the checksum of the colour component **cIdx** of the decoded picture. The value of **dph_sei_picture_checksum[cIdx]** shall be equal to the value of **checksumVal[cIdx]** obtained as follows:

```
sum = 0
for( y = 0; y < compHeight[ cIdx ]; y++ )
    for( x = 0; x < compWidth[ cIdx ]; x++ ) {
        xorMask = ( x & 0xFF ) ^ ( y & 0xFF ) ^ ( x >> 8 ) ^ ( y >> 8 )
        sum = ( sum + ( ( ComponentSample[ cIdx ][ y * compWidth[ cIdx ] + x ] & 0xFF ) ^
            xorMask ) ) & 0xFFFFFFFF
        if( compDepth[ cIdx ] > 8 )
            sum = ( sum + ( ( ComponentSample[ cIdx ][ y * compWidth[ cIdx ] + x ] >> 8 ) ^
                xorMask ) ) & 0xFFFFFFFF
    }
checksumVal[ cIdx ] = sum
```

(35)

8.9 Mastering display colour volume SEI message

8.9.1 Mastering display colour volume SEI message syntax

	Descriptor
mastering_display_colour_volume(payloadSize) {	
for(c = 0; c < 3; c++) {	
mdcv_display primaries_x [c]	u(16)
mdcv_display primaries_y [c]	u(16)
}	
mdcv_white_point_x	u(16)
mdcv_white_point_y	u(16)
mdcv_max_display mastering luminance	u(32)
mdcv_min_display mastering luminance	u(32)
}	

8.9.2 Mastering display colour volume SEI message semantics

This SEI message identifies the colour volume (the colour primaries, white point, and luminance range) of a display considered to be the mastering display for the associated video content – e.g., the colour volume of a display that was used for viewing while authoring the video content. The described mastering display is a three-colour additive display system that has been configured to use the indicated mastering colour volume.

This SEI message does not identify the measurement methodologies and procedures used for determining the indicated values or provide any description of the mastering environment. It also does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of SMPTE ST 2086.

When a mastering display colour volume SEI message is present for any picture of a CLVS of a particular layer, a mastering display colour volume SEI message shall be present for the first picture of the CLVS. The mastering display colour volume SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All mastering display colour volume SEI messages that apply to the same CLVS shall have the same content.

mdcv_display primaries_x[c], when in the range of 5 to 37 000, inclusive, specifies the normalized x chromaticity coordinate of the colour primary component c of the mastering display, according to the CIE 1931 definition of x as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in increments of 0.00002. When **mdcv_display primaries_x**[c] is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

mdcv_display primaries_y[c], when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the colour primary component c of the mastering display, according to the CIE 1931 definition of y as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in increments of 0.00002. When **mdcv_display primaries_y**[c] is not in the range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

For describing mastering displays that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary specified in the VUI parameters.

mdcv_white_point_x, when in the range of 5 to 37 000, inclusive, specifies the normalized x chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of x as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in normalized increments of 0.00002. When **mdcv_white_point_x** is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

mdcv_white_point_y, when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of y as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in normalized increments of 0.00002. When **mdcv_white_point_y** is not in the

range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 1 – SMPTE ST 2086 specifies that the normalized x and y chromaticity coordinate values for the mastering display colour primaries and white point are to be represented with four decimal places. This would correspond with using values of the syntax elements `mdcv_display_primaries_x[c]`, `mdcv_display_primaries_y[c]`, `mdcv_white_point_x`, and `mdcv_white_point_y`, as defined in this Specification, that are multiples of 5.

NOTE 2 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G uses normalized (x, y) chromaticity coordinate values of (0,0) for the white point to indicate that the white point chromaticity is unknown.

mdcv_max_display_mastering_luminance, when in the range of 50 000 to 100 000 000, specifies the nominal maximum display luminance of the mastering display in units of 0.0001 candelas per square metre. When `mdcv_max_display_mastering_luminance` is not in the range of 50 000 to 100 000 000, the nominal maximum display luminance of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 3 – SMPTE ST 2086 specifies that the nominal maximum display luminance of the mastering display is to be specified as a multiple of 1 candela per square metre. This would correspond with using values of the syntax element `mdcv_max_display_mastering_luminance`, as defined in this Specification, that are a multiple of 10 000.

NOTE 4 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G uses the value 0 for the nominal maximum display luminance of the mastering display to indicate that the nominal maximum display luminance of the mastering display is unknown.

mdcv_min_display_mastering_luminance, when in the range of 1 to 50 000, specifies the nominal minimum display luminance of the mastering display in units of 0.0001 candelas per square metre. When `mdcv_min_display_mastering_luminance` is not in the range of 1 to 50 000, the nominal maximum display luminance of the mastering display is unknown or unspecified or specified by other means not specified in this Specification. When `mdcv_max_display_mastering_luminance` is equal to 50 000, `mdcv_min_display_mastering_luminance` shall not be equal to 50 000.

NOTE 5 – SMPTE ST 2086 specifies that the nominal minimum display luminance of the mastering display is to be specified as a multiple of 0.0001 candelas per square metre, which corresponds to the semantics specified in this Specification.

NOTE 6 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G uses the value 0 for the nominal minimum display luminance of the mastering display to indicate that the nominal minimum display luminance of the mastering display is unknown.

NOTE 7 – Another example of the potential use of values outside the range for which semantics are specified in this Specification is that SMPTE ST 2086 indicates that values outside the specified range could be used to indicate that the black level and contrast of the mastering display have been adjusted using picture line-up generation equipment (PLUGE).

At the minimum luminance, the mastering display is considered to have the same nominal chromaticity as the white point.

8.10 Content light level information SEI message

8.10.1 Content light level information SEI message syntax

<code>content_light_level_info(payloadSize) {</code>	Descriptor
<code> clli_max_content_light_level</code>	<code>u(16)</code>
<code> clli_max_pic_average_light_level</code>	<code>u(16)</code>
<code>}</code>	

8.10.2 Content light level information SEI message semantics

This SEI message identifies upper bounds for the nominal target brightness light level of the pictures of the CLVS.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of the CEA 861.3 specification.

The semantics of the content light level information SEI message are defined in relation to the values of samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the pictures of the CLVS, in units of candelas per square metre. However, this SEI message does not, by itself, identify a conversion process for converting the sample values of a decoded picture to the samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the picture.

NOTE 1 – Other syntax elements, such as `vui_colour_primaries`, `vui_transfer_characteristics`, and `vui_matrix_coeffs`, when present, could assist in the identification of such a conversion process.

Given the red, green, and blue colour primary intensities in the linear light domain for the location of a luma sample in a corresponding 4:4:4 representation, denoted as E_R , E_G , and E_B , the maximum component intensity is defined as $E_{Max} = \text{Max}(E_R, \text{Max}(E_G, E_B))$. The light level corresponding to the stimulus is then defined as the CIE 1931 luminance corresponding to equal amplitudes of E_{Max} for all three colour primary intensities for red, green, and blue (with appropriate scaling to reflect the nominal luminance level associated with peak white – e.g., ordinarily scaling to associate peak white with 10 000 candelas per square metre when `vui_transfer_characteristics` is equal to 16).

NOTE 2 – Since the maximum value E_{Max} is used in this definition at each sample location, rather than a direct conversion from E_R , E_G , and E_B to the corresponding CIE 1931 luminance, the CIE 1931 luminance at a location could in some cases be less than the indicated light level. This situation would occur, for example, when E_R and E_G are very small and E_B is large, in which case the indicated light level would be much larger than the true CIE 1931 luminance associated with the (E_R, E_G, E_B) triplet.

All content light level information SEI messages that apply to the same CLVS shall have the same content.

cli_max_content_light_level, when not equal to 0, indicates an upper bound on the maximum light level among all individual samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for the pictures of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by `cli_max_content_light_level`.

cli_max_pic_average_light_level, when not equal to 0, indicates an upper bound on the maximum average light level among the samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for any individual picture of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by `cli_max_pic_average_light_level`.

When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated average should be performed only within the visually relevant region.

8.11 Dependent random access point indication SEI message

8.11.1 Dependent random access point indication SEI message syntax

<code>dependent_rap_indication(payloadSize) {</code>	Descriptor
<code>}</code>	

8.11.2 Dependent random access point indication SEI message semantics

The picture associated with a dependent random access point (DRAP) indication SEI message is referred to as a DRAP picture.

The presence of the DRAP indication SEI message indicates that the constraints on picture order and picture referencing specified in this clause apply. These constraints can enable a decoder to properly decode the DRAP picture and the pictures that are in the same layer and follow it in both decoding order and output order without needing to decode any other pictures in the same layer except the associated IRAP picture of the DRAP picture.

The constraints indicated by the presence of the DRAP indication SEI message, which shall all apply, are as follows:

- The DRAP picture is a trailing picture.
- The DRAP picture has a temporal sublayer identifier equal to 0.
- The DRAP picture does not include any pictures in the same layer in the active entries of its reference picture lists except the associated IRAP picture of the DRAP picture.
- Any picture that is in the same layer and follows the DRAP picture in both decoding order and output order does not include, in the active entries of its reference picture lists, any picture that is in the same layer and precedes the DRAP picture in decoding order or output order, with the exception of the associated IRAP picture of the DRAP picture.

8.12 Alternative transfer characteristics information SEI message

8.12.1 Alternative transfer characteristics information SEI message syntax

	Descriptor
alternative_transfer_characteristics (payloadSize) {	
preferred_transfer_characteristics	u(8)
}	

8.12.2 Alternative transfer characteristics SEI message semantics

The alternative transfer characteristics SEI message provides a preferred alternative value for the transfer_characteristics syntax element that is indicated by the colour description syntax of the VUI parameters. This SEI message is intended to be used in cases when some value of vui_transfer_characteristics is preferred for interpretation of the pictures of the CLVS although some other value of vui_transfer_characteristics could also be acceptable for interpretation of the pictures of the CLVS and that other value is provided in the colour description syntax of the VUI parameters for interpretation by decoders that do not support interpretation of the preferred value (e.g., because the preferred value had not yet been defined in a previous version of this Specification).

When an alternative transfer characteristics SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an alternative transfer characteristics SEI message shall be present for that IRAP picture. The alternative transfer characteristics SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All alternative transfer characteristics SEI messages that apply to the same CLVS shall have the same content.

preferred_transfer_characteristics specifies a preferred alternative value for the vui_transfer_characteristics syntax element of the colour description syntax of the VUI parameters. The semantics for preferred_transfer_characteristics are otherwise the same as for the vui_transfer_characteristics syntax element specified in the VUI parameters. When preferred_transfer_characteristics is not equal to the value of vui_transfer_characteristics indicated in the VUI parameters, decoders should ignore the value of vui_transfer_characteristics indicated in the VUI parameters and instead use the value indicated by preferred_transfer_characteristics.

8.13 Ambient viewing environment SEI message

8.13.1 Ambient viewing environment SEI message syntax

	Descriptor
ambient_viewing_environment(payloadSize) {	
ambient_illuminance	u(32)
ambient_light_x	u(16)
ambient_light_y	u(16)
}	

8.13.2 Ambient viewing environment SEI message semantics

The ambient viewing environment SEI message identifies the characteristics of the nominal ambient viewing environment for the display of the associated video content. The syntax elements of the ambient viewing environment SEI message can assist the receiving system in adapting the received video content for local display in viewing environments that could be similar or could substantially differ from those assumed or intended when mastering the video content.

This SEI message does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

When an ambient viewing environment SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an ambient viewing environment SEI message shall be present for that IRAP picture. The ambient viewing environment SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All ambient viewing environment SEI messages that apply to the same CLVS shall have the same content.

ambient_illuminance specifies the environmental illuminance of the ambient viewing environment in units of 0.0001 lux. ambient_illuminance shall not be equal to 0.

ambient_light_x and **ambient_light_y** specify the normalized x and y chromaticity coordinates, respectively, of the environmental ambient light in the nominal viewing environment, according to the CIE 1931 definition of x and y as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in normalized increments of 0.00002. The values of **ambient_light_x** and **ambient_light_y** shall be in the range of 0 to 50 000.

NOTE – For example, the conditions identified in Rec. ITU-R BT.2035 could be expressed using **ambient_illuminance** equal to 100 000 with background chromaticity indicating D₆₅ (**ambient_light_x** equal to 15 635, **ambient_light_y** equal to 16 450), or optionally in some regions, background chromaticity indicating D₉₃ (**ambient_light_x** equal to 14 155, **ambient_light_y** equal to 14 855).

8.14 Content colour volume SEI message

8.14.1 Content colour volume SEI message syntax

	Descriptor
content_colour_volume(payloadSize) {	
ccv_cancel_flag	u(1)
if(!ccv_cancel_flag) {	
ccv_persistence_flag	u(1)
ccv primaries_present_flag	u(1)
ccv_min_luminance_value_present_flag	u(1)
ccv_max_luminance_value_present_flag	u(1)
ccv_avg_luminance_value_present_flag	u(1)
ccv_reserved_zero_2bits	u(2)
if(ccv primaries_present_flag)	
for(c = 0; c < 3; c++) {	
ccv primaries_x[c]	i(32)
ccv primaries_y[c]	i(32)
}	
if(ccv_min_luminance_value_present_flag)	
ccv_min_luminance_value	u(32)
if(ccv_max_luminance_value_present_flag)	
ccv_max_luminance_value	u(32)
if(ccv_avg_luminance_value_present_flag)	
ccv_avg_luminance_value	u(32)
}	
}	

8.14.2 Content colour volume SEI message semantics

The content colour volume SEI message describes the colour volume characteristics of the associated pictures. These colour volume characteristics are expressed in terms of a nominal range, although deviations from this range may occur.

The variable **transferCharacteristics** is specified as follows:

- If an alternative transfer characteristics SEI message is present for the CLVS, **transferCharacteristics** is set equal to **preferred_transfer_characteristics**;
- Otherwise, (an alternative transfer characteristics SEI message is not present for the CLVS), **transferCharacteristics** is set equal to **vui_transfer_characteristics**.

The content colour volume SEI message shall not be present, and decoders shall ignore it, when any of the following conditions is true:

- Any of the values of **transferCharacteristics**, **vui_colour primaries**, and **vui_matrix_coeffs** has a value defined as unknown or unspecified.
- The value of **vui_transfer_characteristics** is equal to 2, 4, or 5.

- The value of `vui_colour primaries` is equal to 2.

The following applies when converting the signal from a non-linear to a linear representation:

- If the value of `transferCharacteristics` is equal to 1, 6, 7, 14, or 15, the Rec. ITU-R BT.1886 reference electro-optical transfer function should be used to convert the signal to its linear representation, where the value of screen luminance for white is set equal to 100 candelas per square metre, the value of screen luminance for black is set equal to 0 candelas per square metre, and the value of the exponent of the power function is set equal to 2.4.
- Otherwise, if the value of `transferCharacteristics` is equal to 18, the hybrid log-gamma reference electro-optical transfer function specified in Rec. ITU-R BT.2100 should be used to convert the signal to its linear representation, where the value of nominal peak luminance of the display is set equal to 1000 candelas per square metre, the value of the display luminance for black is set equal to 0 candelas per square metre, and the value of system gamma is set equal to 1.2.
- Otherwise (the value of `transferCharacteristics` is not equal to 1, 6, 7, 14, 15, or 18) when the content colour volume SEI message is present, the exact inverse of the transfer function specified in specified in the VUI parameters should be used to convert the non-linear signal to a linear representation.

`ccv_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous content colour volume SEI message in output order that applies to the current layer. `ccv_cancel_flag` equal to 0 indicates that content colour volume information follows.

`ccv_persistence_flag` specifies the persistence of the content colour volume SEI message for the current layer.

`ccv_persistence_flag` equal to 0 specifies that the content colour volume applies to the current decoded picture only.

`ccv_persistence_flag` equal to 1 specifies that the content colour volume 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 content colour volume SEI message is output that follows the current picture in output order.

`ccv_primaries_present_flag` equal to 1 specifies that the syntax elements `ccv_primaries_x[c]` and `ccv_primaries_y[c]` are present. `ccv_primaries_present_flag` equal to 0 specifies that the syntax elements `ccv_primaries_x[c]` and `ccv_primaries_y[c]` are not present.

`ccv_min_luminance_value_present_flag` equal to 1 specifies that the syntax element `ccv_min_luminance_value` is present. `ccv_min_luminance_value_present_flag` equal to 0 specifies that the syntax element `ccv_min_luminance_value` is not present.

`ccv_max_luminance_value_present_flag` equal to 1 specifies that the syntax element `ccv_max_luminance_value` is present. `ccv_max_luminance_value_present_flag` equal to 0 specifies that the syntax element `ccv_max_luminance_value` is not present.

`ccv_avg_luminance_value_present_flag` equal to 1 specifies that the syntax element `ccv_avg_luminance_value` is present. `ccv_avg_luminance_value_present_flag` equal to 0 specifies that the syntax element `ccv_avg_luminance_value` is not present.

It is a requirement of bitstream conformance that the values of `ccv_primaries_present_flag`, `ccv_min_luminance_value_present_flag`, `ccv_max_luminance_value_present_flag`, and `ccv_avg_luminance_value_present_flag` shall not all be equal to 0.

`ccv_reserved_zero_2bits[i]` shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `reserved_zero_2bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `reserved_zero_2bits[i]`.

`ccv_primaries_x[c]` and **`ccv_primaries_y[c]`** specify the normalized x and y chromaticity coordinates, respectively, of the colour primary component c of the nominal content colour volume, according to the CIE 1931 definition of x and y as specified in ISO/CIE 11664-1 (see also ISO/CIE 11664-3 and CIE 15), in normalized increments of 0.00002. For describing colour volumes that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary specified in the VUI parameters.

The values of `ccv_primaries_x[c]` and `ccv_primaries_y[c]` shall be in the range of $-5\,000\,000$ to $5\,000\,000$, inclusive.

When `ccv_primaries_x[c]` and `ccv_primaries_y[c]` are not present, they are inferred to be equal to the normalized x and y chromaticity coordinates, respectively, specified by `vui_colour primaries`.

ccv_min_luminance_value specifies the normalized minimum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in the VUI parameters according to the indicated transfer characteristics of the signal. The values of `ccv_min_luminance_value` are in normalized increments of 0.0000001.

ccv_max_luminance_value specifies the maximum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in the VUI parameters according to the transfer characteristics of the signal. The values of `ccv_max_luminance_value` are in normalized increments of 0.0000001.

ccv_avg_luminance_value specifies the average luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in the VUI parameters according to the transfer characteristics of the signal. The values of `ccv_avg_luminance_value` are in normalized increments of 0.0000001.

NOTE – The resulting domain from this conversion process might or might not represent light in a source or display domain – it is merely a gamut representation domain rather than necessarily being a representation of actual light in either the scene or display domain. Therefore, the values corresponding to `ccv_min_luminance_value`, `ccv_max_luminance_value`, and `ccv_avg_luminance_value` might not necessarily correspond to a true luminance value.

The value of `ccv_min_luminance_value`, when present, shall be less than or equal to `ccv_avg_luminance_value`, when present. The value of `ccv_avg_luminance_value`, when present, shall be less than or equal to `ccv_max_luminance_value`, when present. The value of `ccv_min_luminance_value`, when present, shall be less than or equal to `ccv_max_luminance_value`, when present.

When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated `ccv_min_luminance_value`, `ccv_max_luminance_value`, and `ccv_avg_luminance_value` should correspond only to values within the visually relevant region.

8.15 Omnidirectional video specific SEI messages

8.15.1 Sample location remapping process

8.15.1.1 General

Use of this process requires the definition of the following variable:

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

To remap colour sample locations of a region-wise packed picture to a unit sphere, the following ordered steps are applied:

1. A region-wise packed picture is obtained as the cropped decoded picture by decoding a coded picture. For purposes of interpretation of chroma samples, the input to the indicated remapping process is the set of decoded sample values after applying an (unspecified) upsampling conversion process to the 4:4:4 colour sampling format as necessary when `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format). This (unspecified) upsampling process should account for the relative positioning relationship between the luma and chroma samples as indicated by `vui_chroma_sample_loc_type_frame`, `vui_chroma_sample_loc_type_top_field`, and `vui_chroma_sample_loc_type_bottom_field` syntax elements in the VUI parameters, when present.
2. If RWP is indicated, the sample locations of the region-wise packed picture are converted to sample locations of the respective projected picture as specified in clause 8.15.1.4. Otherwise, the projected picture is identical to the region-wise packed picture.
3. If frame packing is indicated, the sample locations of the projected picture are converted to sample locations of the respective constituent picture of the projected picture, as specified in clause 8.15.1.5. Otherwise, the constituent picture of the projected picture is identical to the projected picture.
4. The sample locations of a constituent picture of the projected picture are converted to sphere coordinates relative to the local coordinate axes, as specified in clause 8.15.1.2.
5. If rotation is indicated, the sphere coordinates relative to the local coordinate axes are converted to sphere coordinates relative to the global coordinate axes, as specified in clause 8.15.1.3. Otherwise, the global coordinate axes are identical to the local coordinate axes.

The overall process for mapping of luma sample locations within a region-wise packed picture to sphere coordinates relative to the global coordinate axes is normatively specified in clause 8.15.1.5.

For each region-wise packed picture corresponding to a decoded picture, the following applies:

- When an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 that applies to the picture is present, `ErpFlag` is set equal to 1, and `CmpFlag` is set equal to 0.
- When a generalized cubemap projection SEI message with `gcmap_cancel_flag` equal to 0 that applies to the picture is present, `CmpFlag` is set equal to 1, and `ErpFlag` is set equal to 0.
- If a sphere rotation SEI message with `sphere_rotation_cancel_flag` equal to 0 that applies to the picture is present, `RotationFlag` is set equal to 1, and `RotationYaw`, `RotationPitch`, and `RotationRoll` are set equal to $\text{yaw_rotation} \div 2^{16}$, $\text{pitch_rotation} \div 2^{16}$, and $\text{roll_rotation} \div 2^{16}$, respectively.
- Otherwise, `RotationFlag` is set equal to 0.
- If a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0 that applies to the picture is not present, `StereoFlag`, `TopBottomFlag`, and `SideBySideFlag` are all set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 1.
- Otherwise, the following applies:
 - `StereoFlag` is set equal to 1.
 - If the value of `fp_arrangement_type` of the frame packing arrangement SEI message is equal to 3, `TopBottomFlag` is set equal to 0, `SideBySideFlag` is set equal to 1, `HorDiv1` is set equal to 2 and `VerDiv1` is set equal to 1.
 - Otherwise, if the value of `fp_arrangement_type` of the frame packing arrangement SEI message is equal to 4, `TopBottomFlag` is set equal to 1, `SideBySideFlag` is set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 2.
 - Otherwise, `TopBottomFlag` is set equal to 0, `SideBySideFlag` is set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 1.
- If a RWP SEI message with `rwp_cancel_flag` equal to 0 that applies to the picture is not present, `RegionWisePackingFlag` is set equal to 0, and `ConstituentPicWidth` and `ConstituentPicHeight` are set to be equal to $\text{cropPicWidth} / \text{HorDiv1}$ and $\text{cropPicHeight} / \text{VerDiv1}$, respectively, where `cropPicWidth` and `cropPicHeight` are the width and height, respectively, of the cropped decoded picture.
- Otherwise, `RegionWisePackingFlag` is set equal to 1, and `ConstituentPicWidth` and `ConstituentPicHeight` are set equal to $\text{rwp_proj_picture_width} / \text{HorDiv1}$ and $\text{rwp_proj_picture_height} / \text{VerDiv1}$, respectively.

8.15.1.2 Projection for one sample location

Inputs to this process are:

- `pictureWidth` and `pictureHeight`, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units (see clause 8.15.5.2), and
- the centre point of a sample location (`hPos`, `vPos`) along the horizontal and vertical axes, respectively, in relative projected picture sample units, where `hPos` and `vPos` could have non-integer real values.

Outputs of this process are:

- sphere coordinates (ϕ , θ) for the sample location in degrees relative to the local coordinate axes.

The projection for a sample location is derived as follows:

- If `ErpFlag` is equal to 1, the following applies:
 - If `RegionWisePackingFlag` is equal to 0 and `erp_guard_band_flag` is equal to 1, the following applies:

$$\begin{aligned} \text{hPos}' &= \text{hPos} - \text{erp_left_guard_band_width} \\ \text{tmpPicWidth} &= \text{pictureWidth} - \text{erp_left_guard_band_width} - \text{erp_right_guard_band_width} \end{aligned} \quad (36)$$

- Otherwise, the following applies:

$$\begin{aligned} \text{hPos}' &= \text{hPos} \\ \text{tmpPicWidth} &= \text{pictureWidth} \end{aligned} \quad (37)$$

- The following applies:

$$\begin{aligned} \phi &= 180 - \text{hPos}' * (360 \div \text{tmpPicWidth}) \\ \theta &= 90 - \text{vPos} * (180 \div \text{pictureHeight}) \end{aligned} \quad (38)$$

- Otherwise (CmpFlag is equal to 1), the outputs are derived by the following ordered steps:
 1. Clause 8.15.1.7 is invoked with pictureWidth and pictureHeight as inputs, and the output is assigned to faceWidth and faceHeight.
 2. Clause 8.15.1.8 is invoked with hPos, vPos, faceWidth, and faceHeight, where hPos and vPos are within a projected picture, and the output is assigned to hPosFace and vPosFace within a projected face.
 3. Clause 8.15.1.9 is invoked with hPosFace, vPosFace, faceWidth, and faceHeight, and the output is assigned to hPosRot and vPosRot.
 4. If gcmp_packing_type is equal to 4 or 5, clause 8.15.1.10 is invoked with hPosRot, vPosRot, faceWidth, and faceHeight, and the output is assigned to hPosAdj and vPosAdj. Otherwise, hPosAdj and vPosAdj are identical to hPosRot and vPosRot, respectively.

- 5. The following applies:

$$\begin{aligned} \text{hPos}' &= - (2 * \text{hPosAdj} \div \text{faceWidth}) + 1 \\ \text{vPos}' &= - (2 * \text{vPosAdj} \div \text{faceHeight}) + 1 \end{aligned} \quad (39)$$

- If gcmp_mapping_function_type is equal to 0, the following applies:

$$\begin{aligned} \text{hPos}'' &= \text{hPos}' \\ \text{vPos}'' &= \text{vPos}' \end{aligned} \quad (40)$$

- Otherwise, if gcmp_mapping_function_type is equal to 1, the following applies:

$$\begin{aligned} \text{hPos}'' &= \text{Tan}(\text{hPos}' * \pi \div 4) \\ \text{vPos}'' &= \text{Tan}(\text{vPos}' * \pi \div 4) \end{aligned} \quad (41)$$

- Otherwise (gcmp_mapping_function_type is equal to 2), the following applies:

$$\begin{aligned} \text{coeffU}[n] &= (\text{gcmp_function_coeff_u}[n] + 1) \div 128 \\ \text{coeffV}[n] &= (\text{gcmp_function_coeff_v}[n] + 1) \div 128 \\ \text{hPos}'' &= \text{hPos}' \div (1 + \text{coeffU}[n] * (1 - \text{gcmp_function_u_affected_by_v_flag}[n] * \text{vPos}'^2) * \\ &\quad (1 - \text{hPos}'^2)) \\ \text{vPos}'' &= \text{vPos}' \div (1 + \text{coeffV}[n] * (1 - \text{gcmp_function_v_affected_by_u_flag}[n] * \text{hPos}'^2) * \\ &\quad (1 - \text{vPos}'^2)) \end{aligned} \quad (42)$$

- The following applies:

```

if( gcmp_face_index[ n ] == 0 ) { /* positive x front face */
  x = 1.0
  y = hPos''
  z = vPos''
} else if( gcmp_face_index[ n ] == 1 ) { /* negative x back face */
  x = -1.0
  y = -vPos''
  z = -hPos''
} else if( gcmp_face_index[ n ] == 2 ) { /* positive z top face */
  x = -hPos''
  y = -vPos''
  z = 1.0
} else if( gcmp_face_index[ n ] == 3 ) { /* negative z bottom face */
  x = hPos''
  y = -vPos''
  z = -1.0
} else if( gcmp_face_index[ n ] == 5 ) { /* positive y left face */
  x = -hPos''
  y = 1.0
  z = vPos''
} else { /* ( gcmp_face_index[ n ] == 4 ), negative y right face */
  x = hPos''
  y = -1.0
  z = vPos''
}

```

$$\begin{aligned} \phi &= \text{Atan2}(y, x) * 180 \div \pi \\ \theta &= \text{Asin}(z \div \text{Sqrt}(x^2 + y^2 + z^2)) * 180 \div \pi \end{aligned} \quad (43)$$

8.15.1.3 Conversion from the local coordinate axes to the global coordinate axes

Inputs to this process are:

- rotation_yaw (α_d), rotation_pitch (β_d), rotation_roll (γ_d), all in units of degrees, and
- sphere coordinates (ϕ_d, θ_d) relative to the local coordinate axes.

Outputs of this process are:

- sphere coordinates (ϕ', θ') relative to the global coordinate axes.

The outputs are derived as follows:

$$\begin{aligned}\phi &= \phi_d * \pi \div 180 \\ \theta &= \theta_d * \pi \div 180 \\ \alpha &= \alpha_d * \pi \div 180 \\ \beta &= \beta_d * \pi \div 180 \\ \gamma &= \gamma_d * \pi \div 180 \\ x_1 &= \text{Cos}(\phi) * \text{Cos}(\theta) \\ y_1 &= \text{Sin}(\phi) * \text{Cos}(\theta) \\ z_1 &= \text{Sin}(\theta) \\ x_2 &= \text{Cos}(\beta) * \text{Cos}(\alpha) * x_1 - \text{Cos}(\beta) * \text{Sin}(\alpha) * y_1 + \text{Sin}(\beta) * z_1 \\ y_2 &= (\text{Cos}(\gamma) * \text{Sin}(\alpha) + \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\ &\quad (\text{Cos}(\gamma) * \text{Cos}(\alpha) - \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 - \\ &\quad \text{Sin}(\gamma) * \text{Cos}(\beta) * z_1 \\ z_2 &= (\text{Sin}(\gamma) * \text{Sin}(\alpha) - \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\ &\quad (\text{Sin}(\gamma) * \text{Cos}(\alpha) + \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 + \\ &\quad \text{Cos}(\gamma) * \text{Cos}(\beta) * z_1 \\ \phi' &= \text{Atan2}(y_2, x_2) * 180 \div \pi \\ \theta' &= \text{Asin}(z_2) * 180 \div \pi\end{aligned}\tag{44}$$

8.15.1.4 Conversion of sample locations for rectangular region-wise packing

Inputs to this process are:

- sample location (x, y) within the packed region, where x and y are in relative region-wise packed picture sample units (see clause 8.15.5.2), while the sample location is at an integer sample location within the packed picture,
- the width and the height ($\text{projRegWidth}, \text{projRegHeight}$) of the projected region, in relative projected picture sample units,
- the width and the height ($\text{packedRegWidth}, \text{packedRegHeight}$) of the packed region, in relative region-wise packed picture sample units, and
- transform type (transformType).

Outputs of this process are:

- the centre point of the sample location (hPos, vPos) within the projected region in relative projected picture sample units, where hPos and vPos could have non-integer real values.

The outputs are derived as follows:

```
if( transformType == 0 || transformType == 1 || transformType == 2 || transformType == 3 ) {
    horRatio = projRegWidth ÷ packedRegWidth
    verRatio = projRegHeight ÷ packedRegHeight
} else if ( transformType == 4 || transformType == 5 || transformType == 6 ||
    transformType == 7 ) {
    horRatio = projRegWidth ÷ packedRegHeight
    verRatio = projRegHeight ÷ packedRegWidth
}
if( transformType == 0 ) {
    hPos = horRatio * ( x + 0.5 )
    vPos = verRatio * ( y + 0.5 )
} else if ( transformType == 1 ) {
    hPos = horRatio * ( packedRegWidth - x - 0.5 )
    vPos = verRatio * ( y + 0.5 )
} else if ( transformType == 2 ) {
```

```

    hPos = horRatio * ( packedRegWidth - x - 0.5 )
    vPos = verRatio * ( packedRegHeight - y - 0.5 )
} else if ( transformType == 3 ) {
    hPos = horRatio * ( x + 0.5 )
    vPos = verRatio * ( packedRegHeight - y - 0.5 )
} else if ( transformType == 4 ) {
    hPos = horRatio * ( y + 0.5 )
    vPos = verRatio * ( x + 0.5 )
} else if ( transformType == 5 ) {
    hPos = horRatio * ( y + 0.5 )
    vPos = verRatio * ( packedRegWidth - x - 0.5 )
} else if ( transformType == 6 ) {
    hPos = horRatio * ( packedRegHeight - y - 0.5 )
    vPos = verRatio * ( packedRegWidth - x - 0.5 )
} else if ( transformType == 7 ) {
    hPos = horRatio * ( packedRegHeight - y - 0.5 )
    vPos = verRatio * ( x + 0.5 )
}

```

NOTE – The offsets equal to 0.5 result in a sampling position that is in the centre point of a sample in packed picture sample units.

8.15.1.5 Mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes

This clause specifies the mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes.

If RegionWisePackingFlag is equal to 1, the following applies for each packed region n in the range of 0 to NumPackedRegions – 1, inclusive:

- For each sample location (xPackedPicture, yPackedPicture) belonging to the n -th packed region, the following applies:
 - The corresponding sample location (xProjPicture, yProjPicture) of the projected picture is derived as follows:
 - x is set equal to $xPackedPicture - PackedRegionLeft[n]$.
 - y is set equal to $yPackedPicture - PackedRegionTop[n]$.
 - Clause 8.15.1.4 is invoked with x , y , PackedRegionWidth[n], PackedRegionHeight[n], ProjRegionWidth[n], ProjRegionHeight[n] and TransformType[n] as inputs, and the output is assigned to sample location (hPos, vPos).
 - $xProjPicture$ is set equal to $ProjRegionLeft[n] + hPos$.
 - When StereoFlag is equal to 0 or TopBottomFlag is equal to 1, and when $xProjPicture$ is greater than or equal to $rwp_proj_picture_width$, $xProjPicture$ is set equal to $xProjPicture - rwp_proj_picture_width$.
 - When SideBySideFlag is equal to 1, the following applies:
 - When $ProjRegionLeft[n]$ is less than $rwp_proj_picture_width / 2$ and $xProjPicture$ is greater than or equal to $rwp_proj_picture_width / 2$, $xProjPicture$ is set equal to $xProjPicture - rwp_proj_picture_width / 2$.
 - When $ProjRegionLeft[n]$ is greater than or equal to $rwp_proj_picture_width / 2$ and $xProjPicture$ is greater than or equal to $rwp_proj_picture_width$, $xProjPicture$ is set equal to $xProjPicture - rwp_proj_picture_width / 2$.
 - $yProjPicture$ is set equal to $ProjRegionTop[n] + vPos$.
 - Clause 8.15.1.6 is invoked with $xProjPicture$, $yProjPicture$, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the luma sample location (xPackedPicture, yPackedPicture) belonging to the n -th packed region in the decoded picture.

Otherwise if RegionWisePackingFlag is equal 0 and CmpFlag is equal to 1, the following applies for each sample location (x, y) that is not a cubemap projection guard band sample within the cropped decoded picture:

- $xProjPicture$ is set equal to $x + 0.5$.
- $yProjPicture$ is set equal to $y + 0.5$.

- Clause 8.15.1.6 is invoked with `xProjPicture`, `yProjPicture`, `ConstituentPicWidth`, and `ConstituentPicHeight` as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the sample location (x, y) within the cropped decoded picture.

Otherwise (`RegionWisePackingFlag` is equal to 0, and `CmpFlag` is equal to 0), the following applies for each sample location (x, y) that is not an equirectangular projection guard band sample within the cropped decoded picture, where a sample location (x, y) is an equirectangular projection guard band sample when and only when `ErpFlag` is equal to 1, x is in the range of 0 to `erp_left_guard_band_width` – 1, inclusive, or `ConstituentPicWidth` – `erp_right_guard_band_width` to `ConstituentPicWidth` – 1, inclusive, and y is in the range of 0 to `ConstituentPicHeight` – 1, inclusive:

- `xProjPicture` is set equal to $x + 0.5$.
- `yProjPicture` is set equal to $y + 0.5$.
- If `ErpFlag` is equal to 0, `projPicWidth` is set equal to `ConstituentPicWidth`. Otherwise (`ErpFlag` is equal to 1), `projPicWidth` is set equal to `ConstituentPicWidth` – (`erp_left_guard_band_width` + `erp_right_guard_band_width`).
- Clause 8.15.1.6 is invoked with `xProjPicture`, `yProjPicture`, `projPicWidth`, and `ConstituentPicHeight` as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the sample location (x, y) within the region-wise packed picture.

8.15.1.6 Conversion from a sample location in a projected picture to sphere coordinates relative to the global coordinate axes

Inputs to this process are:

- the centre point of a sample location $(xProjPicture, yProjPicture)$ within a projected picture, where `xProjPicture` and `yProjPicture` are in relative projected picture sample units and could have non-integer real values, and
- `pictureWidth` and `pictureHeight`, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

Outputs of this process are:

- sphere coordinates (`azimuthGlobal`, `elevationGlobal`), in units of degrees relative to the global coordinate axes, and
- when `StereoFlag` is equal to 1, the index of the constituent picture (`constituentPicture`) equal to 0 or 1.

The outputs are derived with the following ordered steps:

1. `constituentPicture`, `xProjPicture`, and `yProjPicture` are conditionally set as follows:
 - If `xProjPicture` is greater than or equal to `pictureWidth` or `yProjPicture` is greater than or equal to `pictureHeight`, the following applies:
 - `constituentPicture` is set equal to 1.
 - When `xProjPicture` is greater than or equal to `pictureWidth`, `xProjPicture` is set to `xProjPicture` – `pictureWidth`.
 - When `yProjPicture` is greater than or equal to `pictureHeight`, `yProjPicture` is set to `yProjPicture` – `pictureHeight`.
 - Otherwise, `constituentPicture` is set equal to 0.
2. Clause 8.15.1.2 is invoked with `pictureWidth`, `pictureHeight`, `xProjPicture`, and `yProjPicture` as inputs, and the output is assigned to `azimuthLocal`, `elevationLocal`.
3. `azimuthGlobal` and `elevationGlobal` are set as follows:
 - If `RotationFlag` is equal to 1, clause 8.15.1.3 is invoked with `azimuthLocal`, `elevationLocal`, `RotationYaw`, `RotationPitch`, and `RotationRoll` as inputs, and the output is assigned to `azimuthGlobal` and `elevationGlobal`.
 - Otherwise, `azimuthGlobal` is set equal to `azimuthLocal` and `elevationGlobal` is set equal to `elevationLocal`.

8.15.1.7 Calculation of the cubemap face size for a projected picture

Inputs to this process are:

- `pictureWidth` and `pictureHeight`, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

Outputs of this process are:

- faceWidth and faceHeight, which are the width and height, respectively, of a projected face, in relative projected picture sample units.

The outputs are derived as follows:

```

gcmpPicWidth = pictureWidth
gcmpPicHeight = pictureHeight
gcmpGuardBandSamples = gcmp_guard_band_flag ? gcmp_guard_band_samples_minus1 + 1 : 0
if( gcmp_guard_band_flag && gcmp_guard_band_boundary_exterior_flag ) {
    gcmpPicWidth = pictureWidth - 2 * gcmpGuardBandSamples
    gcmpPicHeight = pictureHeight - 2 * gcmpGuardBandSamples
}
if( gcmp_packing_type == 0 ) {
    if( gcmp_guard_band_flag )
        gcmpPicHeight -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth
    faceHeight = gcmpPicHeight / 6
} else if( gcmp_packing_type == 1 ) {
    if( gcmp_guard_band_flag )
        gcmpPicWidth -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth / 2
    faceHeight = gcmpPicHeight / 3
} else if( gcmp_packing_type == 2 ) {
    if( gcmp_guard_band_flag )
        gcmpPicHeight -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth / 3
    faceHeight = gcmpPicHeight / 2
} else if( gcmp_packing_type == 3 ) {
    if( gcmp_guard_band_flag )
        gcmpPicWidth -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth / 6
    faceHeight = gcmpPicHeight
} else if( gcmp_packing_type == 4 ) {
    if( gcmp_guard_band_flag )
        gcmpPicWidth -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth / 3
    faceHeight = gcmpPicHeight
} else if( gcmp_packing_type == 5 ) {
    if( gcmp_guard_band_flag )
        gcmpPicHeight -= 2 * gcmpGuardBandSamples
    faceWidth = gcmpPicWidth
    faceHeight = gcmpPicHeight / 3
}

```

(46)

The values of faceWidth and faceHeight are constrained as follows:

- If gcmp_packing_type is equal to 4, the following constraints apply:
 - When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), faceWidth shall be a multiple of 4 in units of luma samples.
 - When ChromaFormatIdc is equal to 1 (4:2:0 chroma format), faceHeight shall be a multiple of 2 in units of luma samples.
- Otherwise, if gcmp_packing_type is equal to 5, the following constraints apply:
 - When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), faceWidth shall be a multiple of 2 in units of luma samples.
 - When ChromaFormatIdc is equal to 1 (4:2:0 chroma format), faceHeight shall be a multiple of 4 in units of luma samples.
- Otherwise, the following constraints apply:
 - When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), faceWidth shall be a multiple of 2 in units of luma samples.

- When ChromaFormatIdc is equal to 1 (4:2:0 chroma format), faceHeight shall be a multiple of 2 in units of luma samples.

It is a requirement of bitstream conformance that the following constraints apply:

- If gcmp_packing_type is equal to 0, gcmpPicHeight shall be a multiple of 6, and gcmpPicWidth shall be equal to gcmpPicHeight / 6.
- Otherwise, if gcmp_packing_type is equal to 1, gcmpPicWidth shall be a multiple of 2 and gcmpPicHeight shall be a multiple of 3, and gcmpPicWidth / 2 shall be equal to gcmpPicHeight / 3.
- Otherwise, if gcmp_packing_type is equal to 2, gcmpPicWidth shall be a multiple of 3 and gcmpPicHeight shall be a multiple of 2, and gcmpPicWidth / 3 shall be equal to gcmpPicHeight / 2.
- Otherwise, if gcmp_packing_type is equal to 3, gcmpPicWidth shall be a multiple of 6, and gcmpPicWidth / 6 shall be equal to gcmpPicHeight.
- Otherwise, if gcmp_packing_type is equal to 4, gcmpPicWidth shall be a multiple of 6, and gcmpPicWidth / 3 shall be equal to gcmpPicHeight.
- Otherwise, if gcmp_packing_type is equal to 5, gcmpPicHeight shall be a multiple of 6, and gcmpPicWidth shall be equal to gcmpPicHeight / 3.

8.15.1.8 Conversion from a sample location in a projected picture to a sample location in a projected cubemap face

Inputs to this process are:

- sample location (hPos, vPos) within the projected picture in relative projected picture sample units, where hPos and vPos could have non-integer real values, and
- faceWidth and faceHeight, which are the width and height, respectively, of the projected face, in relative projected picture sample units.

Outputs of this process are:

- the sample location (hPosFace, vPosFace) within the projected face in relative projected picture sample units, where hPosFace and vPosFace could have non-integer real values.

The outputs are derived as follows:

```

gbSize = gcmpGuardBandSamples
tmpHorPos = hPos
tmpVerPos = vPos
if( gcmp_guard_band_flag ) {
    if( gcmp_guard_band_boundary_exterior_flag ) {
        tmpHorPos = hPos - gbSize
        tmpVerPos = vPos - gbSize
    }
    if( gcmp_packing_type == 0 )
        tmpVerPos = tmpVerPos < 3 * faceHeight ? tmpVerPos : tmpVerPos - 2 * gbSize
    else if( gcmp_packing_type == 1 )
        tmpHorPos = tmpHorPos < faceWidth ? tmpHorPos : tmpHorPos - 2 * gbSize
    else if( gcmp_packing_type == 2 )
        tmpVerPos = tmpVerPos < faceHeight ? tmpVerPos : tmpVerPos - 2 * gbSize
    else if( gcmp_packing_type == 3 )
        tmpHorPos = tmpHorPos < 3 * faceWidth ? tmpHorPos : tmpHorPos - 2 * gbSize
    else if( gcmp_packing_type == 4 )
        tmpHorPos = tmpHorPos < faceWidth / 2 ? tmpHorPos : tmpHorPos < 2.5 * faceWidth + gbSize ?
            tmpHorPos - gbSize : tmpHorPos - 2 * gbSize
    else if( gcmp_packing_type == 5 )
        tmpVerPos = tmpVerPos < faceHeight / 2 ? tmpVerPos : tmpVerPos < 2.5 * faceHeight + gbSize ?
            tmpVerPos - gbSize : tmpVerPos - 2 * gbSize
}
w = Floor( tmpHorPos ÷ faceWidth )
h = Floor( tmpVerPos ÷ faceHeight )
hPosFace = tmpHorPos - w * faceWidth
vPosFace = tmpVerPos - h * faceHeight

```

8.15.1.9 Rotation of sample locations for a projected cubemap face

Inputs to this process are:

- sample location (hPosFace, vPosFace) within the n-th projected face in relative projected picture sample units, where hPosFace and vPosFace could have non-integer real values, and
- faceWidth and faceHeight, which are the width and height, respectively, of the projected face, in relative projected picture sample units.

Outputs of this process are:

- the rotated sample location (hPosRot, vPosRot) within the projected face in relative projected picture sample units, where hPosRot and vPosRot could have non-integer real values.

The outputs are derived as follows:

```
if( gcmp_face_rotation[ n ] == 0 ) {
    hPosRot = hPosFace
    vPosRot = vPosFace
} else if( gcmp_face_rotation[ n ] == 1 ) {
    hPosRot = vPosFace
    vPosRot = faceWidth - hPosFace
} else if( gcmp_face_rotation[ n ] == 2 ) {
    hPosRot = faceWidth - hPosFace
    vPosRot = faceHeight - vPosFace
} else if( gcmp_face_rotation[ n ] == 3 ) {
    hPosRot = faceHeight - vPosFace
    vPosRot = hPosFace
}
}
```

(48)

8.15.1.10 Adjustment of a sample location for hemisphere cubemap projection

Inputs to this process are:

- sample location (hPosRot, vPosRot) within the n-th projected face in relative projected picture sample units, where hPosRot and vPosRot could have non-integer real values, and
- faceWidth and faceHeight, which are the width and height, respectively, of the projected face, in relative projected picture sample units.

Outputs of this process are:

- the adjusted sample location (hPosAdj, vPosAdj) within the n-th projected face in relative projected picture sample units, where hPosAdj and vPosAdj could have non-integer real values.

The outputs are derived as follows:

```
leftFaceIdx = {5, 3, 1, 0, 0, 1}
rightFaceIdx = {4, 2, 0, 1, 1, 0}
topFaceIdx = {2, 4, 4, 4, 2, 2}
bottomFaceIdx = {3, 5, 5, 5, 3, 3}
hPosAdj = hPosRot
vPosAdj = vPosRot
if( n != 2 )
    if( face_index[ 2 ] == leftFaceIdx[ face_index[ n ] ] && hPosAdj >= faceWidth / 2 )
        hPosAdj -= faceWidth / 2
    else if( face_index[ 2 ] == rightFaceIdx[ face_index[ n ] ] && hPosAdj < faceWidth / 2 )
        hPosAdj += faceWidth / 2
    else if( face_index[ 2 ] == topFaceIdx[ face_index[ n ] ] && vPosAdj >= faceHeight / 2 )
        vPosAdj -= faceHeight / 2
    else if( face_index[ 2 ] == bottomFaceIdx[ face_index[ n ] ] && vPosAdj < faceHeight / 2 )
        vPosAdj += faceHeight / 2
```

(49)

8.15.2 Equirectangular projection SEI message

8.15.2.1 Equirectangular projection SEI message syntax

	Descriptor
equirectangular_projection(payloadSize) {	
erp_cancel_flag	u(1)
if(!erp_cancel_flag) {	
erp_persistence_flag	u(1)
erp_guard_band_flag	u(1)
erp_reserved_zero_2bits	u(2)
if(erp_guard_band_flag == 1) {	
erp_guard_band_type	u(3)
erp_left_guard_band_width	u(8)
erp_right_guard_band_width	u(8)
}	
}	
}	

8.15.2.2 Equirectangular projection SEI message semantics

The equirectangular projection SEI message provides information to enable remapping (through an equirectangular projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ , θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

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

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

When an equirectangular projection SEI message is present for any picture of a CLVS, an equirectangular projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When the SAR for a picture is indicated by `vui_aspect_ratio_idc` or `sari_aspect_ratio_idc` greater than 1, there should be no equirectangular projection SEI messages applicable for the picture.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

- The value of `fp_arrangement_cancel_flag` is equal to 0.
- The value of `fp_arrangement_type` is equal to 3, 4, or 5.
- The value of `fp_quincunx_sampling_flag` is equal to 0.
- The value of `fp_spatial_flipping_flag` is equal to 0.
- The value of `fp_field_views_flag` is equal to 0.
- The value of `fp_frame0_grid_position_x` is equal to 0.
- The value of `fp_frame0_grid_position_y` is equal to 0.
- The value of `fp_frame1_grid_position_x` is equal to 0.
- The value of `fp_frame1_grid_position_y` is equal to 0.

When a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 that applies to the picture shall not be present. Decoders shall ignore equirectangular projection SEI messages when a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

erp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous equirectangular projection SEI message in output order. **erp_cancel_flag** equal to 0 indicates that equirectangular projection information follows.

erp_persistence_flag specifies the persistence of the equirectangular projection SEI message for the current layer.

erp_persistence_flag equal to 0 specifies that the equirectangular projection SEI message applies to the current decoded picture only.

erp_persistence_flag equal to 1 specifies that the equirectangular projection 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 equirectangular projection SEI message is output that follows the current picture in output order.

erp_guard_band_flag equal to 1 indicates that the constituent picture contains guard band areas for which the sizes are specified by the syntax elements **erp_left_guard_band_width** and **erp_right_guard_band_width**. **erp_guard_band_flag** equal to 0 indicates that the constituent picture does not contains guard band areas for which the sizes are specified by the syntax elements **erp_left_guard_band_width** and **erp_right_guard_band_width**.

erp_reserved_zero_2bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **erp_reserved_zero_2bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **erp_reserved_zero_2bits**.

erp_guard_band_type indicates the type of the guard bands as follows:

- **erp_guard_band_type** equal to 0 indicates that the content of the guard band in relation to the content of the constituent picture is unknown or unspecified or specified by other means not specified in this Specification.
- **erp_guard_band_type** equal to 1 indicates that the content of the guard band suffices for interpolation of sample values at sub-pel sample fractional locations within the constituent picture.

NOTE – **erp_guard_band_type** equal to 1 could be used when the source boundary samples of a constituent picture have been copied horizontally to the guard band.

- **erp_guard_band_type** equal to 2 indicates that the content of the guard band represents actual picture content at a quality that gradually changes from the picture quality of the constituent picture to that of the spherically adjacent region.
- **erp_guard_band_type** equal to 3 indicates that the content of the guard bands represents actual picture content at a similar level of quality as the constituent picture.
- **erp_guard_band_type** values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of **erp_guard_band_type** when the value is greater than 3 as equivalent to the value 0.

erp_left_guard_band_width specifies the width of the guard band on the left side of the constituent picture in units of luma samples. When **erp_guard_band_flag** is equal to 0, the value of **erp_left_guard_band_width** is inferred to be equal to 0. When **ChromaFormatIdc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **erp_left_guard_band_width** shall be an even number.

erp_right_guard_band_width specifies the width of the guard band on the right side of the constituent picture in units of luma samples. When **erp_guard_band_flag** is equal to 0, the value of **erp_right_guard_band_width** is inferred to be equal to 0. When **ChromaFormatIdc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **erp_right_guard_band_width** shall be an even number.

8.15.3 Generalized cubemap projection SEI message

8.15.3.1 Generalized cubemap projection SEI message syntax

generalized_cubemap_projection(payloadSize) {	Descriptor
gcmp_cancel_flag	u(1)
if(!gcmp_cancel_flag) {	
gcmp_persistence_flag	u(1)

gcmp_packing_type	u(3)
gcmp_mapping_function_type	u(2)
for(i = 0; i < (gcmp_packing_type == 4 gcmp_packing_type == 5) ? 5 : 6; i++) {	
gcmp_face_index[i]	u(3)
gcmp_face_rotation[i]	u(2)
if(gcmp_mapping_function_type == 2) {	
gcmp_function_coeff_u[i]	u(7)
gcmp_function_u_affected_by_v_flag[i]	u(1)
gcmp_function_coeff_v[i]	u(7)
gcmp_function_v_affected_by_u_flag[i]	u(1)
}	
}	
gcmp_guard_band_flag	u(1)
if(gcmp_guard_band_flag) {	
gcmp_guard_band_type	u(3)
gcmp_guard_band_boundary_exterior_flag	u(1)
gcmp_guard_band_samples_minus1	u(4)
}	
}	
}	

8.15.3.2 Generalized cubemap projection SEI message semantics

The generalized cubemap projection SEI message provides information to enable remapping (through a generalized cubemap projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ , θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

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

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

When a generalized cubemap projection SEI message is present for any picture of a CLVS, a generalized cubemap projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When the SAR for a picture is indicated by `vui_aspect_ratio_idc` or `sari_aspect_ratio_idc` greater than 1, there should be no generalized cubemap projection SEI messages applicable for the picture.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

- The value of `fp_arrangement_cancel_flag` is equal to 0.
- The value of `fp_arrangement_type` is equal to 3, 4, or 5.
- The value of `fp_quincunx_sampling_flag` is equal to 0.
- The value of `fp_spatial_flipping_flag` is equal to 0.
- The value of `fp_field_views_flag` is equal to 0.
- The value of `fp_frame0_grid_position_x` is equal to 0.
- The value of `fp_frame0_grid_position_y` is equal to 0.
- The value of `fp_frame1_grid_position_x` is equal to 0.
- The value of `fp_frame1_grid_position_y` is equal to 0.

When a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, a generalized cubemap projection SEI message with `gcmp_cancel_flag` equal to 0 that applies to the picture shall not be present. Decoders shall ignore generalized cubemap projection SEI messages when a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

When all of the following conditions are true, the functionality of the generalized cubemap projection SEI message is exactly the same as the cubemap projection SEI message specified in in Rec. ITU-T H.265 | ISO/IEC 23008-2 and Rec. ITU-T H.264 | ISO/IEC 14496-10:

- The value of `gcmp_packing_type` is equal to 2;
- The value of `gcmp_mapping_function_type` is equal to 0;
- The values of `gcmp_face_index[i]` for `i` from 0 to 5, inclusive, are equal to 5, 0, 4, 3, 1 and 2, respectively;
- The value of `gcmp_face_rotation[i]` is equal to 0 for each value of `i` in the range of 0 to 5, inclusive;
- The value of `gcmp_guard_band_flag` is equal to 0.

`gcmp_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous generalized cubemap projection SEI message in output order. `gcmp_cancel_flag` equal to 0 indicates that cubemap projection information follows.

`gcmp_persistence_flag` specifies the persistence of the generalized cubemap projection SEI message for the current layer.

`gcmp_persistence_flag` equal to 0 specifies that the generalized cubemap projection SEI message applies to the current decoded picture only.

`gcmp_persistence_flag` equal to 1 specifies that the generalized cubemap projection SEI message applies to the current decoded picture and persists 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 cubemap projection SEI message is output that follows the current picture in output order.

`gcmp_packing_type` specifies the packing type and the position index of the cubemap packing as specified in Table 10. When the value of `gcmp_packing_type` is in the range of 0 to 3, inclusive, cubemap packing with six faces is used. When `gcmp_packing_type` is 4 or 5, hemisphere cubemap packing with one full face and four half faces is used. The value of `gcmp_packing_type` shall be in the range of 0 to 5, inclusive. Other values for `gcmp_packing_type` are reserved for future use by ITU-T | ISO/IEC.

Table 10 – Specification of packing type and position index based on `gcmp_packing_type`

<code>gcmp_packing_type</code>	Packing type and position index
0	0
	1
	2
	3
	4
	5

Table 10 – Specification of packing type and position index based on gcmp_packing_type

gcmp_packing_type	Packing type and position index						
1	<table border="1"> <tr><td>0</td><td>1</td></tr> <tr><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td></tr> </table>	0	1	2	3	4	5
0	1						
2	3						
4	5						
2	<table border="1"> <tr><td>0</td><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td><td>5</td></tr> </table>	0	1	2	3	4	5
0	1	2					
3	4	5					
3	<table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> </table>	0	1	2	3	4	5
0	1	2	3	4	5		
4	<table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>	0	1	2	3	4	
0	1	2	3	4			
5	<table border="1"> <tr><td>0</td></tr> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table>	0	1	2	3	4	
0							
1							
2							
3							
4							

gcmp_mapping_function_type specifies the mapping function used to adjust the sample locations of the cubemap projection. **gcmp_mapping_function_type** equal to 0 specifies that the same mapping function as specified for the cubemap projection SEI message in Rec. ITU-T H.265 | ISO/IEC 23008-2 and Rec. ITU-T H.264 | ISO/IEC 14496-10 is used. **gcmp_mapping_function_type** equal to 1 specifies that the equi-angular mapping function is applied to adjust the sample locations of the projected face, as defined in clause 8.15.1.2. **gcmp_mapping_function_type** equal to 2 specifies that the coefficients of the mapping function applied to adjust the sample locations of the *i*-th projected face are specified by the syntax elements **gcmp_function_coeff_u[i]**, **gcmp_function_u_affected_by_v_flag[i]**, **gcmp_function_coeff_v[i]**, and **gcmp_function_v_affected_by_u_flag[i]**. The value of **gcmp_mapping_function_type** shall be in the range of 0 to 2, inclusive.

gcmp_face_index[i] specifies the face index for position index *i* in **gcmp_packing_type** and the relationship between the global coordinates 3D (X, Y, Z) and the local coordinate 2D (u, v) as specified in clause 8.15.1.2.

When **gcmp_packing_type** is equal to 4 or 5, it is a requirement of bitstream conformance that the following constraints apply:

- If **gcmp_face_index[2]** is equal to 0 or 1, the value of **gcmp_face_index[i]** for *i* equal to 0, 1, 3 or 4 shall be in the range of 2 to 5, inclusive.
- Otherwise, if **gcmp_face_index[2]** is equal to 2 or 3, the value of **gcmp_face_index[i]** for *i* equal to 0, 1, 3 or 4 shall be 0, 1, 4, or 5.
- Otherwise, the value of **gcmp_face_index[i]** for *i* equal to 0, 1, 3 or 4 shall be in the range of 0 to 3, inclusive.

gcmp_face_rotation[i] specifies the rotation to be applied to the face on position index *i* as specified in Table 11.

Table 11 – Specification of counterclockwise rotation angle based on `gcmp_face_rotation[i]`

<code>gcmp_face_rotation[i]</code>	Rotation angle in degree (anticlockwise)
0	0
1	90
2	180
3	270

When `gcmp_packing_type` is equal to 4, it is a requirement of bitstream conformance that the following constraints apply:

- If `gcmp_face_index[2]` is equal to 0 or 1, the value of `gcmp_face_rotation[i]` for `i` equal to 0, 1, 3 or 4 shall be 0 or 2.
- Otherwise, if `gcmp_face_index[2]` is equal to 2 or 3, when `gcmp_face_index[i]` is equal to 1, the value of `gcmp_face_rotation[i]` shall be 0 or 2, and when `gcmp_face_index[i]` is equal to 0, 4 or 5, the value of `gcmp_face_rotation[i]` shall be 1 or 3.
- Otherwise, when `gcmp_face_index[i]` is equal to 0, the value of `gcmp_face_rotation[i]` shall be 0 or 2, and when `gcmp_face_index[i]` is equal to 1, 2 or 3, the value of `gcmp_face_rotation[i]` shall be 1 or 3.

When `gcmp_packing_type` is equal to 5, it is a requirement of bitstream conformance that the following constraints apply:

- If `gcmp_face_index[2]` is equal to 0 or 1, the value of `gcmp_face_rotation[i]` for `i` equal to 0, 1, 3 or 4 shall be 1 or 3.
- Otherwise, if `gcmp_face_index[2]` is equal to 2 or 3, when `gcmp_face_index[i]` is equal to 1, the value of `gcmp_face_rotation[i]` shall be 1 or 3, and when `gcmp_face_index[i]` is equal to 0, 4 or 5, the value of `gcmp_face_rotation[i]` shall be 0 or 2.
- Otherwise, when `gcmp_face_index[i]` is equal to 0, the value of `gcmp_face_rotation[i]` shall be 1 or 3, and when `gcmp_face_index[i]` is equal to 1, 2 or 3, the value of `gcmp_face_rotation[i]` shall be 0 or 2.

`gcmp_function_coeff_u[i]` specifies the coefficient used in the cubemap mapping function of the u-axis of the `i`-th face. When `gcmp_function_coeff_u[i]` is not present, it is inferred to be equal to 0.

`gcmp_function_u_affected_by_v_flag[i]` equal to 1 indicates that the cubemap mapping function of the u-axis refers to the `v` position of the sample location. `gcmp_function_u_affected_by_v_flag[i]` equal to 0 indicates that the cubemap mapping function in u-axis does not refer to the `v` position of the sample location.

`gcmp_function_coeff_v[i]` specifies the coefficient used in the cubemap mapping function of the v-axis of the `i`-th face. When `gcmp_function_coeff_v[i]` is not present, it is inferred to be equal to 0.

`gcmp_function_v_affected_by_u_flag[i]` equal to 1 indicates that the cubemap mapping function of the v-axis refers to the `u` position of the sample location. `gcmp_function_v_affected_by_u_flag[i]` equal to 0 indicates that the cubemap mapping function in v-axis does not refer to the `u` position of the sample location.

`gcmp_guard_band_flag` equal to 0 indicates that the coded picture does not contain guard band areas. `gcmp_guard_band_flag` equal to 1 indicates that the coded picture contains guard band areas for which the sizes are specified by the syntax element `gcmp_guard_band_samples_minus1`.

`gcmp_guard_band_type` indicates the type of the guard bands as follows:

- `gcmp_guard_band_type` equal to 0 indicates that the content of the guard bands in relation to the content of the coded face is unknown or unspecified or specified by other means not specified in this Specification.
- `gcmp_guard_band_type` equal to 1 indicates that the content of the guard bands suffices for interpolation of sample values at sub-pel sample fractional locations within the coded face.

NOTE – `gcmp_guard_band_type` equal to 1 could be used when the source boundary samples of a coded face have been copied horizontally or vertically to the guard band.

- `gcmp_guard_band_type` equal to 2 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the coded face at quality that gradually changes from the picture quality of the coded face to that of the spherically adjacent region.
- `gcmp_guard_band_type` equal to 3 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the coded face at a similar picture quality as within the coded face.
- `gcmp_guard_band_type` values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of `gcmp_guard_band_type` when the value is greater than 3 as equivalent to the value 0.

gcmp_guard_band_boundary_exterior_flag indicates which face boundaries contain guard bands, as specified in Table 12.

Table 12 – Specification of guard band boundary location based on gcmp_packing_type and gcmp_guard_band_boundary_exterior_flag

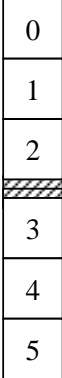
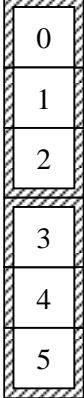
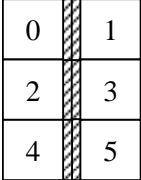
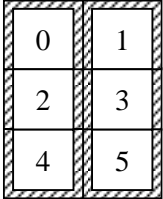
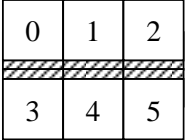
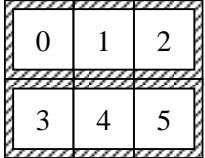
gcmp_packing_type	gcmp_guard_band_boundary_exterior_flag	Location of guard band
0	0	
0	1	
1	0	
1	1	
2	0	
2	1	

Table 12 – Specification of guard band boundary location based on `gcmp_packing_type` and `gcmp_guard_band_boundary_exterior_flag`

<code>gcmp_packing_type</code>	<code>gcmp_guard_band_boundary_exterior_flag</code>	Location of guard band
3	0	
3	1	
4	0	
4	1	
5	0	
5	1	

`gcmp_guard_band_samples_minus1` plus 1 specifies the number of guard band samples, in units of luma samples, used in the cubemap projected picture. When `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), `gcmp_guard_band_samples_minus1` plus 1 shall correspond to an even number of luma samples within the cropped decoded picture.

8.15.4 Sphere rotation SEI message

8.15.4.1 Sphere rotation SEI message syntax

Descriptor	Descriptor
<code>sphere_rotation(payloadSize) {</code>	
sphere_rotation_cancel_flag	u(1)
if(!sphere_rotation_cancel_flag) {	
sphere_rotation_persistence_flag	u(1)
sphere_rotation_reserved_zero_6bits	u(6)
yaw_rotation	i(32)
pitch_rotation	i(32)
roll_rotation	i(32)
}	
}	

8.15.4.2 Sphere rotation SEI message semantics

The sphere rotation SEI message provides information on rotation angles yaw (α), pitch (β), and roll (γ) that are used for conversion between the global coordinate axes and the local coordinate axes.

Relative to an (x, y, z) Cartesian coordinate system, yaw expresses a rotation around the z (vertical, up) axis, pitch rotates around the y (lateral, side-to-side) axis, and roll rotates around the x (back-to-front) axis. Rotations are extrinsic, i.e., around x, y, and z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis.

sphere_rotation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous sphere rotation SEI message in output order. **sphere_rotation_cancel_flag** equal to 0 indicates that sphere rotation information follows.

sphere_rotation_persistence_flag specifies the persistence of the sphere rotation SEI message for the current layer.

sphere_rotation_persistence_flag equal to 0 specifies that the sphere rotation SEI message applies to the current decoded picture only.

sphere_rotation_persistence_flag equal to 1 specifies that the sphere rotation 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 sphere rotation SEI message is output that follows the current picture in output order.

When no omnidirectional video projection is indicated to apply to a picture, e.g., by an equirectangular projection SEI message with **erp_cancel_flag** equal to 0 or a generalized cubemap projection SEI message with **gcmp_cancel_flag** equal to 0 being present in the CLVS that applies to the picture, a sphere rotation SEI message with **sphere_rotation_cancel_flag** equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore sphere rotation SEI messages with **sphere_rotation_cancel_flag** equal to 0 for pictures to which no omnidirectional video projection is indicated to apply.

sphere_rotation_reserved_zero_6bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **sphere_rotation_reserved_zero_6bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **sphere_rotation_reserved_zero_6bits**.

yaw_rotation specifies the value of the yaw rotation angle, in units of 2^{-16} degrees. The value of **yaw_rotation** shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of **yaw_rotation** is inferred to be equal to 0.

pitch_rotation specifies the value of the pitch rotation angle, in units of 2^{-16} degrees. The value of **pitch_rotation** shall be in the range of $-90 * 2^{16}$ (i.e., -5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive. When not present, the value of **pitch_rotation** is inferred to be equal to 0.

roll_rotation specifies the value of the roll rotation angle, in units of 2^{-16} degrees. The value of **roll_rotation** shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of **roll_rotation** is inferred to be equal to 0.

8.15.5 Region-wise packing SEI message

8.15.5.1 Region-wise packing SEI message syntax

	Descriptor
regionwise_packing(payloadSize) {	
rwp_cancel_flag	u(1)
if(!rwp_cancel_flag) {	
rwp_persistence_flag	u(1)
rwp_constituent_picture_matching_flag	u(1)
rwp_reserved_zero_5bits	u(5)
rwp_num_packed_regions	u(8)
rwp_proj_picture_width	u(32)
rwp_proj_picture_height	u(32)

rwp_packed_picture_width	u(16)
rwp_packed_picture_height	u(16)
for(i = 0; i < rwp_num_packed_regions; i++) {	
rwp_reserved_zero_4bits[i]	u(4)
rwp_transform_type[i]	u(3)
rwp_guard_band_flag[i]	u(1)
rwp_proj_region_width[i]	u(32)
rwp_proj_region_height[i]	u(32)
rwp_proj_region_top[i]	u(32)
rwp_proj_region_left[i]	u(32)
rwp_packed_region_width[i]	u(16)
rwp_packed_region_height[i]	u(16)
rwp_packed_region_top[i]	u(16)
rwp_packed_region_left[i]	u(16)
if(rwp_guard_band_flag[i]) {	
rwp_left_guard_band_width[i]	u(8)
rwp_right_guard_band_width[i]	u(8)
rwp_top_guard_band_height[i]	u(8)
rwp_bottom_guard_band_height[i]	u(8)
rwp_guard_band_not_used_for_pred_flag[i]	u(1)
for(j = 0; j < 4; j++)	
rwp_guard_band_type[i][j]	u(3)
rwp_guard_band_reserved_zero_3bits[i]	u(3)
}	
}	
}	

8.15.5.2 Region-wise packing SEI message semantics

The RWP SEI message provides information to enable remapping of the colour samples of the cropped decoded pictures onto projected pictures as well as information on the location and size of the guard bands, if any.

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

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

rwp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous RWP SEI message in output order. **rwp_cancel_flag** equal to 0 indicates that RWP information follows.

rwp_persistence_flag specifies the persistence of the RWP SEI message for the current layer.

rwp_persistence_flag equal to 0 specifies that the RWP SEI message applies to the current decoded picture only.

rwp_persistence_flag equal to 1 specifies that the RWP 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 RWP SEI message is output that follows the current picture in output order.

When no omnidirectional video projection is indicated to apply to a picture, e.g., by an equirectangular projection SEI message with **erp_cancel_flag** equal to 0 or a generalized cubemap projection SEI message with **gcmap_cancel_flag** equal to 0 being present in the CLVS that applies to the current picture, a RWP SEI message with **rwp_cancel_flag** equal to 0 shall not be present in the CLVS that applies to the picture. Decoders shall ignore RWP SEI messages with **rwp_cancel_flag** equal to 0 for pictures to which no omnidirectional video projection is indicated to apply.

When an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 and `erp_guard_band_flag` equal to 1 is present in the CLVS that applies to the current picture, a RWP SEI message with `rwp_cancel_flag` equal to 0 shall not be present in the CLVS that applies to the current picture.

When a generalized cubemap projection SEI message with `gmcp_cancel_flag` equal to 0 is present in the CLVS that applies to the current picture and precedes the RWP SEI message in decoding order, a RWP SEI message with `rwp_cancel_flag` equal to 0 shall not be present in the CLVS that applies to the current picture unless all the following conditions are true for the generalized cubemap projection SEI message:

- The value of `gmcp_packing_type` is equal to 2;
- The values of `gmcp_face_index[i]` for `i` from 0 to 5, inclusive, are equal to 5, 0, 4, 3, 1 and 2, respectively;
- The value of `gmcp_face_rotation[i]` is equal to 0 for each value of `i` in the range of 0 to 5, inclusive;
- The value of `gmcp_guard_band_flag` is equal to 0.

For the frame packing arrangement scheme indicated by a frame packing arrangement SEI message that applies to the current picture, if a RWP SEI message with `rwp_cancel_flag` equal to 0 is present that applies to the current picture, the frame packing arrangement scheme applies to the projected picture, otherwise, the frame packing arrangement scheme applies to the cropped decoded picture.

If a frame packing arrangement SEI message with `fp_arrangement_cancel_flag` equal to 0, `fp_arrangement_type` equal to 3, 4, or 5, and `fp_quincunx_sampling_flag` equal to 0 is not present that applies to the current picture, the variables `StereoFlag`, `TopBottomFlag`, `SideBySideFlag`, and `TempInterleavingFlag` are all set equal to 0, the variables `HorDiv1` and `VerDiv1` are both set equal to 1. Otherwise the following applies:

- `StereoFlag` is set equal to 1.
- When the `fp_arrangement_type` is equal to 3, `SideBySideFlag` is set equal to 1, `TopBottomFlag` and `TempInterleavingFlag` are both set equal to 0, `HorDiv1` is set equal to 2 and `VerDiv1` is set equal to 1.
- When the `fp_arrangement_type` is equal to 4, `TopBottomFlag` is set equal to 1, `SideBySideFlag` and `TempInterleavingFlag` are both set equal to 0, `HorDiv1` is set equal to 1 and `VerDiv1` is set equal to 2.
- When the `fp_arrangement_type` is equal to 5, `TempInterleavingFlag` is set equal to 1, `TopBottomFlag` and `SideBySideFlag` are both set equal to 0, `HorDiv1` and `VerDiv1` are both set equal to 1.

`rwp_constituent_picture_matching_flag` equal to 1 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply individually to each constituent picture and that the packed picture and the projected picture have the same stereoscopic frame packing format indicated by the frame packing arrangement SEI message. `rwp_constituent_picture_matching_flag` equal to 0 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply to the projected picture.

When either of the following conditions is true, the value of `rwp_constituent_picture_matching_flag` shall be equal to 0:

- `StereoFlag` is equal to 0.
- `StereoFlag` is equal to 1 and `fp_arrangement_type` is equal to 5.

`rwp_reserved_zero_5bits` shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `rwp_reserved_zero_5bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `rwp_reserved_zero_5bits[i]`.

`rwp_num_packed_regions` specifies the number of packed regions when `rwp_constituent_picture_matching_flag` is equal to 0. The value of `rwp_num_packed_regions` shall be greater than 0. When `rwp_constituent_picture_matching_flag` is equal to 1, the total number of packed regions is equal to `rwp_num_packed_regions * 2`, and the information in each entry of the loop of `rwp_num_packed_regions` entries applies to each constituent picture of the projected picture and the packed picture.

`rwp_proj_picture_width` and **`rwp_proj_picture_height`** specify the width and height, respectively, of the projected picture, in relative projected picture sample units.

NOTE 1 – Relative project picture sample unit is the unit used for the width or height of a projected picture or projected region. When a projected picture is a region-wise packed picture (i.e., there is a one-to-one mapping between the projected picture samples and the region-wise packed picture samples and a relative project picture sample unit is equivalent to a relative region-wise packed picture sample unit), `rwp_proj_picture_width` and `rwp_proj_picture_height` would have such values that `rwp_proj_picture_width` is an integer multiple of `cropPicWidth` and `rwp_proj_picture_height` is an integer multiple of `cropPicHeight`, where `cropPicWidth` and `cropPicHeight` are the width and height, respectively, of the cropped decoded picture, in units of luma samples.

The values of `rwp_proj_picture_width` and `rwp_proj_picture_height` shall both be greater than 0.

rwp_packed_picture_width and **rwp_packed_picture_height** specify the width and height, respectively, of the packed picture, in relative region-wise packed picture sample units.

The values of **rwp_packed_picture_width** and **rwp_packed_picture_height** shall both be greater than 0.

It is a requirement of bitstream conformance that **rwp_packed_picture_width** and **rwp_packed_picture_height** shall have such values that **rwp_packed_picture_width** is an integer multiple of **cropPicWidth** and **rwp_packed_picture_height** is an integer multiple of **cropPicHeight**, where **cropPicWidth** and **cropPicHeight** are the width and height, respectively, of the cropped decoded picture, in units of luma samples.

rwp_reserved_zero_4bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **rwp_reserved_zero_4bits[i]** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **rwp_reserved_zero_4bits[i]**.

rwp_transform_type[i] specifies the rotation and mirroring to be applied to the *i*-th packed region to remap to the *i*-th projected region. When **rwp_transform_type[i]** specifies both rotation and mirroring, rotation applies before mirroring. The values of **rwp_transform_type[i]** are specified in Table 13.

Table 13 – rwp_transform_type[i] values

Value	Description
0	no transform
1	mirroring horizontally
2	rotation by 180 degrees (anticlockwise)
3	rotation by 180 degrees (anticlockwise) before mirroring horizontally
4	rotation by 90 degrees (anticlockwise) before mirroring horizontally
5	rotation by 90 degrees (anticlockwise)
6	rotation by 270 degrees (anticlockwise) before mirroring horizontally
7	rotation by 270 degrees (anticlockwise)

rwp_guard_band_flag[i] equal to 0 specifies that the *i*-th packed region does not have a guard band. **rwp_guard_band_flag[i]** equal to 1 specifies that the *i*-th packed region has a guard band.

rwp_proj_region_width[i], **rwp_proj_region_height[i]**, **rwp_proj_region_top[i]** and **rwp_proj_region_left[i]** specify the width, height, top sample row, and the left-most sample column, respectively, of the *i*-th projected region, either within the projected picture (when **rwp_constituent_picture_matching_flag** is equal to 0) or within the constituent picture of the projected picture (when **rwp_constituent_picture_matching_flag** is equal to 1).

rwp_proj_region_width[i], **rwp_proj_region_height[i]**, **rwp_proj_region_top[i]**, and **rwp_proj_region_left[i]** are indicated in relative projected picture sample units.

NOTE 2 – Two projected regions could partially or entirely overlap with each other.

rwp_packed_region_width[i], **rwp_packed_region_height[i]**, **rwp_packed_region_top[i]**, and **rwp_packed_region_left[i]** specify the width, height, the top luma sample row, and the left-most luma sample column, respectively, of the packed region, either within the region-wise packed picture (when **rwp_constituent_picture_matching_flag** is equal to 0) or within each constituent picture of the region-wise packed picture (when **rwp_constituent_picture_matching_flag** is equal to 1).

rwp_packed_region_width[i], **rwp_packed_region_height[i]**, **rwp_packed_region_top[i]**, and **rwp_packed_region_left[i]** are indicated in relative region-wise packed picture sample units. **rwp_packed_region_width[i]**, **rwp_packed_region_height[i]**, **rwp_packed_region_top[i]**, and **rwp_packed_region_left[i]** shall represent integer horizontal and vertical coordinates of luma sample units within the cropped decoded pictures.

NOTE 3 – Two packed regions could partially or entirely overlap with each other.

rwp_left_guard_band_width[i] specifies the width of the guard band on the left side of the *i*-th packed region in relative region-wise packed picture sample units. When **ChromaFormatIdc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **rwp_left_guard_band_width[i]** shall correspond to an even number of luma samples within the cropped decoded picture.

rwp_right_guard_band_width[i] specifies the width of the guard band on the right side of the *i*-th packed region in relative region-wise packed picture sample units. When **ChromaFormatIdc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2

chroma format), `rwp_right_guard_band_width[i]` shall correspond to an even number of luma samples within the cropped decoded picture.

`rwp_top_guard_band_height[i]` specifies the height of the guard band above the *i*-th packed region in relative region-wise packed picture sample units. When `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format), `rwp_top_guard_band_height[i]` shall correspond to an even number of luma samples within the cropped decoded picture.

`rwp_bottom_guard_band_height[i]` specifies the height of the guard band below the *i*-th packed region in relative region-wise packed picture sample units. When `ChromaFormatIdc` is equal to 1 (4:2:0 chroma format), `rwp_bottom_guard_band_height[i]` shall correspond to an even number of luma samples within the cropped decoded picture.

When `rwp_guard_band_flag[i]` is equal to 1, `rwp_left_guard_band_width[i]`, `rwp_right_guard_band_width[i]`, `rwp_top_guard_band_height[i]`, or `rwp_bottom_guard_band_height[i]` shall be greater than 0.

The *i*-th packed region as specified by this SEI message shall not overlap with any other packed region specified by the same SEI message or any guard band specified by the same SEI message.

The guard bands associated with the *i*-th packed region, if any, as specified by this SEI message shall not overlap with any packed region specified by the same SEI message or any other guard bands specified by the same SEI message.

`rwp_guard_band_not_used_for_pred_flag[i]` equal to 0 specifies that the guard bands might or might not be used in the inter prediction process. `rwp_guard_band_not_used_for_pred_flag[i]` equal to 1 specifies that the sample values of the guard bands are not used in the inter prediction process.

NOTE 4 – When `rwp_guard_band_not_used_for_pred_flag[i]` is equal to 1, the sample values within guard bands in cropped decoded pictures could be rewritten even if the cropped decoded pictures were used as references for inter prediction of subsequent pictures to be decoded. For example, the content of a packed region could be seamlessly expanded to its guard band with decoded and re-projected samples of another packed region.

`rwp_guard_band_type[i][j]` indicates the type of the guard bands for the *i*-th packed region as follows, with *j* equal to 0, 1, 2, or 3 indicating that the semantics apply to the left, right, top, or bottom edge, respectively, of the packed region:

- `rwp_guard_band_type[i][j]` equal to 0 indicates that the content of the guard bands in relation to the content of the packed regions is unknown or unspecified or specified by other means not specified in this Specification. When `rwp_guard_band_not_used_for_pred_flag[i]` is equal to 0, `rwp_guard_band_type[i][j]` shall not be equal to 0.
- `rwp_guard_band_type[i][j]` equal to 1 indicates that the content of the guard bands suffices for interpolation of sample values at sub-pel sample fractional locations within the packed region and less than one sample outside of the boundary of the packed region.

NOTE 5 – `rwp_guard_band_type[i][j]` equal to 1 could be used when the boundary samples of a packed region have been copied horizontally or vertically to the guard band.

- `rwp_guard_band_type[i][j]` equal to 2 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at quality that gradually changes from the picture quality of the packed region to that of the spherically adjacent packed region.
- `rwp_guard_band_type[i][j]` equal to 3 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at a similar picture quality as within the packed region.
- `rwp_guard_band_type[i][j]` values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of `rwp_guard_band_type[i][j]` when the value is greater than 3 as equivalent to the value 0.

`rwp_guard_band_reserved_zero_3bits[i]` shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `rwp_guard_band_reserved_zero_3bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `rwp_guard_band_reserved_zero_3bits[i]`.

The variables `NumPackedRegions`, `PackedRegionLeft[n]`, `PackedRegionTop[n]`, `PackedRegionWidth[n]`, `PackedRegionHeight[n]`, `ProjRegionLeft[n]`, `ProjRegionTop[n]`, `ProjRegionWidth[n]`, `ProjRegionHeight[n]`, and `TransformType[n]` are derived as follows:

- For *n* in the range of 0 to `rwp_num_packed_regions` – 1, inclusive, the following applies:
 - `PackedRegionLeft[n]` is set equal to `rwp_packed_region_left[n]`.
 - `PackedRegionTop[n]` is set equal to `rwp_packed_region_top[n]`.
 - `PackedRegionWidth[n]` is set equal to `rwp_packed_region_width[n]`.
 - `PackedRegionHeight[n]` is set equal to `rwp_packed_region_height[n]`.

- ProjRegionLeft[n] is set equal to rwp_proj_region_left[n].
- ProjRegionTop[n] is set equal to rwp_proj_region_top[n].
- ProjRegionWidth[n] is set equal to rwp_proj_region_width[n].
- ProjRegionHeight[n] is set equal to rwp_proj_region_height[n].
- TransformType[n] is set equal to rwp_transform_type[n].
- If rwp_constituent_picture_matching_flag is equal to 0, the following applies:
 - NumPackedRegions is set equal to rwp_num_packed_regions.
- Otherwise (rwp_constituent_picture_matching_flag is equal to 1), the following applies:
 - NumPackedRegions is set equal to 2 * rwp_num_packed_regions.
 - When TopBottomFlag is equal to 1, the following applies:
 - projLeftOffset and packedLeftOffset are both set equal to 0.
 - projTopOffset is set equal to $\text{rwp_proj_picture_height} / 2$ and packedTopOffset is set equal to $\text{rwp_packed_picture_height} / 2$.
 - When SideBySideFlag is equal to 1, the following applies:
 - projLeftOffset is set equal to $\text{rwp_proj_picture_width} / 2$ and packedLeftOffset is set equal to $\text{rwp_packed_picture_width} / 2$.
 - projTopOffset and packedTopOffset are both set equal to 0.
- For n in the range of NumPackedRegions / 2 to NumPackedRegions – 1, inclusive, the following applies:
 - nIdx is set equal to $n - \text{NumPackedRegions} / 2$.
 - PackedRegionLeft[n] is set equal to $\text{rwp_packed_region_left}[\text{nIdx}] + \text{packedLeftOffset}$.
 - PackedRegionTop[n] is set equal to $\text{rwp_packed_region_top}[\text{nIdx}] + \text{packedTopOffset}$.
 - PackedRegionWidth[n] is set equal to $\text{rwp_packed_region_width}[\text{nIdx}]$.
 - PackedRegionHeight[n] is set equal to $\text{rwp_packed_region_height}[\text{nIdx}]$.
 - ProjRegionLeft[n] is set equal to $\text{rwp_proj_region_left}[\text{nIdx}] + \text{projLeftOffset}$.
 - ProjRegionTop[n] is set equal to $\text{rwp_proj_region_top}[\text{nIdx}] + \text{projTopOffset}$.
 - ProjRegionWidth[n] is set equal to $\text{rwp_proj_region_width}[\text{nIdx}]$.
 - ProjRegionHeight[n] is set equal to $\text{rwp_proj_region_height}[\text{nIdx}]$.
 - TransformType[n] is set equal to $\text{rwp_transform_type}[\text{nIdx}]$.

For each value of n in the range of 0 to NumPackedRegions – 1, inclusive, the values of ProjRegionWidth[n], ProjRegionHeight[n], ProjRegionTop[n], and ProjRegionLeft[n] are constrained as follows:

- ProjRegionWidth[n] shall be in the range of 1 to rwp_proj_picture_width, inclusive.
- ProjRegionHeight[n] shall be in the range of 1 to rwp_proj_picture_height, inclusive.
- ProjRegionLeft[n] shall be in the range of 0 to rwp_proj_picture_width – 1, inclusive.
- ProjRegionTop[n] shall be in the range of 0 to rwp_proj_picture_height – 1, inclusive.
- If ProjRegionTop[n] is less than $\text{rwp_proj_picture_height} / \text{VerDiv1}$, the sum of ProjRegionTop[n] and ProjRegionHeight[n] shall be less than or equal to $\text{rwp_proj_picture_height} / \text{VerDiv1}$. Otherwise, the sum of ProjRegionTop[n] and ProjRegionHeight[n] shall be less than or equal to $\text{rwp_proj_picture_height} / \text{VerDiv1} * 2$.

For each value of n in the range of 0 to NumPackedRegions – 1, inclusive, the values of PackedRegionWidth[n], PackedRegionHeight[n], PackedRegionTop[n], and PackedRegionLeft[n] are constrained as follows:

- PackedRegionWidth[n] shall be in the range of 1 to rwp_packed_picture_width, inclusive.
- ProjRegionHeight[n] shall be in the range of 1 to rwp_packed_picture_height, inclusive.
- PackedRegionLeft[n] shall be in the range of 0 to rwp_packed_picture_width – 1, inclusive.

- PackedRegionTop[n] shall be in the range of 0 to rwp_packed_picture_height – 1, inclusive.
- If PackedRegionLeft[n] is less than rwp_packed_picture_width / HorDiv1, the sum of PackedRegionLeft[n] and PackedRegionWidth[n] shall be less than or equal to rwp_packed_picture_width / HorDiv1. Otherwise, the sum of PackedRegionLeft[n] and PackedRegionWidth[n] shall be less than or equal to rwp_packed_picture_width / HorDiv1 * 2.
- If PackedRegionTop[n] is less than rwp_packed_picture_height / VerDiv1, the sum of PackedRegionTop[n] and PackedRegionHeight[n] shall be less than or equal to rwp_packed_picture_height / VerDiv1. Otherwise, the sum of PackedRegionTop[n] and PackedRegionHeight[n] shall be less than or equal to rwp_packed_picture_height / VerDiv1 * 2.
- When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), PackedRegionLeft[n] shall correspond to an even horizontal coordinate value of luma sample units, and PackedRegionWidth[n] shall correspond to an even number of luma samples, both within the decoded picture.
- When ChromaFormatIdc is equal to 1 (4:2:0 chroma format), PackedRegionTop[n] shall correspond to an even vertical coordinate value of luma sample units, and ProjRegionHeight[n] shall correspond to an even number of luma samples, both within the decoded picture.

8.15.6 Omnidirectional viewport SEI message

8.15.6.1 Omnidirectional viewport SEI message syntax

omni_viewport(payloadSize) {	Descriptor
omni_viewport_id	u(10)
omni_viewport_cancel_flag	u(1)
if(!omni_viewport_cancel_flag) {	
omni_viewport_persistence_flag	u(1)
omni_viewport_cnt_minus1	u(4)
for(i = 0; i <= omni_viewport_cnt_minus1; i++) {	
omni_viewport_azimuth_centre[i]	i(32)
omni_viewport_elevation_centre[i]	i(32)
omni_viewport_tilt_centre[i]	i(32)
omni_viewport_hor_range[i]	u(32)
omni_viewport_ver_range[i]	u(32)
}	
}	
}	

8.15.6.2 Omnidirectional viewport SEI message semantics

The omnidirectional viewport SEI message specifies the coordinates of one or more regions of spherical-coordinate geometry, bounded by four great circles, corresponding to viewports recommended for display when the user does not have control of the viewing orientation or has released control of the viewing orientation.

When an effectively applicable frame packing arrangement SEI message, as specified in clause 8.15.2.2 or clause 8.15.3.2, that applies to the picture is present, the information indicated by the omnidirectional viewport SEI message applies to both views.

omni_viewport_id contains an identifying number that may be used to identify the purpose of the one or more recommended viewport regions.

omni_viewport_id equal to 0 indicates that the recommended viewports are per "director's cut", i.e., a viewport suggested according to the creative intent of the content author or content provider. omni_viewport_id equal to 1 indicates that the recommended viewports are selected based on measurements of viewing statistics.

Values of omni_viewport_id from 2 to 511, inclusive, may be used as determined by the application. Values of omni_viewport_id from 512 to 1023 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of omni_viewport_id in the range of 512 to 1023, inclusive, shall ignore it.

omni_viewport_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous omnidirectional viewport SEI message in output order. **omni_viewport_cancel_flag** equal to 0 indicates that omnidirectional viewport information follows.

omni_viewport_persistence_flag specifies the persistence of the omnidirectional viewport SEI message for the current layer.

omni_viewport_persistence_flag equal to 0 specifies that the omnidirectional viewport SEI message applies to the current decoded picture only.

omni_viewport_persistence_flag equal to 1 specifies that the omnidirectional viewport 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 omnidirectional viewport SEI message is output that follows the current picture in output order.

When no omnidirectional video projection is indicated to apply to a picture, e.g., by an equirectangular projection SEI message with **erp_cancel_flag** equal to 0 or a generalized cubemap projection SEI message with **gcmap_cancel_flag** equal to 0 being present in the CLVS that applies to the current picture, an omnidirectional viewport SEI message with **omni_viewport_cancel_flag** equal to 0 shall not be present in the CLVS that applies to the picture. Decoders shall ignore omnidirectional viewport SEI messages with **omni_viewport_cancel_flag** equal to 0 for pictures to which no omnidirectional video projection is indicated to apply.

omni_viewport_cnt_minus1 plus 1 specifies the number of recommended viewport regions that are indicated by the SEI message.

When **omni_viewport_cnt_minus1** is greater than 0 and there is no information provided by external means not specified in this Specification on which recommended viewport is suggested to be displayed, the following applies:

- When **omni_viewport_id** is equal to 0 or 1, the 0-th recommended viewport is suggested to be displayed when the user does not have control of the viewing orientation or has released control of the viewing orientation.
- When **omni_viewport_id** is equal to 0, between any two recommended viewports per director's cut, the *i*-th recommended viewport has higher priority than the *j*-th recommended viewport for any values of *i* and *j* when *i* is less than *j*. The 0-th recommended viewport per director's cut has the highest priority.
- When **omni_viewport_id** is equal to 1, between any two recommended viewports, the *i*-th recommended viewport has higher popularity, among some selection of candidate viewports, than the *j*-th recommended viewport for any values of *i* and *j* when *i* is less than *j*. The 0-th most-viewed recommended viewport has the highest popularity. The selection of the candidate viewports is outside the scope of this Specification.

omni_viewport_azimuth_centre[i] and **omni_viewport_elevation_centre[i]** indicate the centre of the *i*-th recommended viewport region, in units of 2^{-16} degrees relative to the global coordinate axes. The value of **omni_viewport_azimuth_centre[i]** shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. The value of **omni_viewport_elevation_centre[i]** shall be in the range of $-90 * 2^{16}$ (i.e., -5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive.

omni_viewport_tilt_centre[i] indicates the tilt angle of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of **omni_viewport_tilt_centre[i]** shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive.

omni_viewport_hor_range[i] indicates the azimuth range of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of **omni_viewport_hor_range[i]** shall be in the range of 1 to $360 * 2^{16}$ (i.e., 23 592 960), inclusive.

omni_viewport_ver_range[i] indicates the elevation range of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of **omni_viewport_ver_range[i]** shall be in the range of 1 to $180 * 2^{16}$ (i.e., 11 796 480), inclusive.

8.16 Frame-field information SEI message

8.16.1 Frame-field information SEI message syntax

frame_field_info(payloadSize) {	Descriptor
ffi_field_pic_flag	u(1)

if(ffi_field_pic_flag) {	
ffi_bottom_field_flag	u(1)
ffi_pairing_indicated_flag	u(1)
if(ffi_pairing_indicated_flag)	
ffi_paired_with_next_field_flag	u(1)
} else {	
ffi_display_fields_from_frame_flag	u(1)
if(ffi_display_fields_from_frame_flag)	
ffi_top_field_first_flag	u(1)
ffi_display_elemental_periods_minus1	u(8)
}	
ffi_source_scan_type	u(2)
ffi_duplicate_flag	u(1)
}	

8.16.2 Frame-field information SEI message semantics

The frame-field information SEI message may be used to indicate how the associated picture should be displayed (although this is merely a suggestion rather than a prescription, as the display process is outside the scope of this Specification), the source scan type of the associated picture, and whether the associated picture is a duplicate of a previous picture, in output order, of the same layer.

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

- A fixed picture rate indicator associated with a temporal sublayer, denoted herein by FixedPicRateWithinCvsFlag, such that value 1 indicates that the temporal distance between the display times of consecutive pictures in output order is constrained and value 0 indicates no such constraint.
- A display elemental period indicator, denoted herein by DisplayElementalPeriods, that indicates the number of elemental picture period intervals that the current coded picture occupies for the display model.

ffi_field_pic_flag equal to 1 indicates that the display model considers the current picture as a field, and **ffi_field_pic_flag** equal to 0 indicates that the display model considers the current picture as a frame.

ffi_bottom_field_flag equal to 1, when present, indicates that the current picture is a bottom field (i.e., that the parity of the current picture is bottom). **ffi_bottom_field_flag** equal to 0 indicates that the current picture is a top field (i.e., that the parity of the current picture is top). The two parities, bottom and top, are considered as opposite parities.

ffi_pairing_indicated_flag equal to 1, when present, indicates that the current picture is considered paired with the next picture in output order or with the previous picture in output order as the two fields of a frame. **ffi_pairing_indicated_flag** equal to 0, when present, indicates that a pairing of the current picture with another picture to form a frame is not expressed.

ffi_paired_with_next_field_flag equal to 1, when present, indicates that the current picture is considered paired with the next picture as the two fields of a frame. **ffi_paired_with_next_field_flag** equal to 0, when present, indicates that the current picture is considered paired with the previous picture as the two fields of a frame.

When **ffi_paired_with_next_field_flag** is present, the following constraints shall apply

- If **ffi_paired_with_next_field_flag** is equal to 0, there shall be at least one picture in the CLVS that precedes the current picture in output order and the picture that precedes the current picture in output order shall have the opposite parity and **ffi_pairing_indicated_flag** equal to 1 and the value of **ffi_paired_with_next_field_flag** for that preceding picture in output order shall be equal to 1.
- Otherwise, there shall be at least one picture in the CLVS that follows the current picture in output order and the picture that follows the current picture in output order shall have the opposite parity and **ffi_pairing_indicated_flag** equal to 1 and the value of **ffi_paired_with_next_field_flag** for that following picture in output order shall be equal to 0.

ffi_display_fields_from_frame_flag equal to 1, when present, indicates that the display model operates by sequentially displaying the individual fields of the frame with alternating parity. **ffi_display_fields_from_frame_flag** equal to 0, when present, indicates that the display model operates by displaying the current picture as a complete frame.

ffi_top_field_first_flag equal to 1, when present, indicates that the first field of the frame that is displayed by the display model is the top field. **ffi_top_field_first_flag** equal to 0, when present, indicates that the first field of the frame that is displayed by the display model is the bottom field.

ffi_display_elemental_periods_minus1 plus 1, when present, indicates the number of elemental picture period intervals that the current coded picture or field occupies for the display model. The value of **ffi_display_elemental_periods_minus1** shall be equal to **DisplayElementalPeriods** – 1.

The interpretation of combinations of **ffi_field_pic_flag**, **FixedPicRateWithinCvsFlag**, **ffi_bottom_field_flag**, **ffi_display_fields_from_frame_flag**, **ffi_top_field_first_flag**, and **ffi_display_elemental_periods_minus1** (through **DisplayElementalPeriods**) is specified in Table 14, in which syntax elements that are not present are indicated by "-". Combinations of syntax elements that are not listed in Table 14 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification.

NOTE 1 – When **FixedPicRateWithinCvsFlag** is equal to 1, the indicated display times are constrained to account for time duration for a display model that follows the display patterns indicated by the values of the syntax elements of the frame-field information SEI message (although the display process is outside the scope of this Specification). Although the video decoder model might be specified to only output the entire cropped decoded picture, the modelled display behaviour sometimes includes other steps, such as the repeated display of a frame for multiple time intervals when **ffi_display_fields_from_frame_flag** is equal to 0 or the sequential display of the individual fields of a frame when **ffi_display_fields_from_frame_flag** is equal to 1.

NOTE 2 – Frame doubling can be used to facilitate the display, for example, of 25 Hz progressive-scan video on a 50 Hz progressive-scan display or 30 Hz progressive-scan video on a 60 Hz progressive-scan display. Using frame doubling and frame tripling in alternating combination on every other frame could be used to facilitate the display of 24 Hz progressive-scan video on a 60 Hz progressive-scan display.

Table 14 – Interpretation of frame-field information syntax elements

ffi_field_pic_flag	FixedPicRateWithinCvsFlag	ffi_bottom_field_flag	ffi_display_fields_from_frame_flag	ffi_top_field_first_flag	DisplayElementalPeriods		Indicated display of the picture by the display model
0	0	-	0	-	1		(progressive) Frame
		-	1	0	2		Bottom field, top field, in that order
		-	1	1	2		Top field, bottom field, in that order
		-	1	0	3		Bottom field, top field, bottom field repeated, in that order
		-	1	1	3		Top field, bottom field, top field repeated, in that order
		-	0	-	n		(progressive) Frame displayed for n elemental periods of time
	1	-	1	0	2		Bottom field, top field, in that order, each displayed for 1 elemental period of time
		-	1	1	2		Top field, bottom field, in that order, each displayed for 1 elemental period of time
		-	1	0	3		Bottom field, top field, bottom field repeated, in that order, each displayed for 1 elemental period of time
		-	1	1	3		Top field, bottom field, top field repeated, in that order, each displayed for 1 elemental period of time
		-	0	-	n		(progressive) Frame displayed for n elemental periods of time
		-	1	0	2		Bottom field, top field, in that order, each displayed for 1 elemental period of time

Table 14 – Interpretation of frame-field information syntax elements

						Indicated display of the picture by the display model
ffi_field_pic_flag	FixedPicRateWithinCvsFlag	ffi_bottom_field_flag	ffi_display_fields_from_frame_flag	ffi_top_field_first_flag	DisplayElementalPeriods	
		–	1	1	3	Top field, bottom field, top field repeated, in that order, each displayed for 1 elemental period of time
1	0	0	–	–	1	Top field
		1	–	–	1	Bottom field
	1	0	–	–	1	Top field displayed for 1 elemental period of time
		1	–	–	–	Bottom field displayed for 1 elemental period of time

ffi_source_scan_type equal to 1 indicates that the source scan type of the associated picture should be interpreted as progressive. **ffi_source_scan_type** equal to 0 indicates that the source scan type of the associated picture should be interpreted as interlaced. **ffi_source_scan_type** equal to 2 indicates that the source scan type of the associated picture is unknown or unspecified or specified by other means not specified in this Specification. **ffi_source_scan_type** equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall interpret the value 3 for **ffi_source_scan_type** as equivalent to the value 2.

ffi_duplicate_flag equal to 1 indicates that the current picture is indicated to be a duplicate of a previous picture in output order. **ffi_duplicate_flag** equal to 0 indicates that the current picture is not indicated to be a duplicate of a previous picture in output order.

NOTE 3 – The **ffi_duplicate_flag** could be used to mark coded pictures known to have originated from a repetition process such as "3:2 pull-down" or other such duplication and picture rate interpolation methods. This flag would commonly be used when a video feed is encoded as a field sequence in a "transport pass-through" fashion, with known duplicate pictures tagged by setting **ffi_duplicate_flag** equal to 1.

NOTE 4 – When **ffi_field_pic_flag** is equal to 1 and **ffi_duplicate_flag** is equal to 1, this could be interpreted as an indication that the AU contains a field that duplicates the content of the previous field in output order with the same parity as the current field.

8.17 Sample aspect ratio information SEI message

8.17.1 Sample aspect ratio information SEI message syntax

sample_aspect_ratio_info(payloadSize) {	Descriptor
sari_cancel_flag	u(1)
if(!sari_cancel_flag) {	
sari_persistence_flag	u(1)
sari_aspect_ratio_idc	u(8)

if(sari_aspect_ratio_idc == 255) {	
sari_sar_width	u(16)
sari_sar_height	u(16)
}	
}	
}	

8.17.2 Sample aspect ratio information SEI message semantics

The SARI SEI message provides information about the sample aspect ratio of the samples of the associated decoded pictures. When `vui_aspect_ratio_constant_flag` is equal to 1, there shall be no SARI SEI messages present in the CLVS.

sari_cancel_flag equal to 1 indicates that the SARI SEI message cancels the persistence of any previous SARI SEI messages in output order that applies to the current layer. `sari_cancel_flag` equal to 0 indicates that SARI follows.

sari_persistence_flag specifies the persistence of the SARI SEI message for the current layer.

`sari_persistence_flag` equal to 0 specifies that the SARI applies to the current decoded picture only.

`sar_persistence_flag` equal to 1 specifies that the SARI 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 SARI SEI message is output that follows the current picture in output order.

sari_aspect_ratio_idc, when not equal to 255, indicates the sample aspect ratio of the luma samples of the decoded output picture, with the same semantics as specified for the `SampleAspectRatio` parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When the `sari_aspect_ratio_idc` syntax element is not present, the value of `sari_aspect_ratio_idc` is inferred to be equal to 0. Values of `sari_aspect_ratio_idc` that are specified as reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret values of `sari_aspect_ratio_idc` that are reserved for future use in Rec. ITU-T H.273 | ISO/IEC 23091-2 as equivalent to the value 0.

sari_sar_width, when present, indicates the horizontal size of the sample aspect ratio (in an arbitrary unit).

sari_sar_height, when present, indicates the vertical size of the sample aspect ratio (in the same arbitrary unit as `sari_sar_width`).

When present, `sari_sar_width` and `sari_sar_height` shall be relatively prime or equal to 0. When `sari_aspect_ratio_idc` is equal to 0 or `sari_sar_width` is equal to 0 or `sari_sar_height` is equal to 0, the sample aspect ratio is unknown or unspecified or specified by other means not specified in this Specification.

8.18 Annotated regions SEI message

8.18.1 Annotated regions SEI message syntax

annotated_regions(payloadSize) {	Descriptor
ar_cancel_flag	u(1)
if(!ar_cancel_flag) {	
ar_not_optimized_for_viewing_flag	u(1)
ar_true_motion_flag	u(1)
ar_occluded_object_flag	u(1)
ar_partial_object_flag_present_flag	u(1)
ar_object_label_present_flag	u(1)
ar_object_confidence_info_present_flag	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence_length_minus1	u(4)

if(ar_object_label_present_flag) {	
ar_object_label_language_present_flag	u(1)
if(ar_object_label_language_present_flag) {	
while(!byte_aligned())	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_object_label_language	st(v)
}	
ar_num_label_updates	ue(v)
for(i = 0; i < ar_num_label_updates; i++) {	
ar_label_idx[i]	ue(v)
ar_label_cancel_flag	u(1)
LabelAssigned[ar_label_idx[i]] = !ar_label_cancel_flag	
if(!ar_label_cancel_flag) {	
while(!byte_aligned())	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_label[ar_label_idx[i]]	st(v)
}	
}	
}	
ar_num_object_updates	ue(v)
for(i = 0; i < ar_num_object_updates; i++) {	
ar_object_idx[i]	ue(v)
ar_object_cancel_flag	u(1)
ObjectTracked[ar_object_idx[i]] = !ar_object_cancel_flag	
if(!ar_object_cancel_flag) {	
if(ar_object_label_present_flag) {	
ar_object_label_update_flag	u(1)
if(ar_object_label_update_flag)	
ar_object_label_idx[ar_object_idx[i]]	ue(v)
}	
ar_bounding_box_update_flag	u(1)
if(ar_bounding_box_update_flag) {	
ar_bounding_box_cancel_flag	u(1)
ObjectBoundingBoxAvail[ar_object_idx[i]] = !ar_bounding_box_cancel_flag	
if(!ar_bounding_box_cancel_flag) {	

ar_bounding_box_top [ar_object_idx[i]]	u(16)
ar_bounding_box_left [ar_object_idx[i]]	u(16)
ar_bounding_box_width [ar_object_idx[i]]	u(16)
ar_bounding_box_height [ar_object_idx[i]]	u(16)
if(ar_partial_object_flag_present_flag)	
ar_partial_object_flag [ar_object_idx[i]]	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence [ar_object_idx[i]]	u(v)
}	
}	
}	
}	
}	

8.18.2 Annotated regions SEI message semantics

The annotated regions SEI message carries parameters that identify annotated regions using bounding boxes representing the size and location of identified objects.

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

- The width and height of a cropped decoded picture in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- A conformance cropping window left offset, ConfWinLeftOffset
- A conformance cropping window top offset, ConfWinTopOffset
- A chroma format indicator, denoted herein by ChromaFormatIdc, as described in clause 7.3.

NOTE – Although some variables used in this clause are related to picture cropping, the coordinates used in these semantics are relative to the entire decoded picture prior to cropping.

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

ar_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous annotated regions SEI message that is associated with one or more layers to which the annotated regions SEI message applies. **ar_cancel_flag** equal to 0 indicates that annotated regions information follows.

When **ar_cancel_flag** equal to 1 or a new CVS of the current layer begins, the variables LabelAssigned[i], ObjectTracked[i], and ObjectBoundingBoxAvail are set equal to 0 for i in the range of 0 to 255, inclusive.

ar_not_optimized_for_viewing_flag equal to 1 indicates that the decoded pictures that the annotated regions SEI message applies to are not optimized for user viewing, but rather are optimized for some other purpose such as algorithmic object classification performance. **ar_not_optimized_for_viewing_flag** equal to 0 indicates that the decoded pictures that the annotated regions SEI message applies to may or may not be optimized for user viewing.

ar_true_motion_flag equal to 1 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to was selected with a goal of accurately representing object motion for objects in the annotated regions. **ar_true_motion_flag** equal to 0 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to may or may not be selected with a goal of accurately representing object motion for objects in the annotated regions.

ar_occluded_object_flag equal to 1 indicates that the **ar_bounding_box_top**[ar_object_idx[i]], **ar_bounding_box_left**[ar_object_idx[i]], **ar_bounding_box_width**[ar_object_idx[i]], and **ar_bounding_box_height**[ar_object_idx[i]] syntax elements represent the size and location of an object or a portion of an object that may not be visible or may be only partially visible within the cropped decoded picture. **ar_occluded_object_flag** equal to 0 indicates that the **ar_bounding_box_top**[ar_object_idx[i]], **ar_bounding_box_left**[ar_object_idx[i]], **ar_bounding_box_width**[ar_object_idx[i]], and **ar_bounding_box_height**[ar_object_idx[i]] syntax elements represent the size and location of an object that is entirely visible within the cropped decoded picture. It is a requirement of bitstream conformance that the value of **ar_occluded_object_flag** shall be the same for all annotated_regions() syntax structures within a CVS.

ar_partial_object_flag_present_flag equal to 1 indicates that ar_partial_object_flag[ar_object_idx[i]] syntax elements are present. ar_partial_object_flag_present_flag equal to 0 indicates that ar_partial_object_flag[ar_object_idx[i]] syntax elements are not present. It is a requirement of bitstream conformance that the value of ar_partial_object_flag_present_flag shall be the same for all annotated_regions() syntax structures within a CVS.

ar_object_label_present_flag equal to 1 indicates that label information corresponding to objects in the annotated regions is present. ar_object_label_present_flag equal to 0 indicates that label information corresponding to the objects in the annotated regions is not present.

ar_object_confidence_info_present_flag equal to 1 indicates that ar_object_confidence[ar_object_idx[i]] syntax elements are present. ar_object_confidence_info_present_flag equal to 0 indicates that ar_object_confidence[ar_object_idx[i]] syntax elements are not present. It is a requirement of bitstream conformance that the value of ar_object_confidence_present_flag shall be the same for all annotated_regions() syntax structures within a CVS.

ar_object_confidence_length_minus1 plus 1 specifies the length, in bits, of the ar_object_confidence[ar_object_idx[i]] syntax elements. It is a requirement of bitstream conformance that the value of ar_object_confidence_length_minus1 shall be the same for all annotated_regions() syntax structures within a CVS.

ar_object_label_language_present_flag equal to 1 indicates that the ar_object_label_language syntax element is present. ar_object_label_language_present_flag equal to 0 indicates that the ar_object_label_language syntax element is not present.

ar_bit_equal_to_zero shall be equal to zero.

ar_object_label_language contains a language tag as specified by IETF RFC 5646 followed by a null termination byte equal to 0x00. The length of the ar_object_label_language syntax element shall be less than or equal to 255 bytes, not including the null termination byte. When not present, the language of the label is unspecified.

ar_num_label_updates indicates the total number of labels associated with the annotated regions that are signalled. The value of ar_num_label_updates shall be in the range of 0 to 255, inclusive.

ar_label_idx[i] indicates the index of the signalled label. The value of ar_label_idx[i] shall be in the range of 0 to 255, inclusive.

ar_label_cancel_flag equal to 1 cancels the persistence scope of the ar_label_idx[i]-th label. ar_label_cancel_flag equal to 0 indicates that the ar_label_idx[i]-th label is assigned a signalled value.

LabelAssigned[ar_label_idx[i]] equal to 1 indicates that the ar_label_idx[i]-th label is assigned. LabelAssigned[ar_label_idx[i]] equal to 0 indicates that the ar_label_idx[i]-th label is not assigned.

ar_label[ar_label_idx[i]] specifies the contents of the ar_label_idx[i]-th label. The length of the ar_label[ar_label_idx[i]] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

ar_num_object_updates indicates the number of object updates to be signalled. ar_num_object_updates shall be in the range of 0 to 255, inclusive.

ar_object_idx[i] is the index of the object parameters to be signalled. ar_object_idx[i] shall be in the range of 0 to 255, inclusive.

ar_object_cancel_flag equal to 1 cancels the persistence scope of the ar_object_idx[i]-th object. ar_object_cancel_flag equal to 0 indicates that parameters associated with the ar_object_idx[i]-th object are signalled.

ObjectTracked[ar_object_idx[i]] equal to 1 indicates that the ar_object_idx[i]-th object is tracked. ObjectTracked[ar_object_idx[i]] equal to 0 indicates that the ar_object_idx[i]-th object is not tracked.

ar_object_label_update_flag equal to 1 indicates that an object label is signalled. ar_object_label_update_flag equal to 0 indicates that an object label is not signalled.

ar_object_label_idx[ar_object_idx[i]] indicates the index of the label corresponding to the ar_object_idx[i]-th object. When ar_object_label_idx[ar_object_idx[i]] is not present, its value is inferred from a previous annotated regions SEI message in output order in the same CVS, if any. The value of ar_object_label_idx[ar_object_idx[i]] shall be in the range of 0 to 255, inclusive.

ar_bounding_box_update_flag equal to 1 indicates that object bounding box parameters are signalled. ar_bounding_box_update_flag equal to 0 indicates that object bounding box parameters are not signalled.

ar_bounding_box_cancel_flag equal to 1 cancels the persistence scope of the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]], ar_bounding_box_height[ar_object_idx[i]],

`ar_partial_object_flag[ar_object_idx[i]]`, and `ar_object_confidence[ar_object_idx[i]]`.
`ar_bounding_box_cancel_flag` equal to 0 indicates that `ar_bounding_box_top[ar_object_idx[i]]`,
`ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]`
`ar_bounding_box_height[ar_object_idx[i]]` and `ar_partial_object_flag[ar_object_idx[i]]`, and
`ar_object_confidence[ar_object_idx[i]]` syntax elements are signalled.

`ObjectBoundingBoxAvail[ar_object_idx[i]]` equal to 1 indicates that the bounding box information of the `ar_object_idx[i]`-th object is signalled. `ObjectBoundingBoxAvail[ar_object_idx[i]]` equal to 0 indicates that the bounding box information of the `ar_object_idx[i]`-th object is not signalled.

`ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`,
`ar_bounding_box_width[ar_object_idx[i]]`, and `ar_bounding_box_height[ar_object_idx[i]]` specify the coordinates of the top-left corner and the width and height, respectively, of the bounding box of the `ar_object_idx[i]`-th object in the cropped decoded picture, relative to the conformance cropping window.

The value of `ar_bounding_box_left[ar_object_idx[i]]` shall be in the range of 0 to `CroppedWidth / SubWidthC - 1`, inclusive.

The value of `ar_bounding_box_top[ar_object_idx[i]]` shall be in the range of 0 to `CroppedHeight / SubHeightC - 1`, inclusive.

The value of `ar_bounding_box_width[ar_object_idx[i]]` shall be in the range of 0 to `CroppedWidth / SubWidthC - ar_bounding_box_left[ar_object_idx[i]]`, inclusive.

The value of `ar_bounding_box_height[ar_object_idx[i]]` shall be in the range of 0 to `CroppedHeight / SubHeightC - ar_bounding_box_top[ar_object_idx[i]]`, inclusive.

The identified object rectangle contains the luma samples with horizontal picture coordinates from $\text{SubWidthC} * (\text{ConfWinLeftOffset} + \text{ar_bounding_box_left}[\text{ar_object_idx}[i]])$ to $\text{SubWidthC} * (\text{ConfWinLeftOffset} + \text{ar_bounding_box_left}[\text{ar_object_idx}[i]] + \text{ar_bounding_box_width}[\text{ar_object_idx}[i]])$ - 1, inclusive, and vertical picture coordinates from $\text{SubHeightC} * (\text{ConfWinTopOffset} + \text{ar_bounding_box_top}[\text{ar_object_idx}[i]])$ to $\text{SubHeightC} * (\text{ConfWinTopOffset} + \text{ar_bounding_box_top}[\text{ar_object_idx}[i]] + \text{ar_bounding_box_height}[\text{ar_object_idx}[i]])$ - 1, inclusive.

When `ChromaFormatIdc` is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates $(x / \text{SubWidthC}, y / \text{SubHeightC})$, where (x, y) are the picture coordinates of the specified luma samples.

The values of `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` persist in output order within the CVS for each value of `ar_object_idx[i]`. When not present, the values of `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` or `ar_bounding_box_height[ar_object_idx[i]]` are inferred from a previous annotated regions SEI message in output order in the CVS, if any.

`ar_partial_object_flag[ar_object_idx[i]]` equal to 1 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object that is only partially visible within the cropped decoded picture. `ar_partial_object_flag[ar_object_idx[i]]` equal to 0 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object that may or may not be only partially visible within the cropped decoded picture. When not present, the value of `ar_partial_object_flag[ar_object_idx[i]]` is inferred from a previous annotated regions SEI message in output order in the CVS, if any.

`ar_object_confidence[ar_object_idx[i]]` indicates the degree of confidence associated with the `ar_object_idx[i]`-th object, in units of $2^{-(\text{ar_object_confidence_length_minus1} + 1)}$, such that a higher value of `ar_object_confidence[ar_object_idx[i]]` indicates a higher degree of confidence. The length of the `ar_object_confidence[ar_object_idx[i]]` syntax element is `ar_object_confidence_length_minus1 + 1` bits. When not present, the value of `ar_object_confidence[ar_object_idx[i]]` is inferred from a previous annotated regions SEI message in output order in the CVS, if any.

8.19 Scalability dimension information SEI message

8.19.1 Scalability dimension information SEI message syntax

	Descriptor
scalability_dimension_info(payloadSize) {	
sdi_max_layers_minus1	u(6)
sdi_multiview_info_flag	u(1)
sdi_auxiliary_info_flag	u(1)
if(sdi_multiview_info_flag sdi_auxiliary_info_flag) {	
if(sdi_multiview_info_flag)	
sdi_view_id_len_minus1	u(4)
for(i = 0; i <= sdi_max_layers_minus1; i++) {	
sdi_layer_id[i]	u(6)
if(sdi_multiview_info_flag)	
sdi_view_id_val[i]	u(v)
if(sdi_auxiliary_info_flag)	
sdi_aux_id[i]	u(8)
if(sdi_aux_id[i] > 0) {	
sdi_num_associated_primary_layers_minus1[i]	u(6)
for(j = 0; j <= sdi_num_associated_primary_layers_minus1[i]; j++)	
sdi_associated_primary_layer_idx[i][j]	u(6)
}	
}	
}	
}	

8.19.2 Scalability dimension information SEI message semantics

The scalability dimension information (SDI) SEI message provides the SDI for each layer in the current CVS, i.e., the CVS containing the SDI SEI message, such as 1) when there may be multiple views, the view ID of each layer; and 2) when there may be auxiliary information (such as depth or alpha) carried by one or more layers, the auxiliary ID of each layer.

When an SDI SEI message is present in any AU of a CVS, an SDI SEI message shall be present for the first AU of the CVS. All SDI SEI messages in a CVS shall have the same content.

sdi_max_layers_minus1 plus 1 indicates the maximum number of layers in the current CVS.

sdi_multiview_info_flag equal to 1 indicates that the current CVS may have multiple views and the **sdi_view_id_val[]** syntax elements are present in the SDI SEI message. **sdi_multiview_info_flag** equal to 0 indicates that the current CVS does not have multiple views and the **sdi_view_id_val[]** syntax elements are not present in the SDI SEI message.

sdi_auxiliary_info_flag equal to 1 indicates that one or more layers in the current CVS may be auxiliary layers, which carry auxiliary information, and the **sdi_aux_id[]** syntax elements are present in the SDI SEI message. **sdi_auxiliary_info_flag** equal to 0 indicates that the current CVS does not have an auxiliary layer and the **sdi_aux_id[]** syntax elements are not present in the SDI SEI message.

sdi_view_id_len_minus1 plus 1 specifies the length, in bits, of the **sdi_view_id_val[i]** syntax element.

sdi_layer_id[i] specifies the layer identifier of the i-th layer that may be present in the current CVS.

sdi_view_id_val[i] specifies the view identifier of the i-th layer in the current CVS. The length of the **sdi_view_id_val[i]** syntax element is **sdi_view_id_len_minus1 + 1** bits.

The variable NumViews, specifying the number of views in the current CVS, and the list ViewId, specifying the view identifiers of the views in the current CVS, are derived as follows:

```

NumViews = 1
if( sdi_multiview_info_flag ) {
  ViewId[ 0 ] = sdi_view_id_val[ 0 ]
  for( i = 1; i <= sdi_max_layers_minus1; i++ ) {
    newViewFlag = 1
    for( j = 0; j < i; j++ )
      if( sdi_view_id_val[ i ] == sdi_view_id_val[ j ] )
        newViewFlag = 0
    if( newViewFlag ) {
      ViewId[ NumViews ] = sdi_view_id_val[ i ]
      NumViews++
    }
  }
}
}

```

(50)

sdi_aux_id[i] equal to 0 indicates that the *i*-th layer in the current CVS does not contain auxiliary pictures. **sdi_aux_id[i]** greater than 0 indicates the type of auxiliary pictures in the *i*-th layer in the current CVS as specified in Table 15. When **sdi_auxiliary_info_flag** is equal to 0, the value of **sdi_aux_id[i]** is inferred to be equal to 0.

Table 15 – Mapping of sdi_aux_id[i] to the type of auxiliary pictures

sdi_aux_id[i]	Name	Type of auxiliary pictures
1	AUX_ALPHA	Alpha plane
2	AUX_DEPTH	Depth picture
3..127		Reserved
128..159		Unspecified
160..255		Reserved

NOTE 1 – The interpretation of auxiliary pictures associated with **sdi_aux_id[i]** in the range of 128 to 159, inclusive, is specified through means other than the **sdi_aux_id[i]** value.

sdi_aux_id[i] shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, for bitstreams conforming to this version of this Specification. Although the value of **sdi_aux_id[i]** shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, in this version of this Specification, decoders shall also allow other values of **sdi_aux_id[i]** in the range of 0 to 255, inclusive.

If **sdi_aux_id[i]** is equal to 0, the *i*-th layer is referred to as a primary layer. Otherwise, the *i*-th layer is referred to as an auxiliary layer. When **sdi_aux_id[i]** is equal to 1, the *i*-th layer is also referred to as an alpha auxiliary layer. When **sdi_aux_id[i]** is equal to 2, the *i*-th layer is also referred to as a depth auxiliary layer.

sdi_num_associated_primary_layers_minus1[i] plus 1 specifies the number of associated primary layers of *i*-th layer, which is an auxiliary layer. The value of **sdi_num_associated_primary_layers_minus1[i]** shall be less than the total number of primary layers.

sdi_associated_primary_layer_idx[i][j] specifies the layer index of the *j*-th associated primary layer of the *i*-th layer, which is an auxiliary layer. The value of **sdi_aux_id[sdi_associated_primary_layer_idx[i][j]]** shall be equal to 0.

NOTE 2 – An auxiliary layer describes a property of and applies to its associated primary layers.

8.20 Multiview acquisition information SEI message

8.20.1 Multiview acquisition information SEI message syntax

multiview_acquisition_info(payloadSize) {		Descriptor
intrinsic_param_flag		u(1)
extrinsic_param_flag		u(1)
num_views_minus1		ue(v)
if(intrinsic_param_flag) {		
intrinsic_params_equal_flag		u(1)

prec_focal_length	ue(v)
prec_principal_point	ue(v)
prec_skew_factor	ue(v)
for(i = 0; i <= intrinsic_params_equal_flag ? 0 : num_views_minus1; i++) {	
sign_focal_length_x [i]	u(1)
exponent_focal_length_x [i]	u(6)
mantissa_focal_length_x [i]	u(v)
sign_focal_length_y [i]	u(1)
exponent_focal_length_y [i]	u(6)
mantissa_focal_length_y [i]	u(v)
sign_principal_point_x [i]	u(1)
exponent_principal_point_x [i]	u(6)
mantissa_principal_point_x [i]	u(v)
sign_principal_point_y [i]	u(1)
exponent_principal_point_y [i]	u(6)
mantissa_principal_point_y [i]	u(v)
sign_skew_factor [i]	u(1)
exponent_skew_factor [i]	u(6)
mantissa_skew_factor [i]	u(v)
}	
}	
if(extrinsic_param_flag) {	
prec_rotation_param	ue(v)
prec_translation_param	ue(v)
for(i = 0; i <= num_views_minus1; i++)	
for(j = 0; j < 3; j++) { /* row */	
for(k = 0; k < 3; k++) { /* column */	
sign_r [i][j][k]	u(1)
exponent_r [i][j][k]	u(6)
mantissa_r [i][j][k]	u(v)
}	
sign_t [i][j]	u(1)
exponent_t [i][j]	u(6)
mantissa_t [i][j]	u(v)
}	
}	
}	

8.20.2 Multiview acquisition information SEI message semantics

The multiview acquisition information (MAI) SEI message specifies various parameters of the acquisition environment for the layers that may be present in the current CVS, i.e., the CVS containing the MAI SEI message. Specifically, intrinsic and extrinsic camera parameters are specified. These parameters could be used for processing the decoded views prior to rendering on a 3D display.

When an MAI SEI message is present in any AU of a CVS, an MAI SEI message shall be present for the first AU of the CVS. All MAI SEI messages in a CVS shall have the same content.

When a CVS does not contain an SDI SEI message, the CVS shall not contain an MAI SEI message.

When an AU contains both an SDI SEI message and an MAI SEI message, the SDI SEI message shall precede the MAI SEI message in decoding order.

Some of the views for which the MAI is included in an MAI SEI message may not be present in the current CVS.

In the semantics below, syntax elements and variables with index i refer to the syntax elements and variables that apply to the i -th view in the current CVS specified by the SDI SEI message, i.e., the view with view identifier equal to $\text{ViewId}[i]$.

The extrinsic camera parameters are specified according to a right-handed coordinate system, where the upper left corner of the image is the origin, i.e., the $(0, 0)$ coordinate, with the other corners of the image having non-negative coordinates. With these specifications, a 3-dimensional world point, $wP = [x\ y\ z]$ is mapped to a 2-dimensional camera point, $cP[i] = [u\ v\ 1]$, for the i -th camera according to:

$$s * cP[i] = A[i] * R^{-1}[i] * (wP - T[i]) \quad (51)$$

where $A[i]$ denotes the intrinsic camera parameter matrix, $R^{-1}[i]$ denotes the inverse of the rotation matrix $R[i]$, $T[i]$ denotes the translation vector and s (a scalar value) is an arbitrary scale factor chosen to make the third coordinate of $cP[i]$ equal to 1. The elements of $A[i]$, $R[i]$ and $T[i]$ are determined according to the syntax elements signalled in this SEI message and as specified below.

intrinsic_param_flag equal to 1 indicates the presence of intrinsic camera parameters. **intrinsic_param_flag** equal to 0 indicates the absence of intrinsic camera parameters.

extrinsic_param_flag equal to 1 indicates the presence of extrinsic camera parameters. **extrinsic_param_flag** equal to 0 indicates the absence of extrinsic camera parameters.

num_views_minus1 plus 1 specifies the number of views for which the MAI is included in the MAI SEI message. The value of **num_views_minus1** shall be equal to $\text{NumViews} - 1$.

intrinsic_params_equal_flag equal to 1 indicates that the intrinsic camera parameters are equal for all cameras and only one set of intrinsic camera parameters is present. **intrinsic_params_equal_flag** equal to 0 indicates that the intrinsic camera parameters are different for each camera and that a set of intrinsic camera parameters is present for each camera.

prec_focal_length specifies the exponent of the maximum allowable truncation error for **focal_length_x[i]** and **focal_length_y[i]** as given by $2^{-\text{prec_focal_length}}$. The value of **prec_focal_length** shall be in the range of 0 to 31, inclusive.

prec_principal_point specifies the exponent of the maximum allowable truncation error for **principal_point_x[i]** and **principal_point_y[i]** as given by $2^{-\text{prec_principal_point}}$. The value of **prec_principal_point** shall be in the range of 0 to 31, inclusive.

prec_skew_factor specifies the exponent of the maximum allowable truncation error for skew factor as given by $2^{-\text{prec_skew_factor}}$. The value of **prec_skew_factor** shall be in the range of 0 to 31, inclusive.

sign_focal_length_x[i] equal to 0 indicates that the sign of the focal length of the i -th camera in the horizontal direction is positive. **sign_focal_length_x[i]** equal to 1 indicates that the sign is negative.

exponent_focal_length_x[i] specifies the exponent part of the focal length of the i -th camera in the horizontal direction. The value of **exponent_focal_length_x[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_x[i] specifies the mantissa part of the focal length of the i -th camera in the horizontal direction. The length of the **mantissa_focal_length_x[i]** syntax element in units of bits is variable and determined as follows:

- If **exponent_focal_length_x[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_focal_length} - 30)$.
- Otherwise (**exponent_focal_length_x[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_focal_length_x}[i] + \text{prec_focal_length} - 31)$.

sign_focal_length_y[i] equal to 0 indicates that the sign of the focal length of the i -th camera in the vertical direction is positive. **sign_focal_length_y[i]** equal to 1 indicates that the sign is negative.

exponent_focal_length_y[i] specifies the exponent part of the focal length of the i -th camera in the vertical direction. The value of **exponent_focal_length_y[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_y[i] specifies the mantissa part of the focal length of the i -th camera in the vertical direction.

The length of the **mantissa_focal_length_y[i]** syntax element in units of bits is variable and determined as follows:

- If **exponent_focal_length_y[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_focal_length} - 30)$.
- Otherwise (**exponent_focal_length_y[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_focal_length_y}[i] + \text{prec_focal_length} - 31)$.

sign_principal_point_x[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the horizontal direction is positive. **sign_principal_point_x[i]** equal to 1 indicates that the sign is negative.

exponent_principal_point_x[i] specifies the exponent part of the principal point of the i-th camera in the horizontal direction. The value of **exponent_principal_point_x[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_x[i] specifies the mantissa part of the principal point of the i-th camera in the horizontal direction. The length of the **mantissa_principal_point_x[i]** syntax element in units of bits is variable and is determined as follows:

- If **exponent_principal_point_x[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_principal_point} - 30)$.
- Otherwise (**exponent_principal_point_x[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_principal_point_x}[i] + \text{prec_principal_point} - 31)$.

sign_principal_point_y[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the vertical direction is positive. **sign_principal_point_y[i]** equal to 1 indicates that the sign is negative.

exponent_principal_point_y[i] specifies the exponent part of the principal point of the i-th camera in the vertical direction. The value of **exponent_principal_point_y[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_y[i] specifies the mantissa part of the principal point of the i-th camera in the vertical direction. The length of the **mantissa_principal_point_y[i]** syntax element in units of bits is variable and is determined as follows:

- If **exponent_principal_point_y[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_principal_point} - 30)$.
- Otherwise (**exponent_principal_point_y[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_principal_point_y}[i] + \text{prec_principal_point} - 31)$.

sign_skew_factor[i] equal to 0 indicates that the sign of the skew factor of the i-th camera is positive.

sign_skew_factor[i] equal to 1 indicates that the sign is negative.

exponent_skew_factor[i] specifies the exponent part of the skew factor of the i-th camera. The value of **exponent_skew_factor[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified skew factor.

mantissa_skew_factor[i] specifies the mantissa part of the skew factor of the i-th camera. The length of the **mantissa_skew_factor[i]** syntax element in units of bits is variable and determined as follows:

- If **exponent_skew_factor[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_skew_factor} - 30)$.
- Otherwise (**exponent_skew_factor[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_skew_factor}[i] + \text{prec_skew_factor} - 31)$.

The intrinsic matrix $A[i]$ for i-th camera is represented by

$$\begin{bmatrix} \text{focalLengthX}[i] & \text{skewFactor}[i] & \text{principalPointX}[i] \\ 0 & \text{focalLengthY}[i] & \text{principalPointY}[i] \\ 0 & 0 & 1 \end{bmatrix} \quad (52)$$

prec_rotation_param specifies the exponent of the maximum allowable truncation error for $rE[i][j][k]$ (see Equation 50) as given by $2^{-\text{prec_rotation_param}}$. The value of **prec_rotation_param** shall be in the range of 0 to 31, inclusive.

prec_translation_param specifies the exponent of the maximum allowable truncation error for $tE[i][j]$ (see Equation 51) as given by $2^{-\text{prec_translation_param}}$. The value of **prec_translation_param** shall be in the range of 0 to 31, inclusive.

sign_r[i][j][k] equal to 0 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is positive. **sign_r[i][j][k]** equal to 1 indicates that the sign is negative.

exponent_r[i][j][k] specifies the exponent part of (j, k) component of the rotation matrix for the i-th camera. The value of **exponent_r[i][j][k]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified rotation matrix.

mantissa_r[i][j][k] specifies the mantissa part of (j, k) component of the rotation matrix for the i-th camera. The length of the **mantissa_r[i][j][k]** syntax element in units of bits is variable and determined as follows:

- If $\text{exponent_r}[i]$ is equal to 0, the length is $\text{Max}(0, \text{prec_rotation_param} - 30)$.
- Otherwise ($\text{exponent_r}[i]$ is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_r}[i] + \text{prec_rotation_param} - 31)$.

The rotation matrix $R[i]$ for i -th camera is represented as follows:

$$\begin{bmatrix} rE[i][0][0] & rE[i][0][1] & rE[i][0][2] \\ rE[i][1][0] & rE[i][1][1] & rE[i][1][2] \\ rE[i][2][0] & rE[i][2][1] & rE[i][2][2] \end{bmatrix} \quad (53)$$

$\text{sign_t}[i][j]$ equal to 0 indicates that the sign of the j -th component of the translation vector for the i -th camera is positive. $\text{sign_t}[i][j]$ equal to 1 indicates that the sign is negative.

$\text{exponent_t}[i][j]$ specifies the exponent part of the j -th component of the translation vector for the i -th camera. The value of $\text{exponent_t}[i][j]$ shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified translation vector.

$\text{mantissa_t}[i][j]$ specifies the mantissa part of the j -th component of the translation vector for the i -th camera. The length v of the mantissa $\text{t}[i][j]$ syntax element in units of bits is variable and is determined as follows:

- If $\text{exponent_t}[i]$ is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_translation_param} - 30)$.
- Otherwise ($0 < \text{exponent_t}[i] < 63$), the length v is set equal to $\text{Max}(0, \text{exponent_t}[i] + \text{prec_translation_param} - 31)$.

The translation vector $T[i]$ for the i -th camera is represented by:

$$\begin{bmatrix} tE[i][0] \\ tE[i][1] \\ tE[i][2] \end{bmatrix} \quad (54)$$

The association between the camera parameter variables and corresponding syntax elements is specified by Table 16. Each component of the intrinsic and rotation matrices and the translation vector is obtained from the variables specified in Table 16 as the variable x computed as follows:

- If e is in the range of 0 to 63, exclusive, x is set equal to $(-1)^s * 2^{e-31} * (1 + n \div 2^v)$.
- Otherwise (e is equal to 0), x is set equal to $(-1)^s * 2^{-(30+v)} * n$.

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

Table 16 – Association between camera parameter variables and syntax elements

x	s	e	n
focalLengthX [i]	$\text{sign_focal_length_x}[i]$	$\text{exponent_focal_length_x}[i]$	$\text{mantissa_focal_length_x}[i]$
focalLengthY [i]	$\text{sign_focal_length_y}[i]$	$\text{exponent_focal_length_y}[i]$	$\text{mantissa_focal_length_y}[i]$
principalPointX [i]	$\text{sign_principal_point_x}[i]$	$\text{exponent_principal_point_x}[i]$	$\text{mantissa_principal_point_x}[i]$
principalPointY [i]	$\text{sign_principal_point_y}[i]$	$\text{exponent_principal_point_y}[i]$	$\text{mantissa_principal_point_y}[i]$
skewFactor [i]	$\text{sign_skew_factor}[i]$	$\text{exponent_skew_factor}[i]$	$\text{mantissa_skew_factor}[i]$
rE [i][j][k]	$\text{sign_r}[i][j][k]$	$\text{exponent_r}[i][j][k]$	$\text{mantissa_r}[i][j][k]$
tE [i][j]	$\text{sign_t}[i][j]$	$\text{exponent_t}[i][j]$	$\text{mantissa_t}[i][j]$

8.21 Multiview view position SEI message

8.21.1 Multiview view position SEI message syntax

	Descriptor
multiview_view_position(payloadSize) {	
num_views_minus1	ue(v)
for(i = 0; i <= num_views_minus1; i++)	
view_position[i]	ue(v)
}	

8.21.2 Multiview view position SEI message semantics

The multiview view position SEI message specifies the relative view position along a single horizontal axis of views within a CVS.

When a multiview view position SEI message is present in any AU of a CVS, a multiview view position SEI message shall be present for the first AU of the CVS. All multiview view position SEI messages in a CVS shall have the same content.

When a CVS does not contain an SDI SEI message, the CVS shall not contain a multiview view position SEI message.

When an AU contains both an SDI SEI message and a multiview view position SEI message, the SDI SEI message shall precede the multiview view position SEI message in decoding order.

Some of the views for which the view position information is included in a multiview view position SEI message may not be present in the current CVS.

num_views_minus1 plus 1 shall be equal to NumViews derived from the SDI SEI message for the CVS.

view_position[i] indicates the order of the view with view identifier equal to ViewId[i] among all the views from left to right for the purpose of display, with the order for the left-most view being equal to 0 and the value of the order increasing by 1 for next view from left to right. The value of **view_position[i]** shall be in the range of 0 to 62, inclusive.

8.22 Depth representation information SEI message

8.22.1 Depth representation information SEI message syntax

	Descriptor
depth_representation_info(payloadSize) {	
z_near_flag	u(1)
z_far_flag	u(1)
d_min_flag	u(1)
d_max_flag	u(1)
depth_representation_type	ue(v)
if(d_min_flag d_max_flag)	
disparity_ref_view_id	ue(v)
if(z_near_flag)	
depth_rep_info_element(ZNearSign, ZNearExp, ZNearMantissa, ZNearManLen)	
if(z_far_flag)	
depth_rep_info_element(ZFarSign, ZFarExp, ZFarMantissa, ZFarManLen)	
if(d_min_flag)	
depth_rep_info_element(DMinSign, DMinExp, DMinMantissa, DMinManLen)	
if(d_max_flag)	
depth_rep_info_element(DMaxSign, DMaxExp, DMaxMantissa, DMaxManLen)	
if(depth_representation_type == 3) {	
depth_nonlinear_representation_num_minus1	ue(v)

for(i = 1; i <= depth_nonlinear_representation_num_minus1 + 1; i++)	
depth_nonlinear_representation_model [i]	ue(v)
}	
}	

8.2.2.1.1 Depth representation information element syntax

depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen) {	Descriptor
da_sign_flag	u(1)
da_exponent	u(7)
da_mantissa_len_minus1	u(5)
da_mantissa	u(v)
}	

8.2.2.2 Depth representation information SEI message semantics

The syntax elements in the depth representation information (DRI) SEI message specify various parameters for auxiliary pictures of type AUX_DEPTH for the purpose of processing decoded primary and auxiliary pictures prior to rendering on a 3D display, such as view synthesis. Specifically, depth or disparity ranges for depth pictures are specified.

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

- A bit depth for the samples of the luma component, denoted herein by BitDepth_Y.

When a CVS does not contain an SDI SEI message with sdi_aux_id[i] equal to 2 for at least one value of i, no picture in the CVS shall be associated with a DRI SEI message.

When an AU contains both an SDI SEI message with sdi_aux_id[i] equal to 2 for at least one value of i and a DRI SEI message, the SDI SEI message shall precede the DRI SEI message in decoding order.

When present, the DRI SEI message shall be associated with one or more layers that are indicated as depth auxiliary layers by an SDI SEI message. The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the DRI SEI message applies.

When present, the DRI SEI message may be included in any access unit. It is recommended that, when present, the SEI message is included for the purpose of random access in an access unit in which the coded picture with nuh_layer_id equal to targetLayerId is an IRAP picture.

The information indicated in the SEI message applies to all the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message up to but excluding the next picture, in decoding order, associated with a DRI SEI message applicable to targetLayerId or to the end of the CLVS of the nuh_layer_id equal to targetLayerId, whichever is earlier in decoding order.

z_near_flag equal to 0 specifies that the syntax elements specifying the nearest depth value are not present in the syntax structure. z_near_flag equal to 1 specifies that the syntax elements specifying the nearest depth value are present in the syntax structure.

z_far_flag equal to 0 specifies that the syntax elements specifying the farthest depth value are not present in the syntax structure. z_far_flag equal to 1 specifies that the syntax elements specifying the farthest depth value are present in the syntax structure.

d_min_flag equal to 0 specifies that the syntax elements specifying the minimum disparity value are not present in the syntax structure. d_min_flag equal to 1 specifies that the syntax elements specifying the minimum disparity value are present in the syntax structure.

d_max_flag equal to 0 specifies that the syntax elements specifying the maximum disparity value are not present in the syntax structure. d_max_flag equal to 1 specifies that the syntax elements specifying the maximum disparity value are present in the syntax structure.

depth_representation_type specifies the representation definition of decoded luma samples of auxiliary pictures as specified in Table 17. In Table 17, disparity specifies the horizontal displacement between two texture views and Z value specifies the distance from a camera. The value of depth_representation_type shall be in the range of 0 to 3, inclusive, in

bitstreams conforming to this version of this Specification. The values of 4 to 15, inclusive, for `depth_representation_type` are reserved for future use by ITU-T | ISO/IEC. Although the value of `depth_representation_type` is required to be in the range of 0 to 3, inclusive, in this version of this Specification, decoders shall also allow values of `depth_representation_type` in the range of 4 to 15, inclusive, to appear in the syntax. Decoders conforming to this version of this Specification shall ignore the bits that follow a value of `depth_representation_type` in the range of 4 to 15, inclusive, in the depth representation information SEI message.

The variable `maxVal` is set equal to $(1 \ll \text{BitDepth}_Y) - 1$.

Table 17 – Definition of `depth_representation_type`

<code>depth_representation_type</code>	Interpretation
0	Each decoded luma sample value of an auxiliary picture represents an inverse of Z value that is uniformly quantized into the range of 0 to <code>maxVal</code> , inclusive. When <code>z_far_flag</code> is equal to 1, the luma sample value equal to 0 represents the inverse of ZFar (specified below). When <code>z_near_flag</code> is equal to 1, the luma sample value equal to <code>maxVal</code> represents the inverse of ZNear (specified below).
1	Each decoded luma sample value of an auxiliary picture represents disparity that is uniformly quantized into the range of 0 to <code>maxVal</code> , inclusive. When <code>d_min_flag</code> is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When <code>d_max_flag</code> is equal to 1, the luma sample value equal to <code>maxVal</code> represents DMax (specified below).
2	Each decoded luma sample value of an auxiliary picture represents a Z value uniformly quantized into the range of 0 to <code>maxVal</code> , inclusive. When <code>z_far_flag</code> is equal to 1, the luma sample value equal to 0 corresponds to ZFar (specified below). When <code>z_near_flag</code> is equal to 1, the luma sample value equal to <code>maxVal</code> represents ZNear (specified below).
3	Each decoded luma sample value of an auxiliary picture represents a nonlinearly mapped disparity, normalized in range from 0 to <code>maxVal</code> , as specified by <code>depth_nonlinear_representation_num_minus1</code> and <code>depth_nonlinear_representation_model[i]</code> . When <code>d_min_flag</code> is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When <code>d_max_flag</code> is equal to 1, the luma sample value equal to <code>maxVal</code> represents DMax (specified below).
Other values	Reserved for future use

`disparity_ref_view_id` specifies the view identifier for which the disparity values are derived. The value of `disparity_ref_view_id` shall be in the range of 0 to 1023, inclusive.

NOTE 1 – The view identifier of the *i*-th view in the current CVS is equal to `ViewId[i]` as specified in the semantics of the SDI SEI message in clause 8.19.2.

NOTE 2 – `disparity_ref_view_id` is present only if `d_min_flag` is equal to 1 or `d_max_flag` is equal to 1 and is useful for `depth_representation_type` values equal to 1 and 3.

The variables in the *x* column of Table 18 are derived from the respective variables in the *s*, *e*, *n* and *v* columns of Table 18 as follows:

- If the value of *e* is in the range of 0 to 127, exclusive, *x* is set equal to $(-1)^s * 2^{e-31} * (1 + n \div 2^v)$.
- Otherwise (*e* is equal to 0), *x* is set equal to $(-1)^s * 2^{-(30+v)} * n$.

NOTE 3 – The above specification is similar to that found in IEC 60559:1989.

Table 18 – Association between depth parameter variables and syntax elements

<i>x</i>	<i>s</i>	<i>e</i>	<i>n</i>	<i>v</i>
ZNear	ZNearSign	ZNearExp	ZNearMantissa	ZNearManLen
ZFar	ZFarSign	ZFarExp	ZFarMantissa	ZFarManLen
DMax	DMaxSign	DMaxExp	DMaxMantissa	DMaxManLen
DMin	DMinSign	DMinExp	DMinMantissa	DMinManLen

The DMin and DMax values, when present, are specified in units of a luma sample width of the associated primary picture of the auxiliary picture of type AUX_DEPTH.

The units for the ZNear and ZFar values, when present, are identical but unspecified.

depth_nonlinear_representation_num_minus1 plus 2 specifies the number of piece-wise linear segments for mapping of depth values to a scale that is uniformly quantized in terms of disparity. The value of **depth_nonlinear_representation_num_minus1** shall be in the range of 0 to 62, inclusive.

depth_nonlinear_representation_model[i] for *i* ranging from 0 to **depth_nonlinear_representation_num_minus1** + 2, inclusive, specify the piece-wise linear segments for mapping of decoded luma sample values of an auxiliary picture to a scale that is uniformly quantized in terms of disparity. The value of **depth_nonlinear_representation_model[i]** shall be in the range of 0 to 65 535, inclusive. The values of **depth_nonlinear_representation_model[0]** and **depth_nonlinear_representation_model[depth_nonlinear_representation_num_minus1 + 2]** are both inferred to be equal to 0.

NOTE 4 – When **depth_representation_type** is equal to 3, an auxiliary picture contains non-linearly transformed depth samples. The variable **DepthLUT[i]**, as specified below, is used to transform decoded depth sample values from the non-linear representation to the linear representation, i.e., uniformly quantized disparity values. The shape of this transform is defined by means of line-segment approximation in two-dimensional linear-disparity-to-non-linear-disparity space. The first (0, 0) and the last (maxVal, maxVal) nodes of the curve are predefined. Positions of additional nodes are transmitted in form of deviations (**depth_nonlinear_representation_model[i]**) from the straight-line curve. These deviations are uniformly distributed along the whole range of 0 to maxVal, inclusive, with spacing depending on the value of **depth_nonlinear_representation_num_minus1**. The variable **DepthLUT[i]** for *i* in the range of 0 to maxVal, inclusive, is specified as follows:

```

for( k = 0; k <= depth_nonlinear_representation_num_minus1 + 1; k++ ) {
    pos1 = ( maxVal * k ) / ( depth_nonlinear_representation_num_minus1 + 2 )
    dev1 = depth_nonlinear_representation_model[ k ]
    pos2 = ( maxVal * ( k + 1 ) ) / ( depth_nonlinear_representation_num_minus1 + 2 )
    dev2 = depth_nonlinear_representation_model[ k + 1 ]
}

x1 = pos1 - dev1
y1 = pos1 + dev1
x2 = pos2 - dev2
y2 = pos2 + dev2

for( x = Max( x1, 0 ); x <= Min( x2, maxVal ); x++ )
    DepthLUT[ x ] = Clip3( 0, maxVal, Round( ( ( x - x1 ) * ( y2 - y1 ) ) ÷ ( x2 - x1 ) + y1 ) )
}

```

When **depth_representation_type** is equal to 3, **DepthLUT[dS]** for all decoded luma sample values *dS* of an auxiliary picture in the range of 0 to maxVal, inclusive, represents disparity that is uniformly quantized into the range of 0 to maxVal, inclusive.

8.22.2.1 Depth representation information element semantics

The syntax structure specifies the value of an element in the DRI SEI message.

The **depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen)** syntax structure sets the values of the **OutSign**, **OutExp**, **OutMantissa** and **OutManLen** variables that represent a floating-point value. When the syntax structure is included in another syntax structure, the variable names **OutSign**, **OutExp**, **OutMantissa** and **OutManLen** are to be interpreted as being replaced by the variable names used when the syntax structure is included.

da_sign_flag equal to 0 indicates that the sign of the floating-point value is positive. **da_sign_flag** equal to 1 indicates that the sign is negative. The variable **OutSign** is set equal to **da_sign_flag**.

da_exponent specifies the exponent of the floating-point value. The value of **da_exponent** shall be in the range of 0 to $2^7 - 2$, inclusive. The value $2^7 - 1$ is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value $2^7 - 1$ as indicating an unspecified value. The variable **OutExp** is set equal to **da_exponent**.

da_mantissa_len_minus1 plus 1 specifies the number of bits in the **da_mantissa** syntax element. The variable **OutManLen** is set equal to **da_mantissa_len_minus1** + 1.

da_mantissa specifies the mantissa of the floating-point value. The variable **OutMantissa** is set equal to **da_mantissa**.

8.23 Alpha channel information SEI message

8.23.1 Alpha channel information SEI message syntax

alpha_channel_info(payloadSize) {	Descriptor
alpha_channel_cancel_flag	u(1)
if(!alpha_channel_cancel_flag) {	

alpha_channel_use_idc	u(3)
alpha_channel_bit_depth_minus8	u(3)
alpha_transparent_value	u(v)
alpha_opaque_value	u(v)
alpha_channel_incr_flag	u(1)
alpha_channel_clip_flag	u(1)
if(alpha_channel_clip_flag)	
alpha_channel_clip_type_flag	u(1)
}	
}	

8.23.2 Alpha channel information SEI message semantics

The alpha channel information (ACI) SEI message provides information about alpha channel sample values and post-processing applied to the decoded alpha planes coded in auxiliary pictures of type AUX_ALPHA and one or more associated primary pictures.

When a CVS does not contain an SDI SEI message with `sdi_aux_id[i]` equal to 1 for at least one value of `i`, no picture in the CVS shall be associated with an ACI SEI message.

When an AU contains both an SDI SEI message with `sdi_aux_id[i]` equal to 1 for at least one value of `i` and an ACI SEI message, the SDI SEI message shall precede the ACI 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.

The following semantics apply separately to each `nuh_layer_id` `targetLayerId` among the `nuh_layer_id` values to which the ACI SEI message applies.

alpha_channel_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous ACI SEI message in output order that applies to the current layer. **alpha_channel_cancel_flag** equal to 0 indicates that ACI follows.

Let `currPic` be the picture that the ACI SEI message is associated with. The semantics of ACI SEI message persist for 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 ACI SEI message is output that follows the current picture in output order.

alpha_channel_use_idc equal to 0 indicates that for alpha blending purposes the decoded samples of the associated primary picture should be multiplied by the interpretation sample values of the decoded auxiliary picture in the display process after output from the decoding process. **alpha_channel_use_idc** equal to 1 indicates that for alpha blending purposes the decoded samples of the associated primary picture should not be multiplied by the interpretation sample values of the decoded auxiliary picture in the display process after output from the decoding process. **alpha_channel_use_idc** equal to 2 indicates that the usage of the auxiliary picture is unspecified. Values greater than 2 for **alpha_channel_use_idc** are reserved for future use by ITU-T | ISO/IEC. When not present, the value of **alpha_channel_use_idc** is inferred to be equal to 2. Decoders shall ignore alpha channel information SEI messages in which **alpha_channel_use_idc** is greater than 2.

alpha_channel_bit_depth_minus8 plus 8 specifies the bit depth of the samples of the luma sample array of the auxiliary picture. **alpha_channel_bit_depth_minus8** plus 8 shall be equal to the bit depth of the associated primary picture.

alpha_transparent_value specifies the interpretation sample value of a decoded auxiliary picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered transparent for purposes of alpha

blending. The number of bits used for the representation of the `alpha_transparent_value` syntax element is `alpha_channel_bit_depth_minus8 + 9`.

alpha_opaque_value specifies the interpretation sample value of a decoded auxiliary picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered opaque for purposes of alpha blending. The number of bits used for the representation of the `alpha_opaque_value` syntax element is `alpha_channel_bit_depth_minus8 + 9`.

A value of `alpha_opaque_value` that is equal to `alpha_transparent_value` indicates that the auxiliary coded picture is not intended for alpha blending purposes.

NOTE 1 – For alpha blending purposes, `alpha_opaque_value` can be greater than `alpha_transparent_value` or it can be less than or equal to `alpha_transparent_value`.

alpha_channel_incr_flag equal to 0 indicates that the interpretation sample value for each decoded auxiliary picture luma sample value is equal to the decoded auxiliary picture sample value for purposes of alpha blending. `alpha_channel_incr_flag` equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample value that is greater than $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be increased by one to obtain the interpretation sample value for the auxiliary picture sample and any auxiliary picture luma sample value that is less than or equal to $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be used, without alteration, as the interpretation sample value for the decoded auxiliary picture sample value.

When `alpha_transparent_value` is equal to `alpha_opaque_value` or $\text{Log}_2(\text{Abs}(\text{alpha_opaque_value} - \text{alpha_transparent_value}))$ does not have an integer value, `alpha_channel_incr_flag` shall be equal to 0.

alpha_channel_clip_flag equal to 0 indicates that no clipping operation is applied to obtain the interpretation sample values of the decoded auxiliary picture. `alpha_channel_clip_flag` equal to 1 indicates that the interpretation sample values of the decoded auxiliary picture are altered according to the clipping process described by the `alpha_channel_clip_type_flag` syntax element.

alpha_channel_clip_type_flag equal to 0 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than $(\text{alpha_opaque_value} + \text{alpha_transparent_value}) / 2$ is set equal to $\text{Max}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less or equal than $(\text{alpha_opaque_value} + \text{alpha_transparent_value}) / 2$ is set equal to $\text{Min}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ to obtain the interpretation sample value for the auxiliary picture luma sample. `alpha_channel_clip_type_flag` equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than $\text{Max}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ is set equal to $\text{Max}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less than or equal to $\text{Min}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ is set equal to $\text{Min}(\text{alpha_transparent_value}, \text{alpha_opaque_value})$ to obtain the interpretation sample value for the auxiliary picture luma sample.

When both `alpha_channel_incr_flag` and `alpha_channel_clip_flag` are equal to one, the clipping operation specified by `alpha_channel_clip_type_flag` should be applied first, followed by the alteration specified by `alpha_channel_incr_flag`, to obtain the interpretation sample value for the auxiliary picture luma sample.

Alpha blending composition is ordinarily performed with a background picture B, a foreground picture F, and a decoded auxiliary picture A, all of the same size. Assume for purposes of example illustration that the chroma resolutions of B and F, if different from the luma resolution, have been upsampled to the same resolution as the luma. Denote corresponding samples of B, F and A by *b*, *f* and *a*, respectively. Denote luma and chroma samples by subscripts *Y*, *Cb* and *Cr*. Each component, e.g., *Y*, is also assumed for purposes of example illustration to have the same bit depth in each of the pictures B and F. However, different components, e.g., *Y* and *Cb*, can have different bit depths in this example. The samples of pictures B and F may alternatively represent green, blue and red component values (see clause 7.3), although the equations use the subscripts *Y*, *Cb* and *Cr* for the three components.

Define the variables `alphaRange`, `alphaFwt` and `alphaBwt` for each luma sample a_Y of the auxiliary picture A as follows:

$$\text{alphaRange} = \text{Abs}(\text{alpha_opaque_value} - \text{alpha_transparent_value}) \quad (56)$$

$$\text{alphaFwt} = \text{Abs}(a_Y - \text{alpha_transparent_value}) \quad (57)$$

$$\text{alphaBwt} = \text{Abs}(a_Y - \text{alpha_opaque_value}) \quad (58)$$

A picture format that is often useful for editing or direct viewing, and that is commonly used, is called pre-multiplied-black video. Pre-multiplied-black video has the characteristic that the decoded picture F will appear the same regardless of whether it is viewed directly without alpha blending composition or is alpha blended with a black background. The use

of `alpha_channel_use_idc` equal to 0 corresponds with source video that is not pre-multiplied-black video, and the use of `alpha_channel_use_idc` equal to 1 corresponds with source video that is pre-multiplied-black video.

An example of operation of the alpha blending composition process to produce a displayed picture D with sample values `d` from the pictures B and F is as follows:

- If `alpha_channel_use_idc` is equal to 0, the samples `d` of the displayed picture D are calculated as follows:

$$d_Y = (\text{alphaFwt} * f_Y + \text{alphaBwt} * b_Y + \text{alphaRange} / 2) / \text{alphaRange} \quad (59)$$

$$d_{Cb} = (\text{alphaFwt} * f_{Cb} + \text{alphaBwt} * b_{Cb} + \text{alphaRange} / 2) / \text{alphaRange} \quad (60)$$

$$d_{Cr} = (\text{alphaFwt} * f_{Cr} + \text{alphaBwt} * b_{Cr} + \text{alphaRange} / 2) / \text{alphaRange} \quad (61)$$

- Otherwise (`alpha_channel_use_idc` is equal to 1), the samples `d` of the displayed picture D are calculated as follows:

$$d_Y = f_Y + (\text{alphaBwt} * b_Y + \text{alphaRange} / 2) / \text{alphaRange} \quad (62)$$

$$d_{Cb} = f_{Cb} + (\text{alphaBwt} * b_{Cb} + \text{alphaRange} / 2) / \text{alphaRange} \quad (63)$$

$$d_{Cr} = f_{Cr} + (\text{alphaBwt} * b_{Cr} + \text{alphaRange} / 2) / \text{alphaRange} \quad (64)$$

NOTE 2 – In this case, it is expected that the encoder produces its pre-multiplied-black source video picture S with sample values `s` from some original input picture T with sample values `t` as expressed by Equations 65 to 67, so that when the decoded picture F is a close approximation of the pre-multiplied-black source video picture S, the cascaded effect of Equations 65 to 67 followed by Equations 62 to 64 is approximately the same as expressed in Equations 59 to 61.

$$s_Y = (\text{alphaFwt} * t_Y) / \text{alphaRange} \quad (65)$$

$$s_{Cb} = (\text{alphaFwt} * t_{Cb}) / \text{alphaRange} \quad (66)$$

$$s_{Cr} = (\text{alphaFwt} * t_{Cr}) / \text{alphaRange} \quad (67)$$

NOTE 3 – In the event that the background picture B is represented using green, blue and red component values (see clause 7.3) in a manner such that the colour black is represented by all three component values being equal to 0, when the background picture B is black, the operation expressed by Equations 62 to 64 becomes simply $d_Y = f_Y$, $d_{Cb} = f_{Cb}$, and $d_{Cr} = f_{Cr}$. This can help to explain the "pre-multiplied black" term, as the expressions in Equations 65 to 67 are referred to as the pre-multiplication for the black background combination.

For the case with `alpha_channel_use_idc` equal to 1, somewhat modified processing should be applied when the colour representation domain is different from the use of green, blue, and red colour component values, or with the use of a non-zero black level. Unless the colour black is represented by all three component values b_Y , b_{Cb} , and b_{Cr} being equal to 0, Equations 62 to 64 do not simplify to $d_Y = f_Y$, $d_{Cb} = f_{Cb}$, and $d_{Cr} = f_{Cr}$ for pre-multiplied-black video content.

8.24 Extended DRAP indication SEI message

8.24.1 Extended DRAP indication SEI message syntax

extended_drap_indication(payloadSize) {	Descriptor
edrap_rap_id_minus1	u(16)
edrap_leading_pictures_decodable_flag	u(1)
edrap_reserved_zero_12bits	u(12)
edrap_num_ref_rap_pics_minus1	u(3)
for(i = 0; i <= edrap_num_ref_rap_pics_minus1; i++)	
edrap_ref_rap_id[i]	u(16)
}	

8.24.2 Extended DRAP indication SEI message semantics

The picture associated with an extended DRAP (EDRAP) indication SEI message is referred to as an EDRAP picture.

The presence of the EDRAP indication SEI message indicates that the constraints on picture order and picture referencing specified in this clause apply. These constraints can enable a decoder to properly decode the EDRAP picture and the pictures that are in the same layer and follow it in both decoding order and output order without needing to decode any other pictures in the same layer except the list of pictures referenceablePictures. The list referenceablePictures consists of the list of IRAP or EDRAP pictures that are present in the current CLVS and are identified by the `edrap_ref_rap_id[i]` syntax elements, and these pictures are listed in decoding order in the list.

The constraints indicated by the presence of the EDRAP indication SEI message, which shall all apply, are as follows:

- The EDRAP picture is a trailing picture.
- The EDRAP picture has a temporal sublayer identifier equal to 0.
- The EDRAP picture does not include any pictures in the same layer in the active entries of its reference picture lists except the referenceablePictures.
- Any picture that is in the same layer and follows the EDRAP picture in both decoding order and output order does not include, in the active entries of its reference picture lists, any picture that is in the same layer and precedes the EDRAP picture in decoding order or output order, with the exception of the referenceablePictures.
- Any picture in the list referenceablePictures does not include, in the active entries of its reference picture lists, any picture that is in the same layer and is not a picture at an earlier position in the list referenceablePictures.

NOTE – Consequently, the first picture in referenceablePictures, even when it is an EDRAP picture instead of an IRAP picture, does not include any picture from the same layer in the active entries of its reference picture lists.

edrap_rap_id_minus1 plus 1 specifies the RAP picture identifier, denoted as RapPicId, of the EDRAP picture.

Each IRAP or EDRAP picture is associated with a RapPicId value. The RapPicId value for an IRAP picture is inferred to be equal to 0. The RapPicId values for any two EDRAP pictures associated with the same IRAP picture shall be different.

edrap_leading_pictures_decodable_flag equal to 1 specifies that both of the following constraints apply:

- Any picture that is in the same layer and follows the EDRAP picture in decoding order shall follow, in output order, any picture that is in the same layer and precedes the EDRAP picture in decoding order.
- Any picture that is in the same layer and follows the EDRAP picture in decoding order and precedes the EDRAP picture in output order shall not include, in the active entries of its reference picture lists, any picture that is in the same layer and precedes the EDRAP picture in decoding order, with the exception of the referenceablePictures.

edrap_leading_pictures_decodable_flag equal to 0 does not impose such constraints.

edrap_reserved_zero_12bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **edrap_reserved_zero_12bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **edrap_reserved_zero_12bits**.

edrap_num_ref_rap_pics_minus1 plus 1 indicates the number of IRAP or EDRAP pictures that are within the same CLVS as the EDRAP picture and may be included in the active entries of the reference picture lists of the EDRAP picture.

edrap_ref_rap_id[i] indicates RapPicId of the i-th RAP picture that may be included in the active entries of the reference picture lists of the EDRAP picture. The i-th RAP picture shall be either the IRAP picture associated with the current EDRAP picture or an EDRAP picture associated with the same IRAP picture as the current EDRAP picture.

8.25 Display orientation SEI message

8.25.1 Display orientation SEI message syntax

display_orientation(payloadSize) {	Descriptor
display_orientation_cancel_flag	u(1)
if(!display_orientation_cancel_flag) {	
display_orientation_persistence_flag	u(1)
display_orientation_transform_type	u(3)
display_orientation_reserved_zero_3bits	u(3)
}	

8.25.2 Display orientation SEI message semantics

When the associated picture has PicOutputFlag equal to 1, the display orientation SEI message informs the decoder of a transformation that is recommended to be applied to the cropped decoded picture prior to display.

display_orientation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous display orientation SEI message in output order. **display_orientation_cancel_flag** equal to 0 indicates that display orientation information follows.

display_orientation_persistence_flag specifies the persistence of the display orientation SEI message for the current layer.

display_orientation_persistence_flag equal to 0 specifies that the display orientation SEI message applies to the current decoded picture only.

display_orientation_persistence_flag equal to 1 specifies that the display orientation 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 display orientation SEI message is output that follows the current picture in output order.

display_orientation_transform_type specifies the rotation and mirroring to be applied to the picture. When display_orientation_transform_type specifies both rotation and mirroring, rotation applies before mirroring. The values of display_transform_type are specified in Table 19.

Table 19 – display_orientation_transform_type values

Value	Description
0	no transform
1	mirroring horizontally
2	rotation by 180 degrees (anticlockwise)
3	rotation by 180 degrees (anticlockwise) before mirroring horizontally
4	rotation by 90 degrees (anticlockwise) before mirroring horizontally
5	rotation by 90 degrees (anticlockwise)
6	rotation by 270 degrees (anticlockwise) before mirroring horizontally
7	rotation by 270 degrees (anticlockwise)

display_orientation_reserved_zero_3bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for display_orientation_reserved_zero_3bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of display_orientation_reserved_zero_3bits.

8.26 Colour transform information SEI message

8.26.1 Colour transform information SEI message syntax

	Descriptor
colour_transform_info(payloadSize) {	
colour_transform_id	ue(v)
colour_transform_cancel_flag	u(1)
if(!colour_transform_cancel_flag) {	
colour_transform_persistence_flag	u(1)
colour_transform_video_signal_info_present_flag	u(1)
if(colour_transform_video_signal_info_present_flag) {	
colour_transform_full_range_flag	u(1)
colour_tranform primaries	u(8)
colour_transform_transfer_function	u(8)
colour_transform_matrix_coefficients	u(8)
}	
colour_transform_bit_depth_minus8	u(4)
colour_transform_log2_number_of_points_per_lut_minus1	u(3)
colour_transform_cross_component_flag	u(1)
}	

if(colour_transform_cross_component_flag)	
colour_transform_cross_comp_inferred_flag	u(1)
for(i = 0; i < colourTransformSize; i++)	
colour_transf_lut[0][i]	u(v)
if(colour_transform_cross_component_flag == 0 colour_transform_cross_comp_inferred_flag == 0) {	
colour_transform_lut2_present_flag	u(1)
for(i = 0; i < colourTransformSize; i++)	
colour_transf_lut[1][i]	u(v)
if(colour_transform_lut2_present_flag)	
for(i = 0; i < colourTransformSize; i++)	
colour_transf_lut[2][i]	u(v)
} else	
colour_transform_chroma_offset	u(v)
}	
}	

8.2.6.2 Colour transform information SEI message semantics

The colour transform information (CTI) SEI message provides information to enable remapping of the reconstructed colour samples of the output pictures for purposes such as converting the output pictures to a representation that is more suitable for an alternative display. The colour transform model used in the CTI SEI message is composed of a first piece-wise linear function applied to the first colour component. Depending on the values of syntax elements `colour_transform_cross_component_flag`, `colour_transform_cross_comp_inferred_flag`, and `colour_transform_lut2_present_flag`, one or two additional piece-wise linear functions may be signalled for the second and third colour components.

When `ChromaFormatIdc` is equal to 0 (monochrome), the CTI SEI message shall not be present, although decoders shall also allow such messages to be present and shall ignore any such CTI SEI messages when present.

colour_transform_id contains an identifying number that may be used to identify the purpose of the CTI. The value of `colour_transform_id` may be used (in a manner not specified in this Specification) to indicate that the input to the remapping process is the output of some conversion process that is not specified in this Specification, such as a conversion of the picture to some alternative colour representation (e.g., conversion from a YCbCr colour representation to a GBR colour representation). When more than one CTI SEI message is present with the same value of `colour_transform_id`, the content of these CTI SEI messages shall be the same. When CTI SEI messages are present that have more than one value of `colour_transform_id`, this may indicate that the remapping processes indicated by the different values of `colour_transform_id` are alternatives that are provided for different purposes or that a cascading of remapping processes is to be applied in a sequential order (an order that is not specified in this Specification). The value of `colour_transform_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `colour_transform_id` from 0 to 255 and from 512 to $2^{31} - 1$ may be used as determined by the application. Values of `colour_transform_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the CTI SEI messages containing a value of `colour_transform_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, and bitstreams conforming to this version of this Specification shall not contain `colour_transform_id` with such values.

NOTE – The `colour_transform_id` can be used to support different remapping processes that are suitable for different display scenarios. For example, different values of `colour_transform_id` may correspond to different remapped colour spaces supported by displays.

colour_transform_cancel_flag equal to 1 indicates that the CTI SEI message cancels the persistence of any previous CTI SEI message in output order that applies to the current layer. `colour_transform_cancel_flag` equal to 0 indicates that CTI follows.

colour_transform_persistence_flag specifies the persistence of the CTI SEI message for the current layer.

`colour_transform_persistence_flag` equal to 0 specifies that the CTI SEI message applies to the current decoded picture only.

colour_transform_persistence_flag equal to 1 specifies that the CTI 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 CTI SEI message is output that follows the current picture in output order.

colour_transform_video_signal_info_present_flag equal to 1 specifies that syntax elements colour_transform_full_range_flag, colour_transform primaries, colour_transform_transfer_function and colour_transform_matrix_coefficients are present, colour_transform_video_signal_info_present_flag equal to 0 specifies that syntax elements colour_transform_full_range_flag, colour_transform primaries, colour_transform_transfer_function and colour_transform_matrix_coefficients are not present.

colour_transform_full_range_flag has the same semantics as specified in clause 7.3 for the vui_full_range_flag syntax element, except that colour_transform_full_range_flag identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_transform_full_range_flag is inferred to be equal to the value of vui_full_range_flag.

colour_transform primaries has the same semantics as specified in clause 7.3 for the vui_colour primaries syntax element, except that colour_transform primaries identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_transform primaries is inferred to be equal to the value of vui_colour primaries.

colour_transform_transfer_function has the same semantics as specified in clause 7.3 for the vui_transfer_characteristics syntax element, except that colour_transform_transfer_function identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_transform_transfer_function is inferred to be equal to the value of vui_transfer_characteristics.

colour_transform_matrix_coefficients has the same semantics as specified in clause 7.3 for the vui_matrix_coeffs syntax element, except that colour_transform_matrix_coefficients identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_transform_matrix_coefficients is inferred to be equal to the value of vui_matrix_coeffs.

colour_transform_bit_depth_minus8 plus 8 specifies the bit depth of the colour components of the associated pictures for purposes of interpretation of the CTI SEI message. When any CTI SEI message is present with the value of colour_transform_bit_depth plus 8 not equal to the bit depth of the decoded colour components, the SEI message refers to the hypothetical result of a conversion operation performed to convert the decoded colour component samples to the bit depth equal to colour_transform_input_bit_depth plus 8.

The value of colour_transform_bit_depth plus 8 shall be in the range of 8 to 16, inclusive. Values of colour_transform_bit_depth from in the range of 17 to 23, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the CTI SEI messages that contain a value of colour_transform_bit_depth in the range of 17 to 23, inclusive, and bitstreams conforming to this version of this Specification shall not contain colour_transform_bit_depth with such values.

bitDepth is set equal to (colour_transform_bit_depth + 8).

colour_transform_log2_number_of_points_per_lut_minus1 specifies the log2 of the number of pivot points in the piece-wise linear remapping functions minus 1.

log2numLutPoints is set equal to (colour_transform_log2_number_of_points_per_lut_minus1 + 1).

numLutPoints is set equal to (1 << log2numLutPoints).

colourTransformSize is set equal to (numLutPoints + 1).

log2distX is set equal to (bitDepth – log2numLutPoints).

colour_transform_cross_component_flag equal to 1 indicates that the remapping of the second and third colour components is performed as cross-component remapping based on the first colour component. colour_transform_cross_component_flag equal to 0 indicates that intra-component remapping is applied to the second and third colour components.

maxIntraComp is set equal to (2 * (1 – colour_transform_cross_component_flag)).

colour_transform_cross_comp_inferred_flag equal to 1 indicates that the remapping piece-wise linear functions of the second and third colour components are inferred from the remapping piece-wise linear function of the first colour

component. `colour_transform_cross_comp_inferred_flag` equal to 0 indicates that the remapping piece-wise linear functions of the second and third colour components are signalled. When not present, the value of `colour_transform_cross_comp_inferred_flag` is inferred to be equal to 0.

`colour_transf_lut[c][i]` specifies the piecewise linear remapping function of the colour component of index `c`. When `colour_transf_lut[1][i]` is present and `colour_transf_lut[2][i]` is not present, the value of `colour_transf_lut[2][i]` is inferred to be equal to `colour_transf_lut[1][i]`. The length of `colour_transf_lut[c][i]` is $2 + \text{bitDepth} - \log_2 \text{numLutPoints}$ bits.

`colour_transform_lut2_present_flag` equal to 1 specifies that `colour_transf_lut[2][i]` is present in the CTI SEI message. `colour_transform_lut2_present_flag` equal to 0 specifies that `colour_transf_lut[2][i]` is not present in the CTI SEI message. When not present, the value of `colour_transform_lut2_present_flag` is inferred to be equal to 0.

`colour_transform_chroma_offset` specifies the CTI chroma offset. When not present, `colour_transform_chroma_offset` is inferred to be equal to 0. The length of `colour_transform_chroma_offset` is $2 + \text{bitDepth} - \log_2 \text{numLutPoints}$ bits.

The remapping process of the input picture components `rec[c]`, with width and height equal to `picWidth[c]` and `picHeight[c]`, respectively, to the output remapped picture components `map[c]`, for $c=0..2$, is performed as follows.

The array `pivotPointX` is derived as follows.

- For $j=0..(\text{numLutPoints} - 1)$, `pivotPointX[j]` is set equal to $(j \ll \log_2 \text{distX})$.

For $c=0..\text{maxIntraComp}$, the arrays `pivotPointY[c]` and `slope[c]` are derived as follows:

- `pivotPointY[c][0]` is set equal to `colour_transf_lut[c][0]`

- For $j=1..(\text{numLutPoints} - 1)$, `pivotPointY[c][j]` is derived as follows:

$$\text{pivotPointY}[c][j] = \text{pivotPointY}[c][j - 1] + \text{colour_transf_lut}[c][j] \quad (68)$$

- For $j=0..(\text{numLutPoints} - 1)$, `slope[c][j]` is derived as follows:

$$\text{slope}[c][j] = ((\text{colour_transf_lut}[c][j + 1] \ll 11) + (1 \ll (\log_2 \text{distX} - 1))) \gg \log_2 \text{distX} \quad (69)$$

When `colour_transform_cross_component_flag` is equal to 1, the arrays `ccPivotPointY[c]` and `ccSlope[c]` are derived as follows, for $c=1..2$:

- If `colour_transform_cross_comp_inferred_flag` is equal to 0, `ccPivotPointY[c]` is derived as follows:

- For $j=0..\text{numLutPoints}$, `ccPivotPointY[c][j]` is set equal to $(\text{colour_transf_lut}[c][j] \ll (11 - \log_2 \text{distX}))$.

- Otherwise (`colour_transform_cross_comp_inferred_flag` is equal to 1), `ccPivotPointY[c]` is derived as follows:

- For $j=0..(\text{numLutPoints} - 1)$, `tmpPivotPt[j]` is derived as follows:

- If `colour_transf_lut[0][j + 1]` is equal to 0, `tmpPivotPt[j]` is set equal to $(1 \ll 11)$.
- Otherwise, `tmpPivotPt[j]` is derived as follows:

$$\text{tmpPivotPt}[j] = (\text{colour_transf_lut}[0][j + 1] + \text{colour_transform_chroma_offset}) \ll (11 - \log_2 \text{distX}) \quad (70)$$

- The array `ccPivotPointY[c]` is derived as follows:

- For $j=1..(\text{numLutPoints} - 1)$, `ccPivotPointY[c][j]` is derived as follows:

$$\text{ccPivotPointY}[c][j] = (\text{tmpPivotPt}[j] + \text{tmpPivotPt}[j - 1] + 1) / 2 \quad (71)$$

- `ccPivotPointY[c][0]` is set equal to `tmpPivotPt[0]`.

- `ccPivotPointY[c][numLutPoints]` is set equal to `tmpPivotPt[numLutPoints - 1]`.

- For $j=0..(\text{numLutPoints} - 1)$, the value of `ccSlope[c][j]` is set equal to $(\text{ccPivotPointY}[c][j + 1] - \text{ccPivotPointY}[c][j])$.

For $c=0..\text{maxIntraComp}$, the intra-component remapping process of the input samples picture `rec[c]` into the remapped samples picture `map[c]` is performed as follows.

- for $i=0..\text{picWidth}[c] - 1, j=0..\text{picHeight}[c] - 1$, the following applies:

$$\begin{aligned} \text{idx} &= \text{rec}[c][i][j] \gg \log_2 \text{distX} \\ \text{map}[c][i][j] &= \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \text{pivotPointY}[c][\text{idx}] + \\ &\quad ((\text{slope}[c][\text{idx}] * (\text{rec}[i][j] - \text{pivotPointX}[\text{idx}]) + (1 \ll 10)) \gg 11)) \end{aligned} \quad (72)$$

When `colour_transform_cross_component_flag` is equal to 1, for $c=1..2$, the cross-component remapping process of the input samples picture `rec[c]` into the remapped samples picture `map[c]` is performed as follows:

- offset is set equal to $(1 \ll (\text{bitDepth} - 1))$.
- subWc and subHc are set equal to $(\text{picWidth}[0] / \text{picWidth}[c])$ and $(\text{picHeight}[0] / \text{picHeight}[c])$, respectively.
- For $i=0..\text{picWidth}[c]-1, j=0..\text{picHeight}[c]-1$, the following applies:

$$\begin{aligned}
 \text{coloc} &= \text{rec}[0][i * \text{SubWc}][j * \text{SubHc}] \\
 \text{idx} &= \text{coloc} \gg \log_2 \text{distX} \\
 \text{scale} &= \text{ccPivotPointY}[c][\text{idx}] + ((\text{ccSlope}[c][\text{idx}] * (\text{coloc} - \text{pivotPointX}[\text{idx}])) \gg \log_2 \text{distX}) \\
 \text{map}[c][i][j] &= \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \\
 &\quad ((\text{offset} \ll 11) + \text{scale} * (\text{rec}[c][i][j] - \text{offset}) + (1 \ll 10)) \gg 11)
 \end{aligned}
 \tag{73}$$

8.27 Shutter interval information SEI message

8.27.1 Shutter interval information SEI message syntax

shutter_interval_info(payloadSize) {	Descriptor
sii_time_scale	u(32)
sii_fixed_shutter_interval_within_clvs_flag	u(1)
if(fixed_shutter_interval_within_clvs_flag)	
sii_num_units_in_shutter_interval	u(32)
else {	
sii_max_sub_layers_minus1	u(3)
for(i = 0; i <= sii_max_sub_layers_minus1; i++)	
sii_sub_layer_num_units_in_shutter_interval[i]	u(32)
}	
}	

8.27.2 Shutter interval information SEI message semantics

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is the amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

sii_time_scale specifies the number of time units that pass in one second. The value of **sii_time_scale** shall not be equal to 0. For example, a time coordinate system that measures time using a 27 MHz clock has an **sii_time_scale** of 27 000 000.

sii_fixed_shutter_interval_within_clvs_flag equal to 1 specifies that the indicated shutter interval is the same for all temporal sublayers in the CLVS. **sii_fixed_shutter_interval_within_clvs_flag** equal to 0 specifies that the indicated shutter interval may not be the same for all temporal sublayers in the CLVS.

sii_num_units_in_shutter_interval, when **sii_fixed_shutter_interval_within_clvs_flag** is equal to 1, specifies the number of time units of a clock operating at the frequency **sii_time_scale** Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable **shutterInterval**, in units of seconds, is equal to the quotient of **sii_num_units_in_shutter_interval** divided by **sii_time_scale**. For example, to represent a shutter interval equal to 0.04 seconds, **sii_time_scale** may be equal to 27 000 000 and **sii_num_units_in_shutter_interval** may be equal to 1 080 000.

sii_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sublayers that may be present in the CLVS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2022-03 Annex D when **sii_max_sub_layers_minus1** is equal to 1 and **sii_fixed_shutter_interval_within_clvs_flag** is equal to 0.

sii_sub_layer_num_units_in_shutter_interval[*i*], when present, specifies the number of time units of a clock operating at the frequency **sii_time_scale** Hz that corresponds to the shutter interval of each picture with temporal sublayer identifier equal to *i* in the CLVS. The shutter interval for each picture with temporal sublayer identifier equal to *i* in the CLVS, denoted by the variable **subLayerShutterInterval**[*i*], in units of seconds, is equal to the quotient of **sii_sub_layer_num_units_in_shutter_interval**[*i*] divided by **sii_time_scale**.

The variable **subLayerShutterInterval**[*i*], corresponding to the indicated shutter interval of each picture with temporal sublayer identifier equal to *i* in the CLVS, is thus derived as follows:

$$\begin{aligned} &\text{if(} \text{sii_fixed_shutter_interval_within_clvs_flag} \text{)} \\ &\quad \text{subLayerShutterInterval[} i \text{]} = \text{sii_num_units_in_shutter_interval} \div \text{sii_time_scale} \\ &\text{else} \\ &\quad \text{subLayerShutterInterval[} i \text{]} = \text{sii_sub_layer_num_units_in_shutter_interval[} i \text{]} \div \text{sii_time_scale} \end{aligned} \quad (74)$$

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 neural-network post-processing filter (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, the filtering process for one picture, as specified in clause 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 NNPFs 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. 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.

For any particular pair of pictures **inputPicA** and **inputPicB** consecutive in output order in **CroppedDecodedPictures**, when there are one or more pictures **interpolatedPicSetA** in **ListNnpfOutputPics** between **inputPicA** and **inputPicB** in output order, the pictures in **interpolatedPicSetA** shall be among the pictures that were output by applying a particular NNPF **nnpfA** with **PictureRateUpsamplingFlag** equal to 1 when a particular picture **currPicA** in **CroppedDecodedPictures** was the current picture. The application of any other NNPF that was used in the filtering process for one picture when **currPicA** was the current picture or the application of any NNPF (including **nnpfA**) 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 NNPF output pictures between any particular pair of consecutive input pictures more than once.

8.28.1.2 Filtering process for one picture using an NNPF

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

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 neural-network post-filter characteristics (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.

When the applied NNPF is the last NNPF that is applied to the current picture, 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.

8.28.2 Neural-network post-filter characteristics SEI message

8.28.2.1 Neural-network post-filter characteristics SEI message syntax

	Descriptor
<code>nn_post_filter_characteristics(payloadSize) {</code>	
nnpfc_purpose	u(16)
nnpfc_id	ue(v)
nnpfc_base_flag	u(1)
nnpfc_mode_idc	ue(v)
if(nnpfc_mode_idc == 1) {	
while(!byte_aligned())	
nnpfc_alignment_zero_bit_a	u(1)
nnpfc_tag_uri	st(v)
nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)
if(nnpfc_property_present_flag) {	
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if(nnpfc_num_input_pics_minus1 > 0) {	
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_filtering_flag[i]	u(1)
nnpfc_absent_input_pic_zero_flag	u(1)
}	
if(ChromaUpsamplingFlag)	
nnpfc_out_sub_c_flag	u(1)
if(ColourizationFlag)	
nnpfc_out_colour_format_idc	u(2)
if(ResolutionResamplingFlag) {	
nnpfc_pic_width_num_minus1	ue(v)
nnpfc_pic_width_denom_minus1	ue(v)
nnpfc_pic_height_num_minus1	ue(v)
nnpfc_pic_height_denom_minus1	ue(v)
}	
if(PictureRateUpsamplingFlag)	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
nnpfc_auxiliary_inp_idc	ue(v)
nnpfc_inp_order_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
if(nnpfc_inp_order_idc != 1)	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
if(nnpfc_inp_order_idc > 0)	
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_out_format_idc	ue(v)
nnpfc_out_order_idc	ue(v)

if(nnpfc_out_format_idc == 1) {	
if(nnpfc_out_order_idc != 1)	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
if(nnpfc_out_order_idc != 0)	
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	
nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
if(nnpfc_out_format_idc == 1) {	
nnpfc_matrix_coeffs	u(8)
nnpfc_full_range_flag	u(1)
}	
}	
if(nnpfc_out_order_idc > 0)	
nnpfc_chroma_loc_info_present_flag	u(1)
if(nnpfc_chroma_loc_info_present_flag)	
nnpfc_chroma_sample_loc_type_frame	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
if(nnpfc_inp_order_idc != 1)	
nnpfc_luma_padding_val	ue(v)
if(nnpfc_inp_order_idc != 0) {	
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)
if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)
nnpfc_num_parameters_idc	u(6)
nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
nnpfc_num_metadata_extension_bits	ue(v)
if(nnpfc_num_metadata_extension_bits > 0)	

nnpfc_reserved_metadata_extension	u(v)
}	
/* ISO/IEC 15938-17 bitstream */	
if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_alignment_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

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:

- 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.
- A chroma format indicator, denoted herein by `ChromaFormatIdc`, as described in clause 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 input pictures with index `idx` in the range of 0 to `numInputPics – 1`, inclusive.

Input picture with index 0 corresponds 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.

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

NOTE 1 – More than one NNPFC SEI message can be present for the same picture. When more than one NNPFC SEI message with different values of `nnpfc_id` is present or activated for the same picture, they can have the same value or different values of `nnpfc_purpose` and the same value or different values of `nnpfc_mode_idc`.

nnpfc_purpose indicates the purpose of the NNPF as specified in Table 20, where $(\text{nnpfc_purpose} \& \text{bitMask})$ not equal to 0 indicates that the NNPF has the purpose associated with the `bitMask` value in Table 20. When `nnpfc_purpose` is greater than 0 and $(\text{nnpfc_purpose} \& \text{bitMask})$ is equal to 0, the purpose associated with the `bitMask` value is not applicable to the NNPF. When `nnpfc_purpose` is equal to 0, the NNPF may be used as determined by the application.

All NNPFC SEI messages with a particular value of `nnpfc_id` within a CLVS shall have the same value of `nnpfc_purpose`.

The value of `nnpfc_purpose` shall be in the range of 0 to 63, inclusive, in bitstreams conforming to this version of this Specification. Values of 64 to 65 535, inclusive, for `nnpfc_purpose` 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 conforming to this version of this Specification shall ignore NNPFC SEI messages with `nnpfc_purpose` in the range of 64 to 65 535, inclusive.

Table 20 – Definition of nnpfc_purpose

bitMask	Interpretation
0x01	General visual quality improvement
0x02	Chroma upsampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format)
0x04	Resolution resampling (increasing or decreasing the width or height)
0x08	Picture rate upsampling
0x10	Bit depth upsampling (increasing the luma bit depth or the chroma bit depth)
0x20	Colourization

The variables ChromaUpsamplingFlag, ResolutionResamplingFlag, PictureRateUpsamplingFlag, BitDepthUpsamplingFlag, and ColourizationFlag, specifying whether nnpfc_purpose indicates the purpose of the NNPF to include chroma upsampling, resolution resampling, picture rate upsampling, bit depth upsampling, and colourization, respectively, are derived as follows:

$$\begin{aligned}
 \text{ChromaUpsamplingFlag} &= ((\text{nnpfc_purpose} \& \text{0x02}) > 0) ? 1 : 0 \\
 \text{ResolutionResamplingFlag} &= ((\text{nnpfc_purpose} \& \text{0x04}) > 0) ? 1 : 0 \\
 \text{PictureRateUpsamplingFlag} &= ((\text{nnpfc_purpose} \& \text{0x08}) > 0) ? 1 : 0 \\
 \text{BitDepthUpsamplingFlag} &= ((\text{nnpfc_purpose} \& \text{0x10}) > 0) ? 1 : 0 \\
 \text{ColourizationFlag} &= ((\text{nnpfc_purpose} \& \text{0x20}) > 0) ? 1 : 0
 \end{aligned}
 \tag{75}$$

NOTE 2 – When a reserved value of nnpfc_purpose is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by nnpfc_purpose being equal to that value or any one of a set of values including that value.

When ChromaFormatIdc is equal to 3, ChromaUpsamplingFlag shall be equal to 0.

When ChromaUpsamplingFlag is equal to 1, ColourizationFlag shall be equal to 0.

When PictureRateUpsamplingFlag is equal to 1 and the input picture with index 0 is associated with a frame packing arrangement SEI message with fp_arrangement_type equal to 5, all input pictures are associated with a frame packing arrangement SEI message with fp_arrangement_type equal to 5 and the same value of fp_current_frame_is_frame0_flag.

nnpfc_id contains an identifying number that may be used to identify an NNPF. The value of nnpfc_id shall be in the range of 0 to $2^{32} - 2$, inclusive. Values of nnpfc_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification encountering an NNPF SEI message with nnpfc_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore the SEI message.

When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS, the following applies:

- This SEI message specifies a base NNPF.
- This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS.

nnpfc_base_flag equal to 1 specifies that the SEI message specifies the base NNPF. nnpfc_base_flag equal to 0 specifies that the SEI message specifies an update relative to the base NNPF.

The following constraints apply to the value of nnpfc_base_flag:

- When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS, the value of nnpfc_base_flag shall be equal to 1.
- All NNPF SEI messages in a CLVS that have a particular nnpfc_id value and nnpfc_base_flag equal to 1 shall have identical SEI payload content.

When nnpfc_base_flag is equal to 0, the following applies:

- This SEI message defines an update relative to the preceding base NNPF in decoding order with the same nnpfc_id value. Updates are not cumulative but rather each update is applied on the base NNPF, which is the NNPF specified by the first NNPF SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS. The NNPF defined by this SEI message is obtained by applying the update defined by this SEI message relative to the base NNPF with the same nnpfc_id value.

- This SEI message pertains to 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 follows the current decoded picture in output order within the current CLVS and is associated with a subsequent NNPFC SEI message, in decoding order, having `nnpfc_base_flag` equal to 0 and that particular `nnpfc_id` value within the current CLVS, whichever is earlier.

nnpfc_mode_idc, when equal to 0, indicates that the neural network information is contained in the NNPFC SEI message, and the neural network information is in the format of an ISO/IEC 15938-17 bitstream. `nnpfc_mode_idc` equal to 1 indicates that the neural network information is identified by the URI indicated by `nnpfc_uri` with the format identified by the tag URI `nnpfc_tag_uri`.

The value of `nnpfc_mode_idc` shall be in the range of 0 to 255, inclusive. Values of 2 to 255, inclusive, for `nnpfc_mode_idc` 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 conforming to this version of this Specification shall ignore NNPFC SEI messages with `nnpfc_mode_idc` in the range of 2 to 255, inclusive.

nnpfc_alignment_zero_bit_a shall be equal to 0.

nnpfc_tag_uri contains a tag URI with syntax and semantics as specified in IETF RFC 4151 identifying the format and associated information about the neural network used as a base NNPFC or an update relative to the base NNPFC with the same `nnpfc_id` value specified by `nnpfc_uri`.

NOTE 3 – `nnpfc_tag_uri` enables uniquely identifying the format of neural network data specified by `nnpfc_uri` without needing a central registration authority.

`nnpfc_tag_uri` equal to "tag:iso.org,2023:15938-17" indicates that the neural network data identified by `nnpfc_uri` conforms to ISO/IEC 15938-17.

nnpfc_uri contains a URI with syntax and semantics as specified in IETF Internet Standard 66 identifying the neural network used as a base NNPFC or an update relative to the base NNPFC with the same `nnpfc_id` value.

nnpfc_property_present_flag equal to 1 specifies that syntax elements related to the filter properties including purpose, input formatting, output formatting, and complexity are present. `nnpfc_property_present_flag` equal to 0 specifies that no syntax elements related to the filter properties are present.

When `nnpfc_base_flag` is equal to 1, `nnpfc_property_present_flag` shall be equal to 1.

When `nnpfc_property_present_flag` is equal to 0, the values of all syntax elements that may be present only when `nnpfc_property_present_flag` is equal to 1 are inferred to be equal to their corresponding syntax elements, respectively, in the NNPFC SEI message that contains the base NNPFC for which this SEI message provides an update.

When an NNPFC SEI message `nnpfcCurr` is not the first NNPFC SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, is not a repetition of the first NNPFC SEI message with that particular `nnpfc_id` value (in this case the value of `nnpfc_base_flag` is equal to 0), and the value of `nnpfc_property_present_flag` is equal to 1, the following constraints apply:

- The values of syntax elements following `nnpfc_property_present_flag` and preceding `nnpfc_complexity_info_present_flag`, in decoding order, in the NNPFC SEI message shall be the same as the values of corresponding syntax elements in the first NNPFC SEI message, in decoding order, that has that particular `nnpfc_id` value within the current CLVS.
- Either `nnpfc_complexity_info_present_flag` shall be equal to 0 or both `nnpfc_complexity_info_present_flag` shall be equal to 1 in the first NNPFC SEI message, in decoding order, that has that particular `nnpfc_id` value within the current CLVS (denoted as `nnpfcBase` below) and all the following constraints apply:
- `nnpfc_parameter_type_idc` in `nnpfcCurr` shall be equal to `nnpfc_parameter_type_idc` in `nnpfcBase`.
- `nnpfc_log2_parameter_bit_length_minus3` in `nnpfcCurr`, when present, shall be less than or equal to `nnpfc_log2_parameter_bit_length_minus3` in `nnpfcBase`.
- If `nnpfc_num_parameters_idc` in `nnpfcBase` is equal to 0, `nnpfc_num_parameters_idc` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_num_parameters_idc` in `nnpfcBase` is greater than 0), `nnpfc_num_parameters_idc` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_num_parameters_idc` in `nnpfcBase`.
- If `nnpfc_num_kmac_operations_idc` in `nnpfcBase` is equal to 0, `nnpfc_num_kmac_operations_idc` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_num_kmac_operations_idc` in `nnpfcBase` is greater than 0), `nnpfc_num_kmac_operations_idc` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_num_kmac_operations_idc` in `nnpfcBase`.

- If `nnpfc_total_kilobyte_size` in `nnpfcBase` is equal to 0, `nnpfc_total_kilobyte_size` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_total_kilobyte_size` in `nnpfcBase` is greater than 0), `nnpfc_total_kilobyte_size` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_total_kilobyte_size` in `nnpfcBase`.

`nnpfc_num_input_pics_minus1` plus 1 specifies the number of pictures used as input for the NNPF. The value of `nnpfc_num_input_pics_minus1` shall be in the range of 0 to 63, inclusive. When `PictureRateUpsamplingFlag` is equal to 1, the value of `nnpfc_num_input_pics_minus1` shall be greater than 0.

The variable `numInputPics`, specifying the number of pictures used as input for the NNPF, is derived as follows:

$$\text{numInputPics} = \text{nnpfc_num_input_pics_minus1} + 1 \quad (76)$$

`nnpfc_input_pic_filtering_flag[i]` equal to 1 indicates that for the *i*-th input picture the NNPF generates a corresponding output picture. **`nnpfc_input_pic_filtering_flag[i]`** equal to 0 indicates that for the *i*-th input picture the NNPF does not generate a corresponding output picture. Each NNPF-generated picture is stored in the output tensor of the NNPF. When `nnpfc_num_input_pics_minus1` is equal to 0, `nnpfc_input_pic_filtering_flag[0]` is inferred to be equal to 1. When `PictureRateUpsamplingFlag` is equal to 0 and `nnpfc_num_input_pics_minus1` is greater than 0, `nnpfc_input_pic_filtering_flag[i]` shall be equal to 1 for at least one value of *i* in the range of 0 to `nnpfc_num_input_pics_minus1`, inclusive.

`nnpfc_absent_input_pic_zero_flag` equal to 1 indicates that the NNPF expects an input 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` 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 is present in the bitstream.

`nnpfc_out_sub_c_flag` specifies the values of the variables `outSubWidthC` and `outSubHeightC` when `ChromaUpsamplingFlag` is equal to 1. **`nnpfc_out_sub_c_flag`** equal to 1 specifies that `outSubWidthC` is equal to 1 and `outSubHeightC` is equal to 1. **`nnpfc_out_sub_c_flag`** equal to 0 specifies that `outSubWidthC` is equal to 2 and `outSubHeightC` is equal to 1. When `ChromaFormatIdc` is equal to 2 and **`nnpfc_out_sub_c_flag`** is present, the value of **`nnpfc_out_sub_c_flag`** shall be equal to 1.

`nnpfc_out_colour_format_idc`, when `ColourizationFlag` is equal to 1, specifies the colour format of the NNPF-generated pictures and consequently the values of the variables `outSubWidthC` and `outSubHeightC`. **`nnpfc_out_colour_format_idc`** equal to 1 specifies that the colour format of the NNPF-generated pictures is the 4:2:0 format and `outSubWidthC` and `outSubHeightC` are both equal to 2. **`nnpfc_out_colour_format_idc`** equal to 2 specifies that the colour format of the NNPF-generated pictures is the 4:2:2 format and `outSubWidthC` is equal to 2 and `outSubHeightC` is equal to 1. **`nnpfc_out_colour_format_idc`** equal to 3 specifies that the colour format of the NNPF-generated pictures is the 4:4:4 format and `outSubWidthC` and `outSubHeightC` are both equal to 1. The value of **`nnpfc_out_colour_format_idc`** shall not be equal to 0.

When `ChromaUpsamplingFlag` and `ColourizationFlag` are both equal to 0, `outSubWidthC` and `outSubHeightC` are inferred to be equal to `SubWidthC` and `SubHeightC`, respectively.

`nnpfc_pic_width_num_minus1` plus 1 and **`nnpfc_pic_width_denom_minus1`** plus 1 specify the numerator and denominator, respectively, for the resampling ratio of the width of the NNPF-generated pictures relative to `CroppedWidth`. Both **`nnpfc_pic_width_num_minus1`** and **`nnpfc_pic_width_denom_minus1`** shall be in the range of 0 to 65 535, inclusive.

The value of $(\text{nnpfc_pic_width_num_minus1} + 1) \div (\text{nnpfc_pic_width_denom_minus1} + 1)$ shall be in the range of $1 \div 16$ to 16, inclusive. When **`nnpfc_pic_width_num_minus1`** and **`nnpfc_pic_width_denom_minus1`** are not present, the values of **`nnpfc_pic_width_num_minus1`** and **`nnpfc_pic_width_denom_minus1`** are both inferred to be equal to 0.

The variable `nnpfcOutputPicWidth`, representing the width of the luma sample arrays of the NNPF-generated pictures, is derived as follows:

$$\text{nnpfcOutputPicWidth} = \text{Ceil}(\text{CroppedWidth} * (\text{nnpfc_pic_width_num_minus1} + 1) \div (\text{nnpfc_pic_width_denom_minus1} + 1)) \quad (77)$$

It is a requirement of bitstream conformance that the value of `nnpfcOutputPicWidth % outSubWidthC` shall be equal to 0.

`nnpfc_pic_height_num_minus1` plus 1 and **`nnpfc_pic_height_denom_minus1`** plus 1 specify the numerator and denominator, respectively, for the resampling ratio of the height of the NNPF-generated pictures relative to `CroppedHeight`. Both **`nnpfc_pic_height_num_minus1`** and **`nnpfc_pic_height_denom_minus1`** shall be in the range of 0 to 65 535, inclusive.

The value of $(\text{nnpfc_pic_height_num_minus1} + 1) \div (\text{nnpfc_pic_height_denom_minus1} + 1)$ shall be in the range of $1 \div 16$ to 16, inclusive. When $\text{nnpfc_pic_height_num_minus1}$ and $\text{nnpfc_pic_height_denom_minus1}$ are not present, the values of $\text{nnpfc_pic_height_num_minus1}$ and $\text{nnpfc_pic_height_denom_minus1}$ are both inferred to be equal to 0.

The variable $\text{nnpfcOutputPicHeight}$, representing the height of the luma sample arrays of the NNPF-generated pictures, is derived as follows:

$$\text{nnpfcOutputPicHeight} = \text{Ceil}(\text{CroppedHeight} * (\text{nnpfc_pic_height_num_minus1} + 1) \div (\text{nnpfc_pic_height_denom_minus1} + 1)) \quad (78)$$

It is a requirement of bitstream conformance that the value of $\text{nnpfcOutputPicHeight} \% \text{outSubHeightC}$ shall be equal to 0.

When $\text{ResolutionResamplingFlag}$ is equal to 1, at least one the following conditions shall be true:

- The value of $\text{nnpfcOutputPicWidth}$ is not equal to CroppedWidth .
- The value of $\text{nnpfcOutputPicHeight}$ is not equal to CroppedHeight .

$\text{nnpfc_interpolated_pics}[i]$ specifies the number of interpolated pictures generated by the NNPF between the i -th and the $(i + 1)$ -th input picture for the NNPF. The value of $\text{nnpfc_interpolated_pics}[i]$ shall be in the range of 0 to 63, inclusive. When the $\text{nnpfc_interpolated_pics}[i]$ syntax elements are present, the value of $\text{nnpfc_interpolated_pics}[i]$ shall be greater than 0 for at least one value of i in the range of 0 to $\text{nnpfc_num_input_pics_minus1} - 1$, inclusive.

NOTE 4 – When $\text{PictureRateUpsamplingFlag}$ is equal to 1 for an NNPF and the NNPF SEI message that activated this NNPF has $\text{nnpfa_persistence_flag}$ equal to 1, only for a single value of i in the range of 0 to $\text{numInputPics} - 1$, inclusive, the value of $\text{nnpfc_interpolated_pics}[i]$ is greater than 0.

The variables $\text{NumInpPicsInOutputTensor}$, specifying the number of pictures that have a corresponding input picture and are present in the output tensor of the NNPF, $\text{InpIdx}[idx]$, specifying the input picture index, to the list of input pictures in reverse output order, of the idx -th picture that is present in the output tensor of the NNPF and has a corresponding input picture, and $\text{numPicsInOutputTensor}$, specifying the total number of pictures present in the output tensor of the NNPF, are derived as follows:

```
for( i = 0, numPicsInOutputTensor = 0; i < numInputPics; i++ )
    if( nnpfc_input_pic_filtering_flag[ i ] ) {
        InpIdx[ numPicsInOutputTensor ] = i
        numPicsInOutputTensor++
    }
NumInpPicsInOutputTensor = numPicsInOutputTensor
if( PictureRateUpsamplingFlag )
    for( i = 0; i <= numInputPics - 2; i++ )
        numPicsInOutputTensor += nnpfc_interpolated_pics[ i ]
```

(79)

$\text{nnpfc_component_last_flag}$ equal to 1 indicates that the last dimension in the input tensor inputTensor to the NNPF and the output tensor outputTensor of the NNPF is used for a current channel. $\text{nnpfc_component_last_flag}$ equal to 0 indicates that the third dimension in the input tensor inputTensor to the NNPF and the output tensor outputTensor of the NNPF is used for a current channel.

NOTE 5 – The first dimension in the input tensor and in the output tensor is used for the batch index, which is a common practice in some neural network frameworks. While the equations in the semantics of this SEI message use the batch size corresponding to the batch index equal to 0, it is up to the post-processing implementation to determine the batch size used as the input to the neural network inference process.

NOTE 6 – For example, when $\text{nnpfc_inp_order_idc}$ is equal to 3 and $\text{nnpfc_auxiliary_inp_idc}$ is equal to 1, there are 7 channels in the input tensor, including four luma matrices, two chroma matrices, and one auxiliary input matrix. In this case, the process $\text{DeriveInputTensors}()$ would derive each of these 7 channels of the input tensor one by one, and when a particular channel of these channels is processed, that channel is referred to as the current channel during the process.

$\text{nnpfc_inp_format_idc}$ indicates the method of converting a sample value of the input picture to an input value to the NNPF. The value of $\text{nnpfc_inp_format_idc}$ shall be in the range of 0 to 255, inclusive. Values of $\text{nnpfc_inp_format_idc}$ in the range of 2 to 255, inclusive, are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore NNPF SEI messages with $\text{nnpfc_inp_format_idc}$ in the range of 2 to 255, inclusive.

When $\text{nnpfc_inp_format_idc}$ is equal to 0, the input values to the NNPF are real numbers and the functions $\text{InpY}()$ and $\text{InpC}()$ are specified as follows:

$$\text{InpY}(x) = x \div ((1 \ll \text{BitDepth}_Y) - 1) \quad (80)$$

$$\text{InpC}(x) = x \div ((1 \ll \text{BitDepth}_C) - 1) \quad (81)$$

When `nnpfc_inp_format_idc` is equal to 1, the input values to the NNPF are unsigned integer numbers and the functions `InpY()` and `InpC()` are specified as follows:

```

shiftY = BitDepthY - inpTensorBitDepthY
if( inpTensorBitDepthY >= BitDepthY )
    InpY( x ) = x << ( inpTensorBitDepthY - BitDepthY )
else
    InpY( x ) = Clip3( 0, ( 1 << inpTensorBitDepthY ) - 1, ( x + ( 1 << ( shiftY - 1 ) ) ) >> shiftY )

```

(82)

```

shiftC = BitDepthC - inpTensorBitDepthC
if( inpTensorBitDepthC >= BitDepthC )
    InpC( x ) = x << ( inpTensorBitDepthC - BitDepthC )
else
    InpC( x ) = Clip3( 0, ( 1 << inpTensorBitDepthC ) - 1, ( x + ( 1 << ( shiftC - 1 ) ) ) >> shiftC )

```

(83)

The variable `inpTensorBitDepthY` is derived from the syntax element `nnpfc_inp_tensor_luma_bitdepth_minus8` as specified below. The variable `inpTensorBitDepthC` is derived from the syntax element `nnpfc_inp_tensor_chroma_bitdepth_minus8` as specified below.

nnpfc_auxiliary_inp_idc greater than 0 indicates that auxiliary input data is present in the input tensor of the NNPF. `nnpfc_auxiliary_inp_idc` equal to 0 indicates that auxiliary input data is not present in the input tensor. `nnpfc_auxiliary_inp_idc` equal to 1 specifies that auxiliary input data is derived as specified in Equation 95.

The value of `nnpfc_auxiliary_inp_idc` shall be in the range of 0 to 255, inclusive. Values of 2 to 255, inclusive, for `nnpfc_auxiliary_inp_idc` 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 conforming to this version of this Specification shall ignore NNPF SEI messages with `nnpfc_auxiliary_inp_idc` in the range of 2 to 255, inclusive.

nnpfc_inp_order_idc indicates the method of ordering the sample arrays of an input picture to form an input tensor to the NNPF.

The value of `nnpfc_inp_order_idc` shall be in the range of 0 to 255, inclusive. Values of 4 to 255, inclusive, for `nnpfc_inp_order_idc` 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 conforming to this version of this Specification shall ignore NNPF SEI messages with `nnpfc_inp_order_idc` in the range of 4 to 255, inclusive.

When `ChromaFormatIdc` is not equal to 1, `nnpfc_inp_order_idc` shall not be equal to 3.

When `ChromaFormatIdc` is equal to 0, `nnpfc_inp_order_idc` shall be equal to 0.

When `ChromaUpsamplingFlag` is equal to 1, `nnpfc_inp_order_idc` shall not be equal to 0.

Table 21 contains an informative description of `nnpfc_inp_order_idc` values.

Table 21 – Description of `nnpfc_inp_order_idc` values

<code>nnpfc_inp_order_idc</code>	Description
0	If <code>nnpfc_auxiliary_inp_idc</code> is equal to 0, one luma matrix is present in the input tensor for each input picture, and the number of channels is 1. Otherwise, when <code>nnpfc_auxiliary_inp_idc</code> is equal to 1, one luma matrix and one auxiliary input matrix are present, and the number of channels is 2.
1	If <code>nnpfc_auxiliary_inp_idc</code> is equal to 0, two chroma matrices are present in the input tensor, and the number of channels is 2. Otherwise, when <code>nnpfc_auxiliary_inp_idc</code> is equal to 1, two chroma matrices and one auxiliary input matrix are present, and the number of channels is 3.
2	If <code>nnpfc_auxiliary_inp_idc</code> is equal to 0, one luma and two chroma matrices are present in the input tensor, and the number of channels is 3. Otherwise, when <code>nnpfc_auxiliary_inp_idc</code> is equal to 1, one luma matrix, two chroma matrices and one auxiliary input matrix are present, and the number of channels is 4.

Table 21 – Description of nnpfc_inp_order_idc values

nnpfc_inp_order_idc	Description
3	If nnpfc_auxiliary_inp_idc is equal to 0, four luma matrices and two chroma matrices are present in the input tensor, and the number of channels is 6. Otherwise, when nnpfc_auxiliary_inp_idc is equal to 1, four luma matrices, two chroma matrices, and one auxiliary input matrix are present in the input tensor, and the number of channels is 7. The luma channels are derived in an interleaved manner as illustrated in Figure 12. This nnpfc_inp_order_idc can only be used when the input chroma format is 4:2:0.
4..255	Reserved

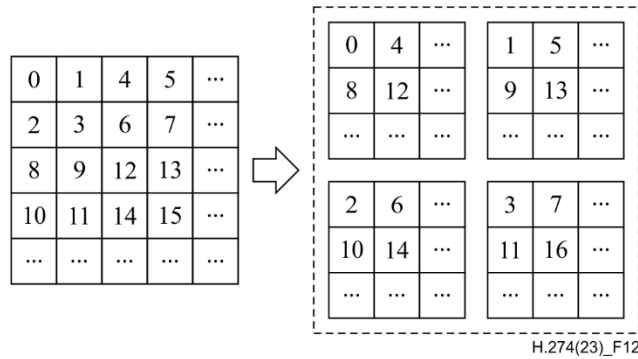


Figure 12 – Illustration of deriving the four luma channels (right) from the luma component (left) when nnpfc_inp_order_idc is equal to 3

nnpfc_inp_tensor_luma_bitdepth_minus8 plus 8 specifies the bit depth of luma sample values in the input integer tensor. The value of $\text{inpTensorBitDepth}_Y$ is derived as follows:

$$\text{inpTensorBitDepth}_Y = \text{nnpfc_inp_tensor_luma_bitdepth_minus8} + 8 \quad (84)$$

It is a requirement of bitstream conformance that the value of **nnpfc_inp_tensor_luma_bitdepth_minus8** shall be in the range of 0 to 24, inclusive.

nnpfc_inp_tensor_chroma_bitdepth_minus8 plus 8 specifies the bit depth of chroma sample values in the input integer tensor. The value of $\text{inpTensorBitDepth}_C$ is derived as follows:

$$\text{inpTensorBitDepth}_C = \text{nnpfc_inp_tensor_chroma_bitdepth_minus8} + 8 \quad (85)$$

It is a requirement of bitstream conformance that the value of **nnpfc_inp_tensor_chroma_bitdepth_minus8** shall be in the range of 0 to 24, inclusive.

nnpfc_out_format_idc equal to 0 indicates that the sample values output by the NNPF are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth bitDepth for subsequent post-processing or displaying.

nnpfc_out_format_idc equal to 1 indicates that the luma sample values output by the NNPF are unsigned integer numbers in the range of 0 to $(1 \ll \text{outTensorBitDepth}_Y) - 1$, inclusive, and the chroma sample values output by the NNPF are unsigned integer numbers in the range of 0 to $(1 \ll \text{outTensorBitDepth}_C) - 1$, inclusive.

The value of **nnpfc_out_format_idc** shall be in the range of 0 to 255, inclusive. Values of 2 to 255, inclusive, for **nnpfc_out_format_idc** are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore NNPF SEI messages with **nnpfc_out_format_idc** in the range of 2 to 255, inclusive.

nnpfc_out_order_idc indicates the output order of samples resulting from the NNPF.

The value of **nnpfc_out_order_idc** shall be in the range of 0 to 255, inclusive. Values of 4 to 255, inclusive, for **nnpfc_out_order_idc** 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 conforming to this version of this Specification shall ignore NNPF SEI messages with **nnpfc_out_order_idc** in the range of 4 to 255, inclusive.

When **ChromaUpsamplingFlag** is equal to 1, **nnpfc_out_order_idc** shall not be equal to 0 or 3.

When ColourizationFlag is equal to 1, nnpfc_out_order_idc shall not be equal to 0.

Table 22 contains an informative description of nnpfc_out_order_idc values.

Table 22 – Description of nnpfc_out_order_idc values

nnpfc_out_order_idc	Description
0	Only the luma matrix is present in the output tensor, thus the number of channels is 1.
1	Only the chroma matrices are present in the output tensor, thus the number of channels is 2.
2	The luma and chroma matrices are present in the output tensor, thus the number of channels is 3.
3	Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This nnpfc_out_order_idc can only be used when the output chroma format is 4:2:0.
4..255	Reserved

nnpfc_out_tensor_luma_bitdepth_minus8 plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of nnpfc_out_tensor_luma_bitdepth_minus8 shall be in the range of 0 to 24, inclusive. The value of outTensorBitDepth_Y is derived as follows:

$$\text{outTensorBitDepth}_Y = \text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8 \quad (86)$$

nnpfc_out_tensor_chroma_bitdepth_minus8 plus 8 specifies the bit depth of chroma sample values in the output integer tensor. The value of nnpfc_out_tensor_chroma_bitdepth_minus8 shall be in the range of 0 to 24, inclusive. The value of outTensorBitDepth_C is derived as follows:

$$\text{outTensorBitDepth}_C = \text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8 \quad (87)$$

When BitDepthUpsamplingFlag is equal to 1, the value of nnpfc_out_format_idc shall be equal to 1 and at least one of the following conditions shall be true:

- nnpfc_out_tensor_luma_bitdepth_minus8 is present and outTensorBitDepth_Y is greater than BitDepth_Y.
- nnpfc_out_tensor_chroma_bitdepth_minus8 is present and outTensorBitDepth_C is greater than BitDepth_C.

When nnpfc_inp_tensor_luma_bitdepth_minus8, nnpfc_inp_tensor_chroma_bitdepth_minus8, nnpfc_out_tensor_luma_bitdepth_minus8, and nnpfc_out_tensor_chroma_bitdepth_minus8 are present and outTensorBitDepth_Y is greater than inpTensorBitDepth_Y, outTensorBitDepth_C shall not be less than inpTensorBitDepth_C. When nnpfc_inp_tensor_luma_bitdepth_minus8, nnpfc_inp_tensor_chroma_bitdepth_minus8, nnpfc_out_tensor_luma_bitdepth_minus8, and nnpfc_out_tensor_chroma_bitdepth_minus8 are present and outTensorBitDepth_C is greater than inpTensorBitDepth_C, outTensorBitDepth_Y shall not be less than inpTensorBitDepth_Y.

nnpfc_separate_colour_description_present_flag equal to 1 indicates that a distinct combination of colour primaries, transfer characteristics, matrix coefficients, and scaling and offset values applied in association with the matrix coefficients for the picture resulting from the NNPF is specified in the SEI message syntax structure. nnpfc_separate_colour_description_present_flag equal to 0 indicates that the combination of colour primaries, transfer characteristics, matrix coefficients, and scaling and offset values applied in association with the matrix coefficients for the picture resulting from the NNPF is the same as implied by the VUI parameters vui_colour_primaries, vui_transfer_characteristics, vui_matrix_coeffs, and vui_full_range_flag that are indicated or inferred for the CLVS.

nnpfc_colour_primaries has the same semantics as specified in clause 7.3 for the vui_colour_primaries syntax element, except as follows:

- nnpfc_colour_primaries specifies the colour primaries of the picture resulting from applying the NNPF specified in the SEI message, rather than the colour primaries used for the CLVS.
- When nnpfc_colour_primaries is not present in the NNPF SEI message, the value of nnpfc_colour_primaries is inferred to be equal to vui_colour_primaries.

nnpfc_transfer_characteristics has the same semantics as specified in clause 7.3 for the vui_transfer_characteristics syntax element, except as follows:

- nnpfc_transfer_characteristics specifies the transfer characteristics of the picture resulting from applying the NNPF specified in the SEI message, rather than the transfer characteristics used for the CLVS.

- When `nnpfc_transfer_characteristics` is not present in the NNPF SEI message, the value of `nnpfc_transfer_characteristics` is inferred to be equal to `vui_transfer_characteristics`.

nnpfc_matrix_coeffs describes the equations used in deriving luma and chroma signals from the green, blue, and red, or Y, Z, and X primaries. Its semantics apply to the pictures resulting from applying the NNPF specified in this SEI message and are as specified for MatrixCoefficients in Rec. ITU-T H.273 | ISO/IEC 23091-2 with `BitDepthY` and `BitDepthC` being equal to `outTensorBitDepthY` and `outTensorBitDepthC`, respectively.

When `nnpfc_matrix_coeffs` is not present in the NNPF SEI message, the value of `nnpfc_matrix_coeffs` is inferred to be equal to `vui_matrix_coeffs`.

`nnpfc_matrix_coeffs` shall not be equal to 0 unless both of the following conditions are true:

- `nnpfc_out_tensor_chroma_bitdepth_minus8` is equal to `nnpfc_out_tensor_luma_bitdepth_minus8`.
- `nnpfc_out_order_idc` is equal to 2, `outSubHeightC` is equal to 1, and `outSubWidthC` is equal to 1.

`nnpfc_matrix_coeffs` shall not be equal to 8 unless one of the following conditions is true:

- `nnpfc_out_tensor_chroma_bitdepth_minus8` is equal to `nnpfc_out_tensor_luma_bitdepth_minus8`.
- `nnpfc_out_tensor_chroma_bitdepth_minus8` is equal to `nnpfc_out_tensor_luma_bitdepth_minus8` + 1, `nnpfc_out_order_idc` is equal to 2, `outSubHeightC` is equal to 1, and `outSubWidthC` is equal to 1.

nnpfc_full_range_flag indicates the scaling and offset values applied in association with the matrix coefficients as specified by `nnpfc_matrix_coeffs`. Its semantics are as specified for the VideoFullRangeFlag parameter in Rec. ITU-T H.273 | ISO/IEC 23091-2. When not present, the value of `nnpfc_full_range_flag` is inferred to be equal to 0.

nnpfc_chroma_loc_info_present_flag equal to 1 indicates the presence of the `nnpfc_chroma_sample_loc_type_frame` syntax element in the NNPF SEI message. `nnpfc_chroma_loc_info_present_flag` equal to 0 indicates the absence of the `nnpfc_chroma_sample_loc_type_frame` syntax element in the NNPF SEI message. When `nnpfc_chroma_loc_info_present_flag` is not present, its value is inferred to be equal to 0. When `ColourizationFlag` is equal to 0 or `nnpfc_out_colour_format_idc` is not equal to 1, the value of `nnpfc_chroma_loc_info_present_flag` shall be equal to 0.

nnpfc_chroma_sample_loc_type_frame, when not equal to 6 and `nnpfc_out_colour_format_idc` is equal to 1, specifies the location of chroma samples of the output pictures, as shown in Figure 1. `nnpfc_chroma_sample_loc_type_frame` equal to 6 and `nnpfc_out_colour_format_idc` equal to 1 indicates that the location of the chroma samples is unknown or unspecified or specified by other means not specified in this Specification. The value of `nnpfc_chroma_sample_loc_type_frame` shall be in the range of 0 to 6, inclusive.

nnpfc_overlap indicates the overlapping horizontal and vertical sample counts of adjacent input tensors of the NNPF. The value of `nnpfc_overlap` shall be in the range of 0 to 16 383, inclusive.

nnpfc_constant_patch_size_flag equal to 1 indicates that the NNPF accepts exactly the patch size indicated by `nnpfc_patch_width_minus1` and `nnpfc_patch_height_minus1` as input. `nnpfc_constant_patch_size_flag` equal to 0 indicates that the NNPF accepts as input any patch size with width `inpPatchWidth` and height `inpPatchHeight` such that the width of an extended patch (i.e., a patch plus the overlapping area), which is equal to `inpPatchWidth` + 2 * `nnpfc_overlap`, is a positive integer multiple of `nnpfc_extended_patch_width_cd_delta_minus1` + 1 + 2 * `nnpfc_overlap`, and the height of the extended patch, which is equal to `inpPatchHeight` + 2 * `nnpfc_overlap`, is a positive integer multiple of `nnpfc_extended_patch_height_cd_delta_minus1` + 1 + 2 * `nnpfc_overlap`.

nnpfc_patch_width_minus1 plus 1, when `nnpfc_constant_patch_size_flag` equal to 1, indicates the horizontal sample counts of the patch size required for the input to the NNPF. The value of `nnpfc_patch_width_minus1` shall be in the range of 0 to $\text{Min}(32\ 766, \text{CroppedWidth} - 1)$, inclusive.

nnpfc_patch_height_minus1 plus 1, when `nnpfc_constant_patch_size_flag` equal to 1, indicates the vertical sample counts of the patch size required for the input to the NNPF. The value of `nnpfc_patch_height_minus1` shall be in the range of 0 to $\text{Min}(32\ 766, \text{CroppedHeight} - 1)$, inclusive.

nnpfc_extended_patch_width_cd_delta_minus1 plus 1 plus 2 * `nnpfc_overlap`, when `nnpfc_constant_patch_size_flag` equal to 0, indicates a common divisor of all allowed values of the width of an extended patch required for the input to the NNPF. The value of `nnpfc_extended_patch_width_cd_delta_minus1` shall be in the range of 0 to $\text{Min}(32\ 766, \text{CroppedWidth} - 1)$, inclusive.

nnpfc_extended_patch_height_cd_delta_minus1 plus 1 plus 2 * `nnpfc_overlap`, when `nnpfc_constant_patch_size_flag` equal to 0, indicates a common divisor of all allowed values of the height of an extended patch required for the input to the NNPF. The value of `nnpfc_extended_patch_height_cd_delta_minus1` shall be in the range of 0 to $\text{Min}(32\ 766, \text{CroppedHeight} - 1)$, inclusive.

Let the variables `inpPatchWidth` and `inpPatchHeight` be the patch size width and the patch size height, respectively.

If `nnpfc_constant_patch_size_flag` is equal to 0, the following applies:

- The values of `inpPatchWidth` and `inpPatchHeight` are either provided by external means not specified in this Specification or set by the post-processor itself.
- The value of `inpPatchWidth + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchWidth` shall be less than or equal to `CroppedWidth`. The value of `inpPatchHeight + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchHeight` shall be less than or equal to `CroppedHeight`.

Otherwise (`nnpfc_constant_patch_size_flag` is equal to 1), the value of `inpPatchWidth` is set equal to `nnpfc_patch_width_minus1 + 1` and the value of `inpPatchHeight` is set equal to `nnpfc_patch_height_minus1 + 1`.

The variables `outPatchWidth`, `outPatchHeight`, `horCScaling`, `verCScaling`, `outPatchCWidth`, and `outPatchCHeight` are derived as follows:

$$\text{outPatchWidth} = (\text{nnpfcOutputPicWidth} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (88)$$

$$\text{outPatchHeight} = (\text{nnpfcOutputPicHeight} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (89)$$

$$\text{horCScaling} = \text{SubWidthC} / \text{outSubWidthC} \quad (90)$$

$$\text{verCScaling} = \text{SubHeightC} / \text{outSubHeightC} \quad (91)$$

$$\text{outPatchCWidth} = \text{outPatchWidth} * \text{horCScaling} \quad (92)$$

$$\text{outPatchCHeight} = \text{outPatchHeight} * \text{verCScaling} \quad (93)$$

It is a requirement of bitstream conformance that `outPatchWidth * CroppedWidth` shall be equal to `nnpfcOutputPicWidth * inpPatchWidth` and `outPatchHeight * CroppedHeight` shall be equal to `nnpfcOutputPicHeight * inpPatchHeight`.

`nnpfc_padding_type` indicates the process of padding when referencing sample locations outside the boundaries of the input picture as described in Table 23. The value of `nnpfc_padding_type` shall be in the range of 0 to 15, inclusive. Values of 5 to 15, inclusive, for `nnpfc_padding_type` 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 conforming to this version of this Specification shall ignore NNPFCE SEI messages with `nnpfc_padding_type` in the range of 5 to 15, inclusive.

Table 23 – Informative description of `nnpfc_padding_type` values

<code>nnpfc_padding_type</code>	Description
0	Zero padding
1	Replication padding
2	Reflection padding
3	Wrap-around padding
4	Fixed padding
5..15	reserved

`nnpfc_luma_padding_val` indicates the luma value to be used for padding when `nnpfc_padding_type` is equal to 4. The value of `nnpfc_luma_padding_val` shall be in the range of 0 to $(1 \ll \text{BitDepth}_Y) - 1$, inclusive.

`nnpfc_cb_padding_val` indicates the Cb value to be used for padding when `nnpfc_padding_type` is equal to 4. The value of `nnpfc_cb_padding_val` shall be in the range of 0 to $(1 \ll \text{BitDepth}_C) - 1$, inclusive.

`nnpfc_cr_padding_val` indicates the Cr value to be used for padding when `nnpfc_padding_type` is equal to 4. The value of `nnpfc_cr_padding_val` shall be in the range of 0 to $(1 \ll \text{BitDepth}_C) - 1$, inclusive.

The function `InpSampleVal(y, x, picHeight, picWidth, croppedPic, cIdx)` with inputs being a vertical sample location `y`, a horizontal sample location `x`, a picture height `picHeight`, a picture width `picWidth`, sample array `croppedPic`, and component index `cIdx` (equal to 0 for luma, 1 for Cb, and 2 for Cr) returns the value of `sampleVal` derived as follows:

NOTE 7 – For the inputs to the function `InpSampleVal()`, the vertical location is listed before the horizontal location for compatibility with input tensor conventions of some inference engines.

```

if( nnpfc_padding_type == 0 )
    if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
        sampleVal = 0
    else
        sampleVal = croppedPic[ x ][ y ]
else if( nnpfc_padding_type == 1 )
    sampleVal = croppedPic[ Clip3( 0, picWidth - 1, x ) ][ Clip3( 0, picHeight - 1, y ) ]
else if( nnpfc_padding_type == 2 )
    sampleVal = croppedPic[ Reflect( picWidth - 1, x ) ][ Reflect( picHeight - 1, y ) ]
else if( nnpfc_padding_type == 3 )
    if( y >= 0 && y < picHeight )
        sampleVal = croppedPic[ Wrap( picWidth - 1, x ) ][ y ]
else if( nnpfc_padding_type == 4 )
    if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
        sampleVal = ( cIdx == 0 ? nnpfc_luma_padding_val :
            ( cIdx == 1 ? nnpfc_cb_padding_val : nnpfc_cr_padding_val ) )
    else
        sampleVal = croppedPic[ x ][ y ]

```

When `nnpfc_auxiliary_inp_idc` is equal to 1, the variable `strengthControlScaledVal` is derived as follows:

```

for( i = 0; i < numInputPics; i++ )
    if( nnpfc_inp_format_idc == 1 )
        if( nnpfc_inp_order_idc == 0 || nnpfc_inp_order_idc == 2 ||
            nnpfc_inp_order_idc == 3 )
            strengthControlScaledVal[ i ] =
                Floor ( StrengthControlVal[ i ] * ( ( 1 << inpTensorBitDepthY ) - 1 ) )
        else if( nnpfc_inp_order_idc == 1 )
            strengthControlScaledVal[ i ] =
                Floor ( StrengthControlVal[ i ] * ( ( 1 << inpTensorBitDepthC ) - 1 ) )
    else
        strengthControlScaledVal[ i ] = StrengthControlVal[ i ]

```

A patch is a rectangular array of samples from a component (e.g., a luma or chroma component) of a picture.

The process `DeriveInputTensors()`, for deriving the input tensor `inputTensor` for a given vertical sample coordinate `cTop` and a horizontal sample coordinate `cLeft` specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

for( i = 0; i < numInputPics; i++ ) {
    if( nnpfc_inp_order_idc == 0 )
        for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++ )
            for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++ ) {
                inpVal = InpY( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight,
                    CroppedWidth, CroppedYPic[ i ], 0 ) )
                yPovlp = yP + nnpfc_overlap
                xPovlp = xP + nnpfc_overlap
                if( !nnpfc_component_last_flag )
                    inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpVal
                else
                    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpVal
            }
        if( nnpfc_auxiliary_inp_idc == 1 )
            if( !nnpfc_component_last_flag )
                inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal[ i ]
            else
                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = strengthControlScaledVal[ i ]
    }

```

```

}
else if( nnpfc_inp_order_idc == 1 )
for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++) {
inpCbVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight / SubHeightC,
CroppedWidth / SubWidthC, CroppedCbPic[ i ], 1 ) )
inpCrVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight / SubHeightC,
CroppedWidth / SubWidthC, CroppedCrPic[ i ], 2 ) )
yPovlp = yP + nnpfc_overlap
xPovlp = xP + nnpfc_overlap
if( !nnpfc_component_last_flag ) {
inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpCbVal
inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpCrVal
} else {
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpCbVal
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpCrVal
}
if( nnpfc_auxiliary_inp_idc == 1 )
if( !nnpfc_component_last_flag )
inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal[ i ]
else
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] = strengthControlScaledVal[ i ]
}
}
else if( nnpfc_inp_order_idc == 2 )
for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++) {
yY = cTop + yP
xY = cLeft + xP
yC = yY / SubHeightC
xC = xY / SubWidthC
inpYVal = InpY( InpSampleVal( yY, xY, CroppedHeight,
CroppedWidth, CroppedYPic[ i ], 0 ) )
inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,
CroppedWidth / SubWidthC, CroppedCbPic[ i ], 1 ) )
inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,
CroppedWidth / SubWidthC, CroppedCrPic[ i ], 2 ) )
yPovlp = yP + nnpfc_overlap
xPovlp = xP + nnpfc_overlap
if( !nnpfc_component_last_flag ) {
inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpYVal
inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpCbVal
inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = inpCrVal
} else {
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpYVal
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpCbVal
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] = inpCrVal
}
if( nnpfc_auxiliary_inp_idc == 1 )
if( !nnpfc_component_last_flag )
inputTensor[ 0 ][ i ][ 3 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal[ i ]
else
inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 3 ] = strengthControlScaledVal[ i ]
}
}
else if( nnpfc_inp_order_idc == 3 )
for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++) {
yTL = cTop + yP * 2
xTL = cLeft + xP * 2
yBR = yTL + 1
xBR = xTL + 1
yC = cTop / 2 + yP
xC = cLeft / 2 + xP
inpTLVal = InpY( InpSampleVal( yTL, xTL, CroppedHeight,

```

```

        CroppedWidth, CroppedYPic[ i ], 0 ) )
inpTRVal = InpY( InpSampleVal( yTL, xBR, CroppedHeight,
        CroppedWidth, CroppedYPic[ i ], 0 ) )
inpBLVal = InpY( InpSampleVal( yBR, xTL, CroppedHeight,
        CroppedWidth, CroppedYPic[ i ], 0 ) )
inpBRVal = InpY( InpSampleVal( yBR, xBR, CroppedHeight,
        CroppedWidth, CroppedYPic[ i ], 0 ) )
inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,
        CroppedWidth / 2, CroppedCbPic[ i ], 1 ) )
inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,
        CroppedWidth / 2, CroppedCrPic[ i ], 2 ) )
yPovlp = yP + nnpfc_overlap
xPovlp = xP + nnpfc_overlap
if( !nnpfc_component_last_flag ) {
    inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpTLVal
    inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpTRVal
    inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = inpBLVal
    inputTensor[ 0 ][ i ][ 3 ][ yPovlp ][ xPovlp ] = inpBRVal
    inputTensor[ 0 ][ i ][ 4 ][ yPovlp ][ xPovlp ] = inpCbVal
    inputTensor[ 0 ][ i ][ 5 ][ yPovlp ][ xPovlp ] = inpCrVal
} else {
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpTLVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpTRVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] = inpBLVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 3 ] = inpBRVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 4 ] = inpCbVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 5 ] = inpCrVal
}
if( nnpfc_auxiliary_inp_idc == 1 )
    if( !nnpfc_component_last_flag )
        inputTensor[ 0 ][ i ][ 6 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal[ i ]
    else
        inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 6 ] = strengthControlScaledVal[ i ]
}
}
}

```

The process StoreOutputTensors(), for deriving sample values in the sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, for the NNPF-generated pictures, from the output tensor outputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

for( i = 0; i < numPicsInOutTensor; i++ ) {
    if( nnpfc_out_order_idc == 0 )
        for( yP = 0; yP < outPatchHeight; yP++ )
            for( xP = 0; xP < outPatchWidth; xP++ ) {
                yY = cTop * outPatchHeight / inpPatchHeight + yP
                xY = cLeft * outPatchWidth / inpPatchWidth + xP
                if( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth )
                    if( !nnpfc_component_last_flag )
                        FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
                    else
                        FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
            }
    else if( nnpfc_out_order_idc == 1 )
        for( yP = 0; yP < outPatchCHeight; yP++ )
            for( xP = 0; xP < outPatchCWidth; xP++ ) {
                xSrc = cLeft * horCScaling + xP
                ySrc = cTop * verCScaling + yP
                if( ySrc < nnpfcOutputPicHeight / outSubHeightC &&
                    xSrc < nnpfcOutputPicWidth / outSubWidthC )
                    if( !nnpfc_component_last_flag ) {
                        FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
                    }
            }
}
}

```

```

        FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
    } else {
        FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
        FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
    }
}
else if( nnpfc_out_order_idc == 2 )
for( yP = 0; yP < outPatchHeight; yP++)
for( xP = 0; xP < outPatchWidth; xP++) {
    yY = cTop * outPatchHeight / inpPatchHeight + yP
    xY = cLeft * outPatchWidth / inpPatchWidth + xP
    yC = yY / outSubHeightC
    xC = xY / outSubWidthC
    yPc = ( yP / outSubHeightC ) * outSubHeightC
    xPc = ( xP / outSubWidthC ) * outSubWidthC
    if ( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth )
        if ( !nnpfc_component_last_flag ) {
            FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
            FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 1 ][ yPc ][ xPc ]
            FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 2 ][ yPc ][ xPc ]
        } else {
            FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
            FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 1 ]
            FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 2 ]
        }
    }
}
else if( nnpfc_out_order_idc == 3 )
for( yP = 0; yP < outPatchHeight; yP++)
for( xP = 0; xP < outPatchWidth; xP++) {
    ySrc = cTop / 2 * outPatchHeight / inpPatchHeight + yP
    xSrc = cLeft / 2 * outPatchWidth / inpPatchWidth + xP
    if ( ySrc < nnpfcOutputPicHeight / 2 &&
        xSrc < nnpfcOutputPicWidth / 2 )
        if ( !nnpfc_component_last_flag ) {
            FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
            FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] = outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
            FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] = outputTensor[ 0 ][ i ][ 2 ][ yP ][ xP ]
            FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] = outputTensor[ 0 ][ i ][ 3 ][ yP ][ xP ]
            FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 4 ][ yP ][ xP ]
            FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 5 ][ yP ][ xP ]
        } else {
            FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
            FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
            FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 2 ]
            FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 3 ]
            FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 4 ]
            FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 5 ]
        }
    }
}
}
}

```

An NNPFC PostProcessingFilter() is the target NNPFC as derived in the semantics of the NNPFC SEI message. The following example process may be used, with the NNPFC PostProcessingFilter(), to generate, in a patch-wise manner, the filtered and/or interpolated picture(s), which contain Y, Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as indicated by nnpfc_out_order_idc:

```

if( nnpfc_inp_order_idc == 0 || nnpfc_inp_order_idc == 2 )
for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight )
for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth ) {
    inputTensor = DeriveInputTensors( )
    outputTensor = PostProcessingFilter( inputTensor )
    StoreOutputTensors( outputTensor )
}

```

```

    }
    else if( nnpfc_inp_order_idc == 1 )
        for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop += inpPatchHeight )
            for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft += inpPatchWidth ) {
                inputTensor = DeriveInputTensors( )
                outputTensor = PostProcessingFilter( inputTensor )
                StoreOutputTensors( outputTensor )
            }
    else if( nnpfc_inp_order_idc == 3 )
        for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight * 2 )
            for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth * 2 ) {
                inputTensor = DeriveInputTensors( )
                outputTensor = PostProcessingFilter( inputTensor )
                StoreOutputTensors( outputTensor )
            }
}

```

(98)

An NNPF-generated picture with index i contains sample arrays $\text{FilteredYPic}[i]$, $\text{FilteredCbPic}[i]$, and $\text{FilteredCrPic}[i]$, when present, that are derived by Equation 98. An NNPF-generated picture does not include the overlap regions.

The NNPF process consists of the process defined by Equation 98 followed by outputting NNPF-generated pictures in their increasing index order, where all NNPF-generated pictures that were interpolated by the NNPF are output and those NNPF-generated pictures that correspond to any input pictures to the NNPF are output as specified in the semantics of the NNPF SEI message.

nnpfc_complexity_info_present_flag equal to 1 specifies that one or more syntax elements that indicate the complexity of the NNPF associated with the **nnpfc_id** are present. **nnpfc_complexity_info_present_flag** equal to 0 specifies that no syntax elements that indicate the complexity of the NNPF associated with the **nnpfc_id** are present.

nnpfc_parameter_type_idc equal to 0 indicates that the neural network uses only integer parameters. **nnpfc_parameter_type_idc** equal to 1 indicates that the neural network may use floating point or integer parameters. **nnpfc_parameter_type_idc** equal to 2 indicates that the neural network uses only binary parameters. **nnpfc_parameter_type_idc** equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore NNPF SEI messages with **nnpfc_parameter_type_idc** equal to 3.

nnpfc_log2_parameter_bit_length_minus3 equal to 0, 1, 2, and 3 indicates that the neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively. When **nnpfc_parameter_type_idc** is present and **nnpfc_log2_parameter_bit_length_minus3** is not present, the neural network does not use parameters of bit length greater than 1.

nnpfc_num_parameters_idc indicates the maximum number of neural network parameters for the NNPF in units of a power of 2 048. **nnpfc_num_parameters_idc** equal to 0 indicates that the maximum number of neural network parameters is unknown. The value **nnpfc_num_parameters_idc** shall be in the range of 0 to 52, inclusive. Values of **nnpfc_num_parameters_idc** greater than 52 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 conforming to this version of this Specification shall ignore NNPF SEI messages with **nnpfc_num_parameters_idc** greater than 52.

If the value of **nnpfc_num_parameters_idc** is greater than zero, the variable **maxNumParameters** is derived as follows:

$$\text{maxNumParameters} = (2\ 048 \ll \text{nnpfc_num_parameters_idc}) - 1 \quad (99)$$

It is a requirement of bitstream conformance that the number of neural network parameters of the NNPF shall be less than or equal to **maxNumParameters**.

nnpfc_num_kmac_operations_idc greater than 0 indicates that the maximum number of multiply-accumulate operations per sample of the NNPF is less than or equal to **nnpfc_num_kmac_operations_idc** * 1 000. **nnpfc_num_kmac_operations_idc** equal to 0 indicates that the maximum number of multiply-accumulate operations of the network is unknown. The value of **nnpfc_num_kmac_operations_idc** shall be in the range of 0 to $2^{32} - 2$, inclusive.

nnpfc_total_kilobyte_size greater than 0 indicates a total size in kilobytes required to store the uncompressed parameters for the neural network. The total size in bits is a number equal to or greater than the sum of bits used to store each parameter. **nnpfc_total_kilobyte_size** is the total size in bits divided by 8 000, rounded up. **nnpfc_total_kilobyte_size** equal to 0 indicates that the total size required to store the parameters for the neural network is unknown. The value of **nnpfc_total_kilobyte_size** shall be in the range of 0 to $2^{32} - 2$, inclusive.

nnpfc_num_metadata_extension_bits equal to 0 specifies that **nnpfc_reserved_metadata_extension** is not present. **nnpfc_num_metadata_extension_bits** greater than 0 specifies the length, in bits, of **nnpfc_reserved_metadata_extension**.

The value of **nnpfc_num_metadata_extension_bits** shall be in the range of 0 to 2 048, inclusive. Values in the range of 1 to 2 048, inclusive, for **nnpfc_num_metadata_extension_bits** 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 conforming to this version of this Specification shall allow any value of **nnpfc_num_metadata_extension_bits** in the range of 0 to 2 048, inclusive.

nnpfc_reserved_metadata_extension shall not be present in bitstreams conforming to this version of this Specification. However, decoders conforming to this version of this Specification shall ignore the presence and value of **nnpfc_reserved_metadata_extension**. When present, the length, in bits, of **nnpfc_reserved_metadata_extension** is equal to **nnpfc_num_metadata_extension_bits**.

nnpfc_alignment_zero_bit_b shall be equal to 0.

nnpfc_payload_byte[i] contains the *i*-th byte of a bitstream conforming to ISO/IEC 15938-17. The byte sequence **nnpfc_payload_byte[i]** for all present values of *i* shall be a complete bitstream that conforms to ISO/IEC 15938-17.

8.28.3 Neural-network post-filter activation SEI message

8.28.3.1 Neural-network post-filter activation SEI message syntax

	Descriptor
nn_post_filter_activation (payloadSize) {	
nnpfa_target_id	ue(v)
nnpfa_cancel_flag	u(1)
if(! nnpfa_cancel_flag) {	
nnpfa_persistence_flag	u(1)
nnpfa_target_base_flag	u(1)
nnpfa_no_prev_clvs_flag	u(1)
if(nnpfa_persistence_flag)	
nnpfa_no_foll_clvs_flag	u(1)
nnpfa_num_output_entries	ue(v)
for(<i>i</i> = 0; <i>i</i> < nnpfa_num_output_entries ; <i>i</i> ++)	
nnpfa_output_flag[i]	u(1)
}	
}	

8.28.3.2 Neural-network post-filter activation SEI message semantics

The neural-network post-filter activation (NNPFA) SEI message activates or de-activates the possible use of the target neural-network post-processing filter (NNPF), identified by **nnpfa_target_id** and **nnpfa_target_base_flag**, for post-processing filtering of a set of pictures. For a particular picture for which the NNPF is activated, the target NNPF is derived as follows:

- If **nnpfa_target_base_flag** is equal to 1, the target NNPF is the base NNPF with **nnpfc_id** equal to **nnpfa_target_id**.
- Otherwise (**nnpfa_target_base_flag** is equal to 0), the target NNPF is the NNPF specified by the last NNPFC SEI message with **nnpfc_id** equal to **nnpfa_target_id** that 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.

NOTE 1 – There can be several NNPFA SEI messages present for the same picture, for example, when the NNPFs are meant for different purposes or for filtering of different colour components.

nnpfa_target_id indicates the **nnpfc_id** of the target NNPF, which is specified by one or more NNPFC SEI messages that pertain to the current picture and have **nnpfc_id** equal to **nnpfa_target_id**. The value of **nnpfa_target_id** shall be in the range of 0 to $2^{32} - 2$, inclusive.

An NNPFA SEI message with a particular value of **nnpfa_target_id** shall not be present in a current PU unless one or both of the following conditions are true:

- Within the current CLVS there is an NNPFC SEI message with **nnpfc_id** equal to the particular value of **nnpfa_target_id** present in a PU preceding the current PU in decoding order.

- There is an NNPFC SEI message with `nnpfc_id` equal to the particular value of `nnpfa_target_id` in the current PU.

When a PU contains both an NNPFC SEI message with a particular value of `nnpfc_id` and an NNPFA SEI message with `nnpfa_target_id` equal to the particular value of `nnpfc_id`, the NNPFC SEI message shall precede the NNPFA SEI message in decoding order.

`nnpfa_cancel_flag` equal to 1 indicates that the persistence of the target NNPFC established by any previous NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message is cancelled, i.e., the target NNPFC is no longer used unless it is activated by another NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message and `nnpfa_cancel_flag` equal to 0. `nnpfa_cancel_flag` equal to 0 indicates that the `nnpfa_persistence_flag`, `nnpfa_target_base_flag`, `nnpfa_no_prev_clvs_flag`, `nnpfa_no_foll_clvs_flag` (when `nnpfa_persistence_flag` is equal to 1), and `nnpfa_num_output_entries` follow.

`nnpfa_persistence_flag` specifies the persistence of the target NNPFC for the current layer.

`nnpfa_persistence_flag` equal to 0 specifies that the target NNPFC may be used for post-processing filtering for the current picture only.

`nnpfa_persistence_flag` equal to 1 specifies that the target NNPFC 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 an NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message is output that follows the current picture in output order.

NOTE 2 – The target NNPFC is not applied for this subsequent picture in the current layer associated with an NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message.

Let the `nnpfcTargetPictures` be the set of pictures to which the NNPFC SEI message corresponding to the target NNPFC pertains. Let `nnpfaTargetPictures` be the set of pictures for which the target NNPFC is activated by the current NNPFA SEI message. It is a requirement of bitstream conformance that any picture included in `nnpfaTargetPictures` shall also be included in `nnpfcTargetPictures`.

`nnpfa_target_base_flag` equal to 1 specifies that the target NNPFC is the base NNPFC with `nnpfc_id` equal to `nnpfa_target_id`. `nnpfa_target_base_flag` equal to 0 specifies that the target NNPFC is the NNPFC specified by the last NNPFC SEI message with `nnpfc_id` equal to `nnpfa_target_id` that 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 NNPFC.

NOTE 3 – An NNPFA message can activate a base NNPFC with a particular `nnpfc_id` value when an update of the base NNPFC is active, which switches the target NNPFC from the updated NNPFC to the base NNPFC.

When `nnpfa_target_base_flag` in an NNPFA SEI message is equal to 0, there shall be at least one NNPFC SEI message with `nnpfc_id` equal to `nnpfa_target_id` and `nnpfc_base_flag` equal to 0 that precedes the NNPFA SEI message in decoding order.

`nnpfa_no_prev_clvs_flag` equal to 1 specifies that the input pictures for the NNPFC do not originate from a previous CLVS. `nnpfa_no_prev_clvs_flag` equal to 0 specifies that the input pictures for the NNPFC 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 from one or more previous CLVSs and therefore is likely to impact the output of the target NNPFC 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 NNPFC, when the following CLVS is spliced from a different bitstream next to the current CLVS. Consequently, the NNPFC process interpolates pictures up to the end of the current CLVS using input pictures originating from the current CLVS only.

`nnpfa_num_output_entries` specifies the number of `nnpfa_output_flag[i]` syntax elements present in the NNPFA SEI message. The value of `nnpfa_num_output_entries` shall be in the range of 0 to `NumInpPicsInOutputTensor`, inclusive. When `PictureRateUpsamplingFlag` is equal to 0 and `nnpfa_num_output_entries` is equal to `NumInpPicsInOutputTensor`, `nnpfa_output_flag[i]` shall be equal to 1 for at least one value of `i` in the range of 0 to `nnpfa_num_output_entries` – 1, inclusive.

nnpfa_output_flag[i] equal to 1 specifies that the NNPf-generated picture that corresponds to the input picture having index **InpIdx**[i] is output by the NNPf process activated by this NNPFA SEI message, where the NNPf process is specified in the semantics of the NNPFC SEI message. **nnpfa_output_flag**[i] equal to 0 specifies that the NNPf-generated picture that corresponds to the input picture having index **InpIdx**[i] is not output by the NNPf process activated by this NNPFA SEI message. When **nnpfa_num_output_entries** is less than **NumInpPicsInOutTensor**, **nnpfa_output_flag**[i] is inferred to be equal to 1 for each value of i in the range of **nnpfa_num_output_entries** to **NumInpPicsInOutTensor** – 1, inclusive.

8.29 Phase indication SEI message

8.29.1 Phase indication SEI message syntax

phase_indication(payloadSize) {	Descriptor
pi_hor_phase_num	u(8)
pi_hor_phase_den_minus1	u(8)
pi_ver_phase_num	u(8)
pi_ver_phase_den_minus1	u(8)
}	

8.29.2 Phase indication SEI message semantics

The phase indication SEI message provides the decoder with information about the position of luma sampling locations in cropped decoded pictures relative to a rendering window. This information may be used by a decoder to ensure the correct spatial alignment of rendered pictures, for example when switching between picture resolutions.

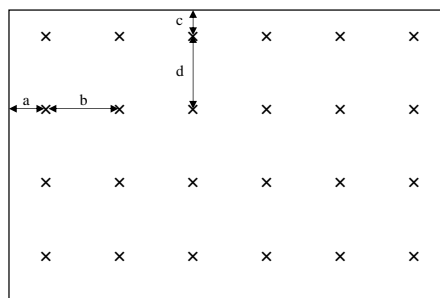


Figure 13 – The ratios $\frac{a}{b}$ and $\frac{c}{d}$ represent the horizontal and vertical locations of the luma samples (marked with x) relative to a rendering window. $\frac{a}{b}$ is equal to $\text{pi_hor_phase_num} \div (\text{pi_hor_phase_den_minus1} + 1)$, and $\frac{c}{d}$ to $\text{pi_ver_phase_num} \div (\text{pi_ver_phase_den_minus1} + 1)$

NOTE 1 – When the number of luma output samples in horizontal direction is equal to the width of the rendering window, and $\text{pi_hor_phase_num} \div (\text{pi_hor_phase_den_minus1} + 1)$ is equal to $1 \div 2$, the picture is intended to be rendered without applying any horizontal phase shift. Correspondingly, when the number of luma output samples in vertical direction is equal to the height of the rendering window, and $\text{pi_ver_phase_num} \div (\text{pi_ver_phase_den_minus1} + 1)$ is equal to $1 \div 2$, the picture is intended to be rendered without applying any vertical phase shift.

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

- The width and height of a cropped decoded picture in units of luma samples, denoted herein by **CroppedWidth** and **CroppedHeight**, respectively.

The phase indication SEI message applies to the current cropped decoded picture and persists for all subsequent pictures of the current layer in output order with the same value of **CroppedWidth** as the current picture and the same value of **CroppedHeight** as the current picture 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 with an associated phase indication SEI message and the same value of CroppedWidth as the current picture and the same value of CroppedHeight as the current picture is output and follows the current picture in output order.

pi_hor_phase_num and **pi_hor_phase_den_minus1** specify the horizontal position of luma sampling locations relative to a rendering window. The horizontal position $\text{pi_hor_phase_num} \div (\text{pi_hor_phase_den_minus1} + 1)$ is expressed in units of the horizontal distance between two horizontally adjacent luma sampling locations. The value of **pi_hor_phase_num** shall be greater than or equal to 0 and less than or equal to $\text{pi_hor_phase_den_minus1} + 1$.

pi_ver_phase_num and **pi_ver_phase_den_minus1** specify the vertical position of luma sampling locations relative to a rendering window. The vertical position $\text{pi_ver_phase_num} \div (\text{pi_ver_phase_den_minus1} + 1)$ is expressed in units of the vertical distance between two vertically adjacent luma sampling locations. The value of **pi_ver_phase_num** shall be greater than or equal to 0 and less than or equal to $\text{pi_ver_phase_den_minus1} + 1$.

NOTE 2 – The phase indicators can be used during the rendering process. For example, texture coordinates can be offset by an amount proportional to the signalled horizontal and vertical phase indicators.

NOTE 3 – The signalled phase indicators applies to the luma samples of the decoded pictures. The phase offset for chroma samples can be derived from the signalled phase indicators taking into account the chroma sample location relative to luma sample location as indicated by **ChromaFormatIdc**, **vui_chroma_sample_loc_type_frame**, **vui_chroma_sample_loc_type_top_field** and **vui_chroma_sample_loc_type_bottom_field**. When **ChromaFormatIdc** is equal to 1 (4:2:0 chroma format) and the value of **vui_chroma_sample_loc_type_frame**, **vui_chroma_sample_loc_type_top_field** and **vui_chroma_sample_loc_type_bottom_field**, as applicable, are equal to 6 or are inferred to be equal to 6, the nominal vertical and horizontal relative locations of luma and chroma samples that corresponds to **vui_chroma_sample_loc_type_frame**, **vui_chroma_sample_loc_type_top_field** and **vui_chroma_sample_loc_type_bottom_field** equal to 0 can be assumed.

8.30 Reserved SEI message

8.30.1 Reserved SEI message syntax

	Descriptor
<code>reserved_message(payloadSize) {</code>	
<code> for(i = 0; i < payloadSize; i++)</code>	
<code> reserved_message_payload_byte</code>	<code>u(8)</code>
<code>}</code>	

8.30.2 Reserved SEI message semantics

The reserved SEI message consists of data reserved for future backward-compatible use by ITU-T | ISO/IEC. Unless otherwise specified by a referencing specification, coded video bitstreams shall not contain reserved SEI messages and systems that make use of such coded video bitstreams shall not otherwise send reserved SEI messages until and unless the use of such messages has been specified by ITU-T | ISO/IEC. Decoders shall ignore reserved SEI messages.

reserved_message_payload_byte, when present, has values to be specified in the future by ITU-T | ISO/IEC.

9 Parsing process for k-th order Exp-Golomb codes

9.1 General

This process is invoked when the descriptor of a syntax element in the syntax tables is equal to `ue(v)` or `se(v)`.

Inputs to this process are bits from the bitstream.

Outputs of this process are syntax element values.

Syntax elements coded as `ue(v)` or `se(v)` are Exp-Golomb-coded with order k equal to 0. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

$$\begin{aligned}
 &\text{leadingZeroBits} = -1 \\
 &\text{for(} b = 0; !b; \text{leadingZeroBits}++ \text{)} \\
 &\quad b = \text{read_bits}(1)
 \end{aligned} \tag{100}$$

The variable `codeNum` is then assigned as follows:

$$\text{codeNum} = (2^{\text{leadingZeroBits} - 1}) * 2^k + \text{read_bits}(\text{leadingZeroBits} + k) \quad (101)$$

where the value returned from `read_bits(leadingZeroBits)` is interpreted as a binary representation of an unsigned integer with most significant bit written first.

Table 24 illustrates the structure of the 0-th order Exp-Golomb code by separating the bit string into "prefix" and "suffix" bits. The "prefix" bits are those bits that are parsed as specified for the computation of `leadingZeroBits`, and are shown as either 0 or 1 in the bit string column of Table 24. The "suffix" bits are those bits that are parsed in the computation of `codeNum` and are shown as x_i in Table 24, with i in the range of 0 to `leadingZeroBits - 1`, inclusive. Each x_i is equal to either 0 or 1.

Table 24 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1..2
0 0 1 $x_1 x_0$	3..6
0 0 0 1 $x_2 x_1 x_0$	7..14
0 0 0 0 1 $x_3 x_2 x_1 x_0$	15..30
0 0 0 0 0 1 $x_4 x_3 x_2 x_1 x_0$	31..62
...	...

Table 25 illustrates explicitly the assignment of bit strings to `codeNum` values.

Table 25 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

Depending on the descriptor, the value of a syntax element is derived as follows:

- If the syntax element is coded as `ue(v)`, the value of the syntax element is equal to `codeNum`.
- Otherwise (the syntax element is coded as `se(v)`), the value of the syntax element is derived by invoking the mapping process for signed Exp-Golomb codes as specified in clause 9.2 with `codeNum` as input.

9.2 Mapping process for signed Exp-Golomb codes

Input to this process is `codeNum` as specified in clause 9.1.

Output of this process is a value of a syntax element coded as `se(v)`.

The syntax element is assigned to the `codeNum` by ordering the syntax element by its absolute value in increasing order and representing the positive value for a given absolute value with the lower `codeNum`. Table 26 provides the assignment rule.

Table 26 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)

codeNum	syntax element value
0	0
1	1
2	-1
3	2
4	-2
5	3
6	-3
k	$(-1)^{k+1} * \text{Ceil}(k \div 2)$

Bibliography

- [1] Recommendation ITU-T H.264 (in force) | ISO/IEC 14496-10 (in force), *Advanced video coding for generic audiovisual services*.
- [2] Recommendation ITU-T H.265 (in force) | ISO/IEC 23008-2 (in force), *High efficiency video coding*.
- [3] Recommendation ITU-T H.266 (in force) | ISO/IEC 23090-3 (in force), *Versatile video coding*.
- [4] Recommendation ITU-T H.271 (in force), *Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender*.
- [5] Supplement ITU-T H-Suppl. 19 (in force) | ISO/IEC TR 23091-4 (in force), *Usage of video signal type code points*.
- [6] Recommendation ITU-R BT.1886 (in force), *Reference electro-optical transfer function for flat panel displays used in HDTV studio production*.
- [7] Recommendation ITU-R BT.2020 (in force), *Parameter values for ultra-high definition television systems for production and international programme exchange*.
- [8] Recommendation ITU-R BT.2035 (in force), *A reference viewing environment for evaluation of HDTV program material or completed programmes*.
- [9] Recommendation ITU-R BT.2100 (in force), *Image parameter values for high dynamic range television for use in production and international programme exchange*.
- [10] ANSI/CTA 861-G (in force), *A DTV Profile for Uncompressed High Speed Digital Interfaces*.
- [11] CIE 15 (in force), *Colorimetry*.
- [12] ISO/CIE 11664-3 (in force), *Colorimetry – Part 3: CIE tristimulus values*.
- [13] SMPTE RDD 5 (in force), *Film Grain Technology – Specifications for H.264/MPEG-4 AVC Bitstreams*.
- [14] SMPTE ST 2086 (in force), *Mastering Display Color Volume Metadata supporting High Luminance and Wide Color Gamut Images*.
- [15] CEA 861.3 (in force), *HDR Static Metadata Extensions*.
- [16] ATSC A/341:2022-03, *ATSC Standard: Video – HEVC*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems