INTERNATIONAL STANDARD

ISO/IEC 15444-2

First edition
2004-05-15

AMENDMENT 3
2015-07-15

Information technology — JPEG 2000 image coding system: Extensions

AMENDMENT 3: Box-based file format for JPEG XR, extended ROI boxes, XML boxing, compressed channel definition boxes, and representation of floating point

Technologies de l'information — Système de codage d'images JPEG 2000: Extensions

AMENDEMENT 3: Format des fichiers basé sur les boîtes pour JPEG-XR, boîtes de région d'intérêt (ROI) étendues, boîtes XML, boîtes de définition des canaux comprimées et représentation des données à virgule flottante

Reference number
ISO/IEC 15444-2:2004/Amd.3:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 3 to ISO/IEC 15444-2:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29 *Coding of audio, picture, multimedia and hypermedia information,* in collaboration with ITU-T. The identical text is published as ITU-T Rec. T.801 (02/2002)/Amd.3.

INTERNATIONAL STANDARD
ISO/IEC 15444-2:2004/FDAM 3(E)
Rec. ITU-T T.801 (02/2002)/Amd.3(200X E)
ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – JPEG 2000 IMAGE CODING SYSTEM: EXTENSIONS

AMENDMENT 3:

Box-based file format for JPEG XR, extended ROI boxes, XML boxing, compressed channel definition boxes, and representation of floating point

1) Clause 2 (References)

Add the following items to the list of normative references:

- ITU-T Recommendation T.805 | ISO/IEC 15444-6, *Information technology - JPEG 2000 image coding system - Part 6: Compound image file format*
- ITU-T Recommendation T.45 *Run-length Colour Encoding*
- ITU-T Recommendation T.832 | ISO/IEC 29199-2, *Information technology – JPEG XR image coding system – Image coding specification*
- IEC 60559 *Binary floating-point arithmetic for microprocessor systems*
- IEC 61966-2-2: *Multimedia systems and equipment – Colour measurement and management – Part 2-2: Colour management – Extended RGB colourspace - scRGB*
- IEEE 754: *IEEE Standard for Floating-Point Arithmetic*

2) Subclause A.3.10 Nonlinearity point transformation (NLT)

Extend the description of the Lnlt syntax element by defining Lnlt to be 6 in case Tnlt equals 0 or 3.

Lnlt: Length of marker segment in bytes (not including the marker). The value of this parameter is determined by the following equation:

$$Lnlt = 6 + \begin{cases} 0 & Tnlt = 0 \quad or \quad Tnlt = 3 \\ 15 & Tnlt = 1 \\ 11 + (N_{points} \cdot \psi_{Tval}) & Tnlt = 2 \end{cases}$$

$$\psi_{Tval} = \begin{cases} 1 & PTval \in [1, 8] \\ 2 & PTval \in [9, 16] \\ 4 & PTval \in [17, 32] \end{cases}$$

(A-7)

Change the description of the Tnlt syntax element to:

Tnlt: Non-linearity type. Table A-44 shows the value for the Tnlt parameter.

Replace Table A-44 by the following:

Value (bits) MSB LSB	Meaning of Tnlt values	STnlt usage
0000 0000	No non-linearity transformation applied	-
0000 0001	Gamma-style non-linearity transformation	Table A-45
0000 0010	LUT-style non-linearity transformation	Table A-46
0000 0011	Binary Complement to Sign Magnitude Conversion	-
	All other values reserved	-

3) Subclause A.3.13 Extended Capabilities

Replace Table A-49 in Subclause A.3.13 by the following (this is defined in AMD.2 of ITU.T 801 | 15444-2):

Parameter	Size (bits)	Value
CAP	16	0xFF50
Lcap	16	6-70
Pcap	32	Table A-50
Ccap[1]	16	Value and meaning specified in ITU.T Rec. 800+(k-1) | ISO/IEC 15444-k, where the i^{th} non-zero bit in Pcap occurs in its k^{th} most-significant bit

4) Subclause K.2 Non-linear transformation specifications

Replace K.2. by the following:

This Recommendation | International Standard allows for the non-linear transformation to be stored in three different forms. The gamma-style non-linearity form specifies the transformation through parameters to an equation, and is specified in subclause K.2.1. The LUT-style non-linearity form specifies the transformation by specifying a set of look up table pairs and is specified in subclause K.2.2. The Binary Complement to Sign Magnitude Conversion transformation takes no parameters, and converts the codestream-internal sample-representation from a binary complement representation to a sign-magnitude representation. This conversion is most suitable for representing floating point data, see subclause O.6 for further informationand its recommended use. The transformation itself is specified in subclause K.2.3.

5) Subclause K.2.3 Binary complement to sign-magnitude conversion transformation

Add a new subclause K.2.3.:

If **Tnlt** equals 3, the data is processed by a transformation that changes the representation of negative sample values. This type of transformation is most useful for representing IEC 60559 floating point data in JPEG 2000 bitstreams. The integer sample values reconstructed from the codestream are then first converted from a binary complement to a sign-magnitude representation using the transformation described below.

NOTE – This transformation is considered to map integer values to integer values, keeping the bit depth and signed-ness of the samples untouched. While not required by this Recommendation | Standard, the resulting bit-patterns are, however, typically interpreted as IEC 60559 floating point numbers by specifying a floating point sample type in the file format. This operation is not part of the JPEG 2000 decoding process, but is a matter of interpreting the reconstructed samples correctly. It is recommended to use the JPX file format defined in Annex M to annotate the data for proper interpretation. Further information on how to encode floating point samples is found in subclause O.6.

If the binary complement to sign-magnitude conversion transformation is used, the bit-depth of the input samples to this transformation shall be equal to the bit-depth of the samples generated by the transformation, and the signed-ness of the input samples shall be equal to the signed-ness of the output samples. That is, the **BDnlt** value relevant to component i shall be equal to either $\mathbf{BDcbd_i}$ if a multi-component transformation is run (see subclause A.3.6, Table A-31), or shall be equal to $\mathbf{Ssiz_i}$ (see subclause A.5.1. of ITU-T Rec. T.800 | ISO/IEC 15444-1, Table A-11) if no such transformation is used.

Let $\mathbf{Z_i}$ be the input sample value of the component i to which this transformation is to be applied, $\mathbf{b_i}$ its bit depth (i.e. $\mathbf{b_i}$=($\mathbf{Ssiz_i}$ AND 0x7f) +1 or $\mathbf{b_i}$=($\mathbf{BDcbd_i}$ AND 0x7f) + 1, and $\mathbf{Y_i}$ the output of the transformation. Then the transformation is defined as:

$$Y_i = Z_i \qquad \text{if } Z_i \geq 0 \qquad \text{or}$$

$$Y_i = \min\left(-2^{bi-1} - Z_i - 1, -1\right) \qquad \text{if } Z_i < 0$$

NOTE – This is intentionally the identity transformation for unsigned components. However, readers should be aware that if the level shifting and/or inverse decorrelation transformation of Annex G of ITU-T Rec. T.800 | ISO/IEC 15444-1 is in effect, an additional DC offset will be added to the reconstructed samples. This offset might be undesirable if the intent is to represent floating point data. To prevent this DC offset, use mechanisms from this Recommendation | International Standard such as the Multiple Component Transformation (MCC and MCO markers) from Annex J, or the Variable DC Offset described in Annex B. For signed components, the transformation maps -1 to -2^{bi-1} and -2^{bi-1} to -1. The motivation for this transformation is given in subclause O.6.

6) Subclause M.2.6 – Storage of a codestream within JPX

Append the following text to the end of M.2.6:

The JPX file format also allows for multiple codestreams to be encapsulated within one or more Multiple Codestream boxes, which contains indexing information to facilitate efficient retrieval of specific codestreams of interest, by rendering and applications.

7) Subclause M.2.8 – Support for various pixel formats

Add a new subclause M.2.8 at the end of Clause M.2:

In ITU.T 800 | ISO/IEC 15444-1, channel values consisting of reconstructed image samples or palette entries were always interpreted as signed or unsigned integers. The JPX Recommendation | International Standard extends this by also allowing the representation of fixed point or floating point data. To this end, it introduces the Pixel Format Box (see M.11.7.8) which defines how the values comprised of reconstructed image data or palette entries are to be interpreted as numerical values. The understanding in ITU-T Rec. T.800 | ISO/IEC 15444-1 is that the codestream data encodes and the palette contains signed or unsigned integer values, but this convention no longer holds in the presence of a Pixel Format Box. If this box is present, the integer channel values are re-interpreted in two stages: In the first stage, the integer values reconstructed from the codestream are represented as bit-patterns encoding integers in binary two's complement notation. In the second stage, these bit patterns are re-interpreted as either integers, IEC 60559 floating point data or fixed point data. Only after this interpretation, the samples are considered to define colour values relative to a colourspace.

NOTE – For most computer architectures, the first conversion stage is transparent and requires no additional operation, and the second stage is usually realized as "casting" operation to the target type.

A non-integer pixel format is specified as follows: The desired sample format is encoded in a Pixel Format box (see subclause M.11.7.8) which is a sub-box of the Compositing Layer Header box or the JP2 Header box. The total number of bits required to represent samples in the requested format replaces the channel depth information of integer formats. For example, IEC 60559 single precision floating point numbers require 32 bits for their representation; hence the channel depth will be 32. This information is either recorded in the Image Header box (see subclause M.11.5.1) if the channel depth is identical for all channels, or in the Bits Per Component box (see subclause M.11.5.2) if the channel depth differs across channels. Furthermore, if the channel data is created indirectly through a palette, the channel depth generated by the palette lookup process is indicated in the $\mathbf{B^i}$ field of the Palette box (see subclause I.5.3.4 of ITU-T Rec. T.800 | ISO/IEC 15444-1) which then carries the relevant information for further sample interpretation.

The information on the total number of bits is augmented by information from the Pixel Format box if it is present. In addition to the numerical representation of the samples in the channel it also refines the format by splitting the total number of bits of a sample into integer and fractional, or exponent and mantissa bits. For fixed point data, the Pixel Format box indicates the number of fractional bits, i.e. the number of bits right of the (binary) point, for floating point data it indicates the mantissa bits of the floating point format, not including any implicit (hidden) bits. The number of integer (non-fractional) or exponent bits can then be derived from the total number of bits per sample and the fractional or mantissa bits. Further information on floating point number encodings are found in IEC 60559.

Fixed point and floating point samples are mapped to device colours by means of the Colour Specification box (see subclause M.11.7.2) in the same way as integer samples are mapped to device colours. For that, the Channel Definition box (see subclause M.11.7.5) defines which channels are used to generate which device colours. This process differs for non-integer samples from integer samples only in so far as the maximum intensity of the device colour is no longer represented by the maximum possible integer channel value, but as the value 1.0 expressed in the corresponding fixed point or floating point format. The handling of sample values outside of this range is implementation specific. For further information on the mapping from channel values to device colours, read subclause M.11.7.2.

8) Subclause M.2.9 – Support for JPEG XR codestreams

Add subclause M.2.9 at the end of Clause M.2:

This Recommendation | International Standard also allows the encapsulation of codestreams of other image compression Recommendations | Standards, such as JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2); the JPX file format can represent all metadata encoded in the TIFF-based file format of JPEG XR in boxes defined in this Recommendation | International Standard. A recommendation of how the JPEG XR TIFF-based container should be mapped into the JPX file format is found in subclause O.5.

The File Type Box of an ITU.T Rec. 802 | ISO/IEC 15444-2 file containing an ITU-T Rec. T.832 | ISO/IEC 29199-2 compliant codestream shall have the following values in the compatibility list **CL**[i] (see ITU-T Rec. T.800 | ISO/IEC 15444-1, subclause I.5.2 and subclause M.11 in this Recommendation | International Standard) if JPEG XR codestreams are present in the file:

Table M-1: Brand Values for JPEG XR Codestreams

Value	Meaning
'jxrc'	JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream is present
'jxr0'	JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) sub-baseline profile bitstream present and the file conforms to the JPEG XR sub-baseline profile defined in M.9.2.
'jxr1'	JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) baseline profile bitstream present and the file conforms to the JPEG XR baseline profile defined in M.9.2.
'jxr2'	JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) main profile bitstream present and the file conforms to the JPEG XR main profile defined in M.9.2.
'jxr3'	JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-3) advanced profile bitstream and the file conforms to the JPEG XR advanced profile defined in M.9.2.

NOTE - Brand values ‘jxr0’ to ‘jxr3’ indicate JPEG XR profiles of this Recommendation | International Standard. The corresponding profiles are defined in Annex M.9.2.

Relabel tables M-1 through M-24 and all references thereto to M-2 through M-25.

9) Subclause M.5.2 – Sharing header and metadata information between codestreams and compositing layers

Replace first paragraph:

To minimize file overhead, it is useful to allow header and metadata information to be shared between codestreams and compositing layers where that information is identical. The JPX file format provides three mechanisms to share information: default headers, cross-references and label associations.

with the following:

To minimize file overhead, it is useful to allow header and metadata information to be shared between codestreams and compositing layers where that information is identical. The JPX file format provides four mechanisms to share information: default headers, compositing layer extensions, cross-references and label associations.

Relabel subclause M.5.2.3 and all references thereto as M.5.2.4

Relabel subclause M.5.2.2 and all references thereto as M.5.2.3

Insert new subclause M.2.2.2 as follows:

M.5.2.2 Compositing layer extensions

The Compositing Layer Extensions box can be used to specify a repeating pattern of compositing layer headers and (optionally) codestream headers.

A file which contains a Compositing Layer Extensions boxes can be meaningfully interpreted by readers which do not understand the Compositing Layer Extensions box, because all top-level Codestream Header boxes and Compositing Layer Header boxes shall precede any Compositing Layer Extensions box and all compositing instructions found within a Compositions box shall refer only to top-level compositing layers. The Compositing Layer Extensions box may be used only when there are a well-defined number of top-level codestream headers and top-level compositing layers, which means that at least one Codestream Header box and at least one Compositing Layer Header box must appear at the top-level of the file.

A Compositing Layer Extensions box defines one or more additional compositing layers, zero or more additional codestream headers and zero or more additional compositing instructions, to augment the information provided by top-level Codestream Header, Compositing Layer Header and Composition boxes.

Each Compositing Layer Extensions box has an associated repetition factor **Mjclx**. The Compositing Layer Extensions box implicitly defines **Mjclx** × **Cjclx** additional codestream headers and **Mjclx** × **Ljclx** additional compositing layers, where **Cjclx** and **Ljclx** are the number of Codestream Header boxes and the number of Compositing Layer Header boxes that are found within the Compositing Layer Extensions box. The additional codestream headers and compositing layers are assigned consecutive indices, starting from the number of codestream headers and compositing layers defined by all top-level Codestream Header boxes and Compositing Layer Header boxes, together with all preceding Compositing Layer Extensions boxes. The codestreams that are associated with the additional compositing layers are determined from the information found in each embedded Compositing Layer Header box, using a codestream index remapping procedure that accounts for the repetition index. A similar remapping procedure is applied to the Number List boxes which may be found within a Compositing Layer Extensions box, so that embedded metadata is correctly associated.

The principle purpose of Compositing Layer Extensions is to facilitate the efficient description of a large number of compositing layers that follow a simple repeating pattern. This can be particularly beneficial if the JPX file is communicated incrementally via the tools and methods described in IS15444-9.

Compositing Layer Extensions also provide a mechanism for extending the single animation described by a Compositions box into multiple alternate presentation threads – see subclause M.5.3.

10) Subclause M.5.3 – Composition

Replace first paragraph:

Composition data is divided into fixed options, contained in the Composition Options box (subclause M.11.10.1), and a sequence of instructions contained in one or more Instruction Set boxes (subclause M.11.10.2) boxes. Each instruction comprises a set of render parameters. Each instruction set has an associated repeat count which allows for the efficient representation of long sequences of repeating instructions such as occur in full motion sequences or in slide shows which

use a repeated frame transition animation. A JPX file reader shall display a JPX file by reading and executing the instructions in sequence order, from each instruction set in sequence order and repeated according to its repeat value. The file is considered fully rendered either when there are no more instructions to execute, or no compositing layer is present for the current instruction.

with:

Composition data is divided into fixed options, contained in the Composition Options box (subclause M.11.10.1), and a sequence of instructions contained in one or more Instruction Set boxes (subclause M.11.10.2) boxes. Instructions may be further divided into those which appear within a top-level Composition box and those which appear within Compositing Layer Extensions box. The former constitute the primary presentation, while the latter constitute supplementary presentation threads.

Each instruction comprises a set of render parameters. Each instruction set has an associated repeat count which allows for the efficient representation of long sequences of repeating instructions such as occur in full motion sequences or in slide shows which use a repeated frame transition animation. A JPX file reader shall display a JPX file by reading and executing the instructions in sequence order, from each instruction set in sequence order and repeated according to its repeat value. The file is considered fully rendered either when there are no more instructions to execute, or no compositing layer is present for the current instruction.

Instructions found within the top-level Composition box are applied only to top-level compositing layers (i.e., compositing layers other than those defined by Compositing Layer Extensions boxes). Instructions found within a Compositing Layer Extensions box apply only to the compositing layers defined by that box.

11) Subclause M.9.2 – Support for JPX feature set boxes

Replace the entire subclause M.9.2 by the following:

In general, a JPX reader is not required to support the entire set of features defined within this Recommendation | International Standard. However, to promote interoperability, five profiles are defined, of which the first defines a set of baseline features required to decode images using codestream representations conforming to ITU.T 800 | ISO/IEC 15444-1 and ITU.T 801 | ISO/IEC 15444-2 only, and four additional profiles describing images containing only codestreams conforming to ITU.T 832 | ISO/IEC 29199-2.
The ITU.T 80x | ISO/IEC 15444-x based profile is denoted **JPX Baseline** in the following; Files that are written in such a way as to allow a reader that supports only this JPX baseline set of features to properly open the file shall contain a CLi field in the File Type box with the value ‘jpxb’ (0x6a70 7862); all JPX baseline readers are required to properly support all files with this code in the compatibility list in the File Type box. The definition of a JPX baseline file given in Annex M.9.2.1 through M.9.2.9, the JPEG XR profiles based on 29199-2 codestreams are defined in Annex M.9.2.10 and following:

12) Subclause M.9.2.10 – JPEG XR Profiles

Insert the following subclauses as Annex M.9.2.10 and following:

In addition to codestreams conforming to the ITU.T 80x | 15444-x series of Recommendations | Standards, a JPX file may also include codestreams conforming to 29199-2 (JPEG XR), and four profiles are defined in the following closely mirroring the profiles of 29199-2. The four profiles are denoted **JPEG XR Sub-baseline Profile**, **JPEG XR Baseline Profile**, **JPEG XR Main Profile**, and **JPEG XR Advanced Profile.** All profiles have in common that files conforming to these profiles shall only contain codestreams conforming to 29199-2.

Files conforming to the JPEG XR profiles shall contain a CL^{i} field in the File Type box with the values ‘jxr0’ through ‘jxr3’, according to the profiles the corresponding 29199-2 codestreams conform to; these compatibilities are defined in Table M-1.

13) Subclause M.9.2.11 – Compression Type

Readers conforming to one of the four JPEG XR profiles only need to support the compression type C = 11 (JPEG XR) indicated in the Image Header Box; see Table M-20 for all compression types defined in this Recommendation | International Standard. Support for other compression types shall not be required to display a file conforming to one of the four JPEG XR profiles.

14) Subclause M.9.2.12 – Compositing Layers

Support for multiple compositing layers is not required to properly display the file; however, the main file may contain multiple compositing layers, but if so, only the first one need to be rendered by an implementation conforming to the JPEG XR profiles. Compositing layers may consist of one or two codestreams that both shall conform to 29199-2. If a compositing layer consists of two codestreams, the two codestreams shall describe images of the same size that are aligned pixel by pixel, and the second codestream shall consist of a single component representing the opacity of the samples encoded in the first codestream. If a second codestream is present in a compositing layer, the first codestream shall not include any opacity information.

15) Subclause M.9.2.13 – Colour Specification

The first composting layer shall contain at least one Color Specification Box from the following list:

- The enumerated method EnumCS value indicating either sRGB, scRGB, sRGB-grey, scRGB-grey, bi-level black on white or bi-level white on black for the JPEG XR baseline and JPEG XR sub-baseline profiles.
- In addition to the above, the enumerated method EnumCS value indicating CMYK or the Any ICC method for the JPEG XR main profile.
- In addition to the above, the enumerated method EnumCS value indicating YCbCr(1) through YCbCr(3) for the JPEG XR advanced profile.

16) Subclause M.9.2.14 – Codestream Fragmentation

The codestreams representing the data of the first compositing layer of files conforming to the JPEG XR profiles shall not be fragmented.

17) Subclause M.9.2.15 – Cross Reference Boxes

Files conforming to the JPEG XR profiles shall not use Cross Reference Boxes for replacing boxes necessary to decode the first compositing layer.

18) Subclause M.9.2.16 – JP2 Header Box Location

The JP2 Header box shall be found in the file before the first Contiguous Codestream box, and Compositing Layer Header box. Any information contained within the JP2 Header box shall be applied to the codestreams encoding the first compositing layer, and as well being used as default information for all other compositing layers; the boxes within the JP2 Header box shall not be found within the Compositing Layer Header box or the Codestream Header box associated with the first compositing layer.

19) Subclause M.9.2.17 – Opacity

A JPEG XR profile conforming reader shall properly interpret opacity channels, through either direct mapping to a codestream component using the Channel Definition box. Other means of indicating opacity, e.g. by the Opacity box, need not to be supported. The use of opacity outside of compositing layers within the JPX file indicates that the decoded image data shall be composited onto an application defined background.

20) Subclause M.9.2.18 – Rotation

A JPEG XR conforming reader shall properly interpret the ROT field of the Instruction Set box defined in Annex M.11.10.2 and Annex M.11.10.2.1. Other instructions defined in the Instruction Set box need not to be honored for compliance to the JPEG XR profiles.

21) Subclause M.9.2.19 – Other Data in the File

A JPEG XR profile file may contain other features or metadata, provided they do not modify the visual appearance of the still image as viewed using a reader that supports only the JPEG XR feature set. All JPX readers should be aware of the existence of this data, as parsing or processing this data may be required in some extended applications. Applications that understand other data or features in the file are encouraged to support the behaviors and functions associated with that extended data.

22) Subclause M.9.2.20 – Conformance Testing

A conformance testing procedure for the JPEG XR profiles as well as test files suitable for conformance testing are defined in ITU.T 834 | ISO/IEC 29199-4.

23) Subclause M.11 – Defined boxes

Edit Table M-6 (now Table M-7) as follows:

- *Add the Pixel Format Box as sub-box to the JP2 Header Box and the Compositing Layer Header Box and add a reference to M.11.7.8.*
- *Add the Compositing Layer Extensions box as a top-level box and add a reference to subclause M.11.21.*
- *Add the Compositing Layer Extensions Info box as a sub-box to the Compositing Layer Extensions box and add a reference to subclause M.11.22.*
- *Add the Multiple Codestream as a top-level box and add a reference to subclause M.11.23. Add a codestream box with reference to M.11.8 and a Fragment Table box with reference to M.11.3.*
- *Add the Multiple Codestream Info box as sub-box to the Multiple Codestream box and add a reference to subclause M.11.24.*
- *Add the Grouping box as a top-level box and add a reference to subclause M.11.25. Add a sub-box labeled "…" to this grouping box.*
- *Add a top-level asoc box to Table M-6 (now Table M-7) with a reference to M.11.11, add a Decomposed XML box as its first sub-box and add a reference to subclause M.11.2.26. Add a second asoc box with reference to M.11.11, and add a XML header box and a reference to subclause M.11.27 as its first sub-box, and a XML box with reference to M.11.18 as its second sub-box.*

24) Table M-13 – Boxes defined within this Recommendation | International Standard

Add the following rows to the table M-13 (now M-14):

Pixel Format box (M.11.7.8)	'pxfm' (0x7078 666d)	NO	This box specifies the interpretation of reconstructed sample values as integer, fixed point or floating point numbers.
XML box (M.11.18)	'xml\040' (0x786D 6C20)	NO	This box contains XML formatted information.
Compositing Layer Extensions box (M.11.21)	'jclx' (0x6A63 6C78)	NO	This box defines an extended set of compositing layers, codestream headers

			and compositing instructions.
Compositing Layer Extensions Info box (M.11.22)	'jlxi' (0x6A6C7869)	NO	This box provides information concerning the repetition factor, compositing layer indices and other attributes of the compositing layers and compositing instructions found within the Compositing Layer Extensions box.
Multiple Codestream box (M.11.23)	'j2cx' (0x6A32 6378)	NO	This box represents a concatenated collection of one or more contiguous codestream boxes or fragment table boxes.
Multiple Codestream Info box (M.11.24)	'j2ci' (0x6A32 6369)	NO	This box contains information describing the Multiple Codestream box in which it is found.
Grouping box (M.11.25)	'grp\040' (0x6772 7020)	NO	This superbox is a container (or wrapper) for any number of boxes which might otherwise be found as the non-initial sub-box of an Association box.
Decomposed XML box (M.11.26)	'dxml' (0x786D 6C64)	NO	This box provides provides front-matter from an XML document as part of a mechanism for decomposing a single XML document into a hierarchical collection of Association boxes.
XML Header box (M.11.27)	'hxml' (0x786D 6C68)	NO	This box provides an element header (the opening element tag with attributes) as part of a mechanism for decomposing a single XML document into a hierarchical collection of Association boxes.

25) Subclause M.11.1 – Reader Requirements box

Add the following rows to Table M-14 (now M-15):

Value	Meaning
75	Codestream contains a JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream.
76	Codestream contains a Sub-baseline profile JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream.
77	Codestream contains a Baseline profile JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream.
78	Codestream contains a Main profile JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream.
79	Codestream contains an Advanced profile JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) compliant bitstream.
80	Pixel format "Fixed Point" is used.
81	Pixel format "Floating Point" is used.

82	Pixel Formats "Mantissa" or "Exponent" are used.
83	Compositing layer uses IEC 61966-2-2 (scRGB) enumerated colourspace
84	Block Coder Extensions (see AMD.4 of ITU.T 801 | ISO/IEC 15444-2)
85	Compositing layer uses scRGB gray scale (IEC 61966-2-2 based) enumerated colourspace

26) Subclause M.11.3–Fragment Table box

Replace the first paragraph:

A Fragment Table box specifies the location of one of the codestreams in a JPX file. A file may contain zero or more Fragment Table boxes. For the purpose of numbering codestreams, the Fragment Table box shall be considered equivalent to a Contiguous Codestream box. Fragment Table boxes shall be found only at the top level of the file; they shall not be found within a superbox.

With the following:

A Fragment Table box specifies the location of one of the codestreams in a JPX file. A file may contain zero or more Fragment Table boxes. For the purpose of numbering codestreams, the Fragment Table box shall be considered equivalent to a Contiguous Codestream box. Fragment Table boxes shall be found only at the top level of the file or within Multiple Codestream boxes; they shall not be found within any other superboxes.

27) Subclause M.11.5.1– Image Header box

Replace paragraph 6, the description of the BPC field of the Image Header box, by the following:

BPC: Bits per component. This parameter identifies the bit depths and signed/unsigned characteristics of the fully decompressed components, stored as a 1-byte field as defined in Table M-21. If the components vary in bit depth or signed/unsigned characterisics, then the value of this field shall be 255, and the superbox that contains this Image Header box (either the JP2 Header box, a Codestream Header box or Compositing Layer Extension Box) shall contain a Bits Per Component box defining the bit depth and signed/unsigned characteristics of each component. If all components have the same bit depth and signed/unsigned characteristics, the BPC parameter identifies the bit depth and signed/unsigned characteristics for all components and has the same interpretation to the BPC^i parameters, as explained in connection with the Bits Per Component box in subclause M.11.5.2.

Add the following row to table M-19 (now M-20), the definition of the compression type field C of the Image Header box:

10	ITU-T Rec. T.45 (run length coding, used in ITU-T Rec. T.805 | ISO/IEC 15444-6)
11	JPEG XR (as defined by ITU-T Rec. T.832 | ISO/IEC 29199-2)

Replace Table M-20 (now M-21), the description of the BPC field of the Image Header box by the following:

Table M-22 -- BPC and BPC^i parameters

Values (bits) MSB LSB	Component Sample Format and Sample Precision
x000 0000 – x010 0101	Component bit depth = value + 1. From 1 bit deep through 38 bits deep respectively (counting the sign bit, if appropriate)

0xxx xxxx	Components are unsigned
1xxx xxxx	Components are signed
1111 1111	Component bit depths vary (Bits Per Component Box only)
all other values	Reserved for future use

Replace in Table M-21(now Table M-22) the row describing the C field

C	8	7

by the following:

C	8	Compression Type, see Table M-20

28) Subclause M.11.5.2 – Bits Per Component box

Replace the definition of the Bits Per Component box by the following:

The Bits Per Component box specifies the bit depth and signed/unsigned characteristics of each fully decompressed component, using a 1-byte field for each component, as defined in Table M-20. This box is optional and is only required in case the bit depths varies between components.

This box encodes the number of bits required to represent the component sample values reconstructed from the codestream, and the value shall match the bits per component specification in the respective codestream format specification. The structure of this box is identical to that defined in subclause I.5.3.2 in the JP2 file format specified in ITU-T Rec. T.800 | ISO/IEC 15444-1.

NOTE - For codestreams conforming to ITU-T Rec. T.80x | ISO/IEC 15444-x (JPEG 2000) this is the bit depths **after** any inverse multiple component transformation or reverse non-linearity transformation extension has been applied to components in the codestream. In case an extended ITU-T Rec. T.801 | ISO/IEC 15444-2 multiple component transformation is used, the component bit depths does **NOT** necessarily match the data in the **SIZ** marker of the codestream. For fixed point and floating point pixel formats, the numerical interpretation of the component sample values depends not only upon the bit depth and signed/unsigned characteristics identified by BPC or BPC^i, but also on the number of fraction bits or mantissa bits, as supplied via the Pixel Format box. The Bits Per Component box then only specifies the total number of bits required to represent the data and whether the data is signed or unsigned.

29) Subclause M.11.6 – Codestream Header box

Replace first three paragraphs:

The Codestream Header box specifies header and metadata information specific to a particular codestream contained within the JPX file in order to create a set of channels. All Codestream Header boxes shall be located at the top-level of the file (not within any superbox).

Both codestreams and Codestream Header boxes are numbered separately, starting with 0, by their order in the file. Codestream Header box *i* shall be applied to codestream *i*. There shall either be one Codestream Header box in the file for each codestream, or there shall be zero Codestream Header boxes in the file. In the event that there are zero

Codestream Header boxes, then the header information for all of the codestreams shall be taken to be the default header information contained within the JP2 Header box.

For the codestreams, the numbering shall consider both Contiguous Codestream boxes and Fragment Table boxes. For example, if a file contains 2 Contiguous Codestream boxes, followed by a Fragment Table box, followed by another Contiguous Codestream box, the JPX file contains 4 codestreams, where the codestreams contained directly in the first two Contiguous Codestream boxes are numbered 0 and 1, the codestream pointed to by the Fragment Table box is numbered 2, and the codestream contained within the last Contiguous Codestream box is numbered 3.

with the following:

The Codestream Header box specifies header and metadata information for a codestream contained within the JPX file in order to create a set of channels. All Codestream Header boxes shall be located either at the top-level of the file (not within any superbox) or within a Compositing Layer Extensions box (see M.11.21). All top-level Codestream Header boxes must precede any Compositing Layer Extensions boxes within the file.

Both codestreams and codestream headers are numbered separately, starting with 0. Codestream header *i* provides header information for codestream *i*. Each top-level Codestream Header box corresponds to exactly one codestream header, meaning that box *i* specifies header information for codestream *i*, starting from *i*=0. Each Codestream Header box found within a Compositing Layer Extensions box corresponds to **Mjclx** additional codestream headers (for **Mjclx** additional codestreams), where **Mjclx** is the repetition factor associated with the Compositing Layer Extensions box. The determination of indices for these additional codestream headers is described in subclause M.11.21.

If Codestream Header boxes appear anywhere in the file, the number of codestreams found in the file shall be the same as the number of available codestream headers. In the event that there are no Codestream Header boxes, then the header information for all of the codestreams shall be taken to be the default header information contained within the JP2 Header box.

For the codestreams, the numbering shall consider Contiguous Codestream boxes, Fragment Table boxes and Multiple Codestream boxes. For example, if a file contains two Contiguous Codestream boxes, followed by a Fragment Table box, followed by another Contiguous Codestream box and two Multiple Codestream boxes, each with two codestreams, the JPX file contains eight codestreams, where the codestreams contained directly in the first two Contiguous Codestream boxes are numbered 0 and 1, the codestream pointed to by the Fragment Table box is numbered 2, the codestream contained within the last Contiguous Codestream box is numbered 3, and the codestreams contained within the first (respectively second) Multiple Codestream box are numbered 4 and 5 (respectively 6 and 7).

30) Subclause M.11.7 – Compositing Layer Header box

Replace first paragraph:

The Compositing Layer Header box specifies header and metadata information specific to a particular compositing layer in the JPX file. Compositing layers are numbered, starting at 0, by the order in the file of the Compositing Layer Header boxes (box *i* specifies header information for compositing layer *i*). There shall be one Compositing Layer Header box in the file for each layer. All Compositing Layer Header boxes shall be located at the top-level of the file (not within any superbox).

with the following:

The Compositing Layer Header box specifies header and metadata information for a compositing layer in the JPX file. All Compositing Layer Header boxes shall be located either at the top-level of the file (not within any superbox) or within a Compositing Layer Extensions box (see M.11.21). All top-level Compositing Layer Header boxes must precede any Compositing Layer Extensions boxes within the file.

Each top-level Compositing Layer Header box corresponds to exactly one compositing layer, meaning that box *i* specifies header information for layer *i*, where layers are numbered starting from *i*=0. Each Compositing Layer Header box found within a Compositing Layer Extensions box corresponds to **Mjclx** additional compositing layers, where **Mjclx** is the repetition factor associated with the Compositing Layer Extensions box. The indexing of these additional compositing layers is described in subclause M.11.21.

Add a new paragraph in the list of allowable sub-boxes of the compositing layer header box below the resolution box:

pxfm: Pixel Format Box. This box defines the interpretation of the values in a channel as either integer, floating point or fixed point values. In the absence of this optional box, the bit patterns shall be interpreted signed or unsigned integers whose bit depths are either defined by the Bits Per Component Box, or the Image Header Box, or the Palette Box. The Pixel Format Box extends the channel description of the Channel Definition Box, the Image Header Box and the Bits Per Component Box (if present). The structure of the Pixel Format Box is specified in subclause M.11.7.8.

31) Subclause M.11.7.2 – Colour Specification box

Change the first paragraph of M.11.7.2 to the following:

Each Colour Specification box defines one method by which an application can interpret the colourspace of the decompressed image data. This colour specification is to be applied to the channel values interpreted according to the Pixel Format Box (see subclause M.11.7.8) and associated to colours according to the Channel Definition Box (see subclause M.11.7.5). This association is to be interpreted using the value $\mathbf{MAX}^i$ for each colour channel i, as given by Table M-26, in combination with the relevant colourspace definition.

In Table M-26, $\mathbf{BPC}^i$ is used to denote the value of the $\mathbf{B}^j$ field of the Palette Box (see subclause L.5.3.4 of ITU-T Rec. T.800 | ISO/IEC 15444-1) if channel i is the output of a palette column j, or the value of the Bits Per Component Box $\mathbf{BPC}^j$ if channel i is the direct output of component j, or the value of the **BPC** field of the Image Header Box if no Bits Per Component Box is present. The pixel format $\mathbf{F}^i$ in this table is either the format indicated in the Pixel Format Box (see Annex M.11.7.8) if it is present, or shall be signed or unsigned integer in the absence of the Pixel Format Box.

If the colourspace is defined by an ICC profile, the input channels should carry unsigned values; usage of signed samples is discouraged and currently not defined by the ICC. The values x^i for channel i, interpreted according to the Pixel Format Box, shall be mapped to device colour values d^i, as follows.

$$d^i = Dmax^i * x^i / MAX^i$$

Here, $Dmax^i$ is the maximum input value associated with the relevant ICC tone reproduction curve and MAX^i depends on the pixel format $\mathbf{F}^i$ as given by Table M-26..

For enumuerated colourspaces for which the format of the EP field is specified, the mapping from channel values x^i to device colour values d^i is defined in the corresponding definition of the EP field, see subclause M.11.7.4.

NOTE - Currently, only the CIELab and CIEJab enumerated colourspaces define a format for the EP field of the Colour Specification box.

For all other colourspaces, if the values x^i for channel i, are unsigned quantities, they shall be mapped to colour values d^i according to

$$d^i = Dmin^i + (Dmax^i - Dmin^i) * x^i / MAX^i,$$

for the purpose of establishing a correct interpretation with respect to the colourspace. Here, $Dmin^i$ and $Dmax^i$ are the minimum and maximum allowed values for the relevant colour channel, in the numerical framework used to define the colourspace. If, however, the values x^i for channel i, are signed quantities, they shall be mapped to colour values d^i according to

$$d^i = Dzero^i + (Dmax^i - Dzero^i) * x^i / MAX^i,$$

for the purpose of establishing a correct interpretation with respect to the colourspace. Here $Dmax^i$ is again the maximum allowed value for the relevant colour, in the numerical framework used to define the colourspace, while $Dzero^i$ is the value of channel i in the representation of the colour that corresponds to the absence of any scene radiance,the complete absorption of visible light or the archromatic level, if this interpretation is applicable and all channel values are uniquely defined in this case. Otherwise, the pixel format $\mathbf{F}^i$ and the channel precision $\mathbf{BPC}^i$ of all channels **i** shall be selected such that they match the numerical framework used to define the colourspace and scaling does not take place.

NOTE - The CIELab and CIEJab colourspaces uses both positive and negative values to represent colours. However, since subclause M.11.7.4 defines EP fields that fix the interpretation and scaling of channel values to device colour values, the scaling procedure defined in this subclause will not take plane. Instead, subclause M.11.7.4 defines the required procedure. As a second example, the YCbCr(2) colourspace provides unique sample values for the absence of light, namely (0,128,128), and hence $\mathbf{Dzero}^0$=0, and $\mathbf{Dzero}^1$=$\mathbf{Dzero}^2$=128. Thus, when storing signed instead of unsigned values in the codestream, the chroma components are encoded without an offset and stored as unbiased signed values.

Insert the following table at the end of subclause M.11.7.2 and relabel all following tables M-25 through M-30 and references thereto to M-27 through M-32:

Table M-26: Nominal maximum sample values

Pixel Format $\mathbf{F}^i$ and Channel Signedness	$\mathbf{MAX}^i$
unsigned, integer ($\mathbf{BPC}^i < 128$)	$2^{\mathbf{BPC}+1} - 1$
signed, integer ($\mathbf{BPC}^i \geq 128$)	$2^{\mathbf{BPC}-128} - 1$
unsigned, fixed point ($\mathbf{BPC}^i < 128$)	1.0
signed, fixed point ($\mathbf{BPC}^i \geq 128$)	1.0
unsigned, floating point ($\mathbf{BPC}^i < 128$)	1.0
signed, floating point ($\mathbf{BPC}^i \geq 128$)	1.0

NOTE – It is colourspacecolourspace dependent whether whether the full range of available samples is meaningful. Specifically, the range of meaningful inputs for signed data might be non-symmetric. Both fixed point and floating point pixel formats allow the representation of data that is out of range. For most applications, clipping such values into range may be an appropriate strategy to handle them when converting to other formats.

NOTE – Fixed point values can be interpreted as integer values after scaling by $2^{\mathbf{Qi}}$, where the value of $\mathbf{Q}^i$ is given by the low 12 bits of the $\mathbf{F}^i$ field of the Pixel Format box. The fixed point value 1.0 given in Table M-26 is thus represented by an integer value of $2^{\mathbf{Qi}}$.

Add the following entries to Table M-25 (now M-27), listing enumerated colourspaces:

Value	Meaning
25	scRGB as defined by IEC 61966-2-2
26	scRGB gray scale, using only a luminance channel but the tone reproduction curves (non-linearities) defined by IEC 61966-2-2.

32) Subclause M.11.7.4.1 – EP field format for the CIELab colourspace

Replace the second paragraph of subclause M.11.7.4.1 and formula M.18 by the following:

The RL, OL, RA, OA, RB and OB fields describe how to convert between the unsigned values N_L, N_a, N_b, as defined by ITU-T Rec. T.42, that are sent to the compressor or received from the decompressor and the signed CIELab values L*, a*, b* as defined by the CIE. According to ITU-T Rec. T.42, the calculations from real values L*a*b* to values encoded in an integer, fixed point or floating point format, which are expressed by $N_L N_a N_b$, are made as follows:

$$N_L = \frac{MAX_L}{RL} \times L^* + \frac{2^{\lceil \log_2 MAX_L \rceil}}{2^{(BPC_L \ AND \ 0x7F)+1}} \times OL$$
$$N_a = \frac{MAX_a}{RA} \times a^* + \frac{2^{\lceil \log_2 MAX_a \rceil}}{2^{(BPC_a \ AND \ 0x7F)+1}} \times OA \qquad \text{(M-18)}$$
$$N_b = \frac{MAX_b}{RB} \times b^* + \frac{2^{\lceil \log_2 MAX_b \rceil}}{2^{(BPC_b \ AND \ 0x7F)+1}} \times OB$$

where MAX_L, MAX_a and MAX_b are the nominal maximum sample values according to table M-26 for the channels carrying the L, a^* and b^* data and BPC_x are the bits per component values for channel x as found in the Image Header Box, the Bits Per Component Box or the Palette Box. The brackets •• indicate rounding up to the next integer. The numerical values of MAX_L, MAX_a and MAX_b depend on the Image Header box, the Bits Per Component Box, the Palette Box and the Pixel Format Box as explained in subclause M.11.7.2.

Replace the description of the individual fields of the EP field below equation M-18 by the following:

RL: Range for L*. This field specifies the *RL* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

OL: Offset for L*. This field specifies the *OL* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

RA: Range for a*. This field specifies the *RA* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

OA: Offset for a*. This field specifies the *OA* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

RB: Range for b*. This field specifies the *RB* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

OB: Offset for b*. This field specifies the *OB* value from Equation M-18. It is encoded as a 4-byte big endian unsigned integer.

IL: Illuminant. This field specifies the illuminant data used in calculating the CIELab values. Rather than specify the XYZ values of the normalizing illuminant, which are used in calculating CIELab, the specification of the illuminant data follows ITU-T Rec. T.4 Annex E. The illuminant data consists of 4 bytes, identifying the illuminant. In the case of a standard illuminant, the 4 bytes are one of the following:

Replace the two last paragraphs of subclause M.11.7.4.1 by the following and insert Table M-33. Relable table M-31 to table M-34:

When the EP fields are omitted for the CIELab colourspace, then the following default values shall be used. The default L*, a* and b* range parameters RL, RA and RB are 100, 170 and 200. The default L*, a* and b* offset values depend on the contents of the Pixel Format box, if present, and are specified in Table M-33. If the Pixel Format Box is not present, the table entries for signed or unsigned integer representations shall be used, depending on whether the channel is signed or unsigned:

Table M-33: Default Offset Values and Encoding of Offsets for the CIELab Colourspace

Pixel Format $\mathbf{F^i}$ and Channel Signedness	OL	OA	OB
unsigned, integer ($\mathbf{BPC^i} < 128$)	0	2^{BPCi}	$2^{BPCi-1}+2^{BPCi-2}$
signed, integer ($\mathbf{BPC^i} \geq 128$)	0	0	0
unsigned, fixed point ($\mathbf{BPC^i} < 128$)	0	2^{BPCi}	$2^{BPCi-1}+2^{BPCi-2}$

signed, fixed point (**BPC^i** ≥ 128)	0	0	0
unsigned, floating point (**BPC^i** < 128)	0	2^{BPCi}	$2^{BPCi-1}+2^{BPCi-2}$
signed, floating point (**BPC^i** ≥ 128)	0	0	0

NOTE - The relation between the number of bits in channel i is given by (BPC_i AND 0x7f)+1, thus a value of 2^{BPCi} indicates an offset of half the available range for channel i. "AND" denotes here the bitwise binary AND operation of the two operands, see also Table A-11 in ITU.T Rec. 800 | ISO/IEC 15444-1.

NOTE - Other applications may use other range values by specifying EP field values. For example, the CIELab encoding in the ICC Profile Format Specification, ICC.1:2001-11 specifies ranges and offsets for the CIELab encoding that are different than the defaults given here. If the values specified in the CIELab encoding in the ICC Profile Format Specification, ICC.1:2001-11, are used, then they would have to be explicitly given in the EP fields.

33) Subclause M.11.7.4.2 – EP field format for the CIEJab colourspace

Replace the second paragraph of subclause M.11.7.4.2 and formula M.19 by the following:

These fields describe how to convert between the unsigned values N_J, N_a, N_b, as defined by CIE Publication No. 131, that are sent to the compressor or received from the decompressor and the signed CIEJab values J, a, b as defined by the CIE. According to CIE Publication No. 131, the calculations from real values Jab to integer, fixed point or floating point format, which are expressed by $N_JN_aN_b$, are made as follows:

$$N_J = \frac{MAX_J}{RJ} \times J + \frac{2^{\lceil \log_2 MAX_J \rceil}}{2^{(BPC_J\ AND\ 0x7F)+1}} \times OJ$$

$$N_a = \frac{MAX_a}{RA} \times a + \frac{2^{\lceil \log_2 MAX_a \rceil}}{2^{(BPC_a\ AND\ 0x7F)+1}} \times OA \qquad \text{(M-19)}$$

$$N_b = \frac{MAX_b}{RB} \times b + \frac{2^{\lceil \log_2 MAX_b \rceil}}{2^{(BPC_b\ AND\ 0x7F)+1}} \times OB$$

where MAX_J, MAX_a and MAX_b are the nominal maximum sample values according to Table M-26 for the channels carrying the J, a^* and b^* data and BPC_x are the bits per component values for channel x as found in the Image Header Box, the Bits Per Component Box or the Palette Box. The brackets ⌈ ⌉ indicate rounding up to the next integer. The numerical values of MAX_J, MAX_a and MAX_b depend on the Image Header box, the Bits Per Component Box, the Palette Box and the Pixel Format Box as explained in subclause M.11.7.2.

Replace the enumeration specifying the encoding and default values of the fields RJ,RA and RB by the following, insert Table M-35. Relable tables M-32 through M-35 to tables M-36 through table M-39:

RJ: Range for J. This field specifies the *RJ* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

OJ: Offset for J. This field specifies the *OJ* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

RA: Range for a. This field specifies the *RA* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

OA: Offset for a. This field specifies the *OA* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

RB: Range for b. This field specifies the *RB* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

OB: Offset for b. This field specifies the *OB* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer.

When the EP fields are omitted for the CIEJab colourspace, then the following default values shall be used. The default J, a and b range parameters RJ, RA and RB are 100, 255 and 255. The default J, a and b offset values depend on the contents of the Pixel Format box, if present, and are indicated in Table M-35. If the Pixel Format Box is not present, the table entries for signed or unsigned integer representations shall be used, depending on whether the channel is signed or unsigned:

Table M-35: Default Offset Values and Encoding of Offsets for the CIEJab Colourspace

Pixel Format $\mathbf{F}^i$ and Channel Signedness	OJ	OA	OB
unsigned, integer ($\mathbf{BPC}^i < 128$)	0	2^{BPCi}	2^{BPCi}
signed, integer ($\mathbf{BPC}^i \geq 128$)	0	0	0
unsigned, fixed point ($\mathbf{BPC}^i < 128$)	0	2^{BPCi}	2^{BPCi}
signed, fixed point ($\mathbf{BPC}^i \geq 128$)	0	0	0
unsigned, floating point ($\mathbf{BPC}^i < 128$)	0	2^{BPCi}	2^{BPCi}
signed, floating point ($\mathbf{BPC}^i \geq 128$)	0	0	0

NOTE -: The relation between the number of bits in channel i is given by (BPC_i AND 0x7f)+1, thus a value of 2^{BPCi} indicates an offset of half the available range for channel i. "AND" denotes here the bitwise binary AND operation of the two operands, see also Table A-11 in ITU.T Rec. 800 | ISO/IEC 15444-1.

34) Subclause M.11.7.8 – Pixel Format box

Add the following as subclause M.11.7.8:

This box defines how samples represented in a channel are interpreted as numerical values representing colour according to a colourspace. If this box is absent, the default interpretation of the corresponding codestream is used. For codestreams following ITU-T Rec. T.800 | ISO/IEC 15444-1 (JPEG 2000), the reconstructed samples form signed or unsigned integers whose bit depth is given by the Palette Mapping Box, Bits Per Component Box or Image Header Box. For ITU-T Rec. T.832 | ISO/IEC 29199-2 (JPEG XR), the codestream reconstructs abstract bit patterns which are, in the absence of this box, interpreted to encode numbers in the binary two's complement.

In the presence of this box, a two-stage conversion process converts the reconstructed sample values from its source format to one of the specified target formats: In the first stage, reconstructed integer samples are represented in binary two's complement form. In the second stage, these bit patterns are re-interpreted according to the contents of this box.

NOTE – In typical computer hardware, integers are already represented in the two's complement notation. The first stage hence requires no operation. The second stage is usually nothing more than a C-style "reinterpretation cast" to the target representation; that is, all this box defines is to which target data type the sample values are "casted" before using them as input for a colour conversion or sample interpretation. Other means in the codestream should be provided to ensure that this re-interpretation allows efficient compression of the data, e.g. the two's complement to sign-magnitude conversion by an appropriate NLT marker.

There shall be at most one Pixel Format Box per JP2 Header Box (if present) or per Compositing Layer Box.

The type of this box shall be 'pxfm' (0x7078 666d). The format of this box shall be as follows:

Insert the following as Figure M-24 and relabel Figures M-24 through M-40 and references thereto to M-25 through M-41.

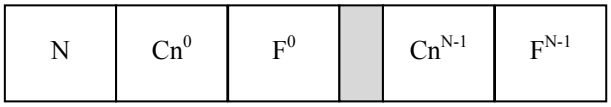

Figure M-24: Layout of the Pixel Format Box

N: Number of channels. This field specifies the number of channels whose pixel format is specified in this box. This number shall be identical to the number of channels defined in the Channel Definition Box if present, or identical to the number of components in the codestream. It is encoded as a two-byte big endian unsigned integer.

Cn^0: Channel index. This field specifies the index of the channel whose pixel format is specified. The value of this field represents the index of the channel as defined within the Component Mapping box or the actual component from the codestream if the file does not contain a Component Mapping box. This field is encoded as a 2-byte big endian unsigned integer.

F^0: Pixel Format. This field specifies the encoding of the pixel value as a bit pattern. For example, the bit pattern reconstructed by the codestream could be interpreted as signed or unsigned integer, as a fixed point value or as a floating point number. These numbers are then interpreted as colour intensities relative to a colourspace. This field is encoded as a 2-byte big endian unsigned integer, Table M-40 lists allowed values for this field and its encoding.

Insert the following as Table M-40 and relabel Tables M-36 through M-40 and references thereto to M-42 through M-46.

Table M-40

Value of $\mathbf{F}^i$	Meaning
`0000 0000 0000 0000`	Bit patterns are interpreted as signed or unsigned integers using a binary two's complement representation.
`0001 0000 0000 0000`	The bit pattern is interpreted as a signed or unsigned integer representing the mantissa of a floating point with a common exponent coming from a channel using an exponent pixel format (see below) associated to the same colour or all of the image. The channel association is defined in the Channel Definition Box. The numerical value of the channel is to be reconstructed by $V = mantissa * 2^{exponent-136}$ The association of the combined sample value is defined by the $\mathbf{Typ}^i$ value in the Channel Definition Box of the channel representing the mantissa.
`0010 0000 0000 0000`	The bit pattern of the channel is to be interpreted as a signed or unsigned integer representing an exponent of a floating point number. The mantissa of this number is described by a channel associated to the same colour or all of the image. The $\mathbf{Typ}^i$ value of an exponent channel in the Channel Definition Box shall be ignored.
`0011 ffff ffff ffff`	The bit pattern of the channel is to be interpreted as fixed point number with **f** fractional bits (i.e. bits to the right of the binary point), or equivalently, a fixed point number that is pre-shifted by **f** bits. The number of integer bits is given as the difference of the bit depths of the corresponding source component minus

	the number of fractional bits. The fractional value is reconstructed by dividing the integer value of the corresponding channel by 2^f.
`0100 mmmm mmmm mmmm`	The bit pattern of the channel is to be interpreted as floating point number with one sign bit, exponent bits and **m** mantissa bits. The number of exponent bits is given by the bit precision of the component or palette entry used to define the channel, minus one, minus the number of mantissa bits. The topmost bit of the bit pattern is the sign bit, followed by the exponent bits, followed by the mantissa bits. The mantissa contains an implicit one bit that is not encoded, and the exponent is encoded as unsigned binary integer with a bias of size $2^{e-1}-1$, where **e** is the number of exponent bits. The reconstruction of the floating point number described by the bit pattern is defined by IEC 60559.
all other values	Reserved for future use.

NOTE – Pixel formats 0001 0000 0000 0000 and 0010 000 0000 0000 are special in the sense that they require the input of two channels to reconstruct the sample value for one colour, where channels are matched corresponding to their association value $Asoc^i$ defined by the Channel Definition Box. A mantissa associated to a specific colour matches an exponent associated to the same colour, or an exponent associated with all of the image - and vice versa. This pixel format is most useful for the RGBE encoding consisting of four channels; the first three channels are associated to the red, green and blue colour and have a pixel format of type 0001 0000 0000 0000 (mantissa), the last channel is associated to all of the image and has a pixel format of type 0010 0000 0000 0000 (exponent).

For fixed point formats, the pixel format specifies only the number of fractional bits, the number of integer bits is implicit. For example, the 2.13 fixed point format of JPEG XR is encoded as a pixel format of 0011 0000 0000 1101 with signed codestream components of 16 bits, the 7.24 fixed point format is represented with 32 bit components and a pixel format of type 0011 0000 0001 1000.

Floating point numbers follow IEC 60559 encodings as much as possible, even though IEC 60559 does not include extensions such as "half-float" numbers. All these encodings share a sign bit **s**, exponent bits **e** and mantissa bits **m** packed into a bit pattern from most significant to least significant in this order. The following table shows the encoding of the most common floating point formats. Note that other formats are possible, and this table provides examples for commonly used floating point formats only:

Table M-41: Common floating point formats (informative)

Value	Meaning
`0100 0000 0001 0111`	IEC 60559 single precision (binary32) format. This format uses 23 mantissa bits, 8 exponent bits and one sign bit. The exponent encoding uses a bias of 127, normalized numbers use exponents between -126 and 127, denormalized numbers have an exponent value of -127, and NaNs and infinities an exponent value of 128. The corresponding components in the codestream must have a bit precision of 32 bits.
`0100 0000 0011 0100`	IEC 60559 double precision (binary64) format. This format uses 52 mantissa bits, 11 exponent bits and one sign bit, the exponent bias is 1023.
`0100 0000 0000 1010`	IEC 60559 half-float (binary16 or half precision) numbers. This format uses 10 mantissa bits, 5 exponent bits and one sign bit. The exponent bias is 15. Normalized numbers use exponent values between -14 and 15, denormalized numbers have an exponent value of -15. Infinities and NaNs are represented by the exponent value 16. This pixel format requires a component of 16 bits precision.

NOTE – If for example the intent is to encode non-negative IEC 60559 single precision numbers with the least significant 8 bits of the mantissa stripped off, the $\mathbf{BPC}^i$ value of a component using this encoding would be 23 as 24 bits in total are used. The corresponding $\mathbf{F}^i$ would then be `0100 0000 0000 1111` as 15 mantissa bits remain in the truncated format.

35) Subclause M.11.10.2 – Instruction Set box

Replace the first paragraph:

An Instruction Set box contains a set of rendering instructions, each represented through a series of composition parameters. In addition, the entire set of instructions contained within this box may be repeated according to a repeat count; this repeating occurs before the reader continues on with the instructions found within the next Instruction Set box in the Composition box. Instruction Set boxes shall be found only within a Composition box; they shall not be found in any other locations in the file.

with the following:

An Instruction Set box contains a set of rendering instructions, each represented through a series of composition parameters. In addition, the entire set of instructions contained within this box may be repeated according to a repeat count; this repeating occurs before the reader continues on with the instructions found within the next Instruction Set box found within the same superbox. Instruction Set boxes shall be found only within either a Composition box or a Compositing Layer Extensions box; they shall not be found in any other locations in the file.

Replace Table M-38 (now M-44) by the follow:

Table M-44 - ITyp field values

Value	Meaning
0000 0000 0000 0000	No instructions are present, and thus no instructions are defined for the compositing layers in the file.
XXXX XXXX XXXX XXX1	Each instruction contains XO and YO parameters.
XXXX XXXX XXXX XX1X	Each instruction contains the WIDTH and HEIGHT parameters.
XXXX XXXX XXXX 1XXX	Each instruction contains the LIFE, N and PERSIST animation parameters.
XXXX XXXX XX1X XXXX	Each instruction defines the crop parameters XC, YC, WC and HC.
XXXX XXXX X1XX XXXX	Each instruction defines the rotation parameter ROT.
All other bit flags	Reserved for future use

36) Subclause M.11.10.2.1 – Instruction parameter

*Add a 4 byte field **ROT** to the end of figure M.27 (now M.28).*

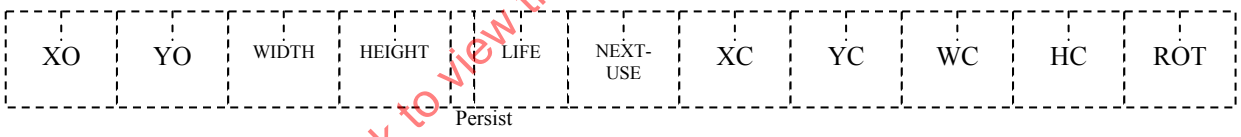

Figure M.28 – Organization of the contents of an INST field within an Instruction Set box

*Add a paragraph below paragraph 11 (starting with **HC**):*

ROT: Rotation. This field specifies an optional rotation and mirroring that is to be applied as last step after cropping and before rescaling and rendering the image to the screen. If the **ROT** field is not present or is zero, the image shall be rendered in an orientation according to the specifications of the corresponding codestream.

This field is encoded as a 4-byte big endian integer; its encoding is specified in **Table M-47**.

*Insert the following row into **Table M-40** (now Table M-46):*

ROT	32	0-31, see **Table M-47**; if ITyp contains XXXX XXXX X1XX XXXX

	0	Not applicable otherwise

*Insert the following **Table M-47** below **Table M-46** (now Table M-46) and relabel Tables M-41 through M-49 and references thereto to Tables M-48 through M-56:*

Table M-47 - Encoding of the ROT field

Values (bits) MSB LSB	Meaning
0000 0000	Orientation not specified, use the orientation defined in the codestream (if any).
000x 0001	Rotate by 0° clockwise
000x 0010	Rotate by 90° clockwise
000x 0011	Rotate by 180° clockwise
000x 0100	Rotate by 270° clockwise
0001 xxxx	Flip image left to right after rotation
all other values	Reserved for future use

Add the following paragraph at the end of C.11.10.2.1:

For rendering an image in the presence of an instruction set box, the following steps shall be applied:

- Crop the image to XC, YC, WC, and HC if cropping parameters are present,
- Rotate and/or flip the image according to the ROT parameter if present. If the ROT parameter is not present, the image takes the orientation defined by the corresponding codestream.
- Rescale the image to WIDTH and HEIGHT if these parameters are present.
- Render the top left edge of the resulting image at position XO,YO into the compositing surface, or at position 0,0 if the XO and YO fields are not present.

37) Subclause M.11.16 – ROI Description box

Replace the definition of **Rtyp^{i}** *with*:

Rtyp^{i}: Region of Interest type and encoding, can be simple rectangular (one value), simple elliptical (one value), quadrilateral refinement (multiple values, corresponding to 4 permutation bit fields A, B, C and D), or oriented elliptical refinement (one value). **Rtyp^{i}** may identify a quadrilateral refinement only if $i > 1$ and **Rtyp^{i-1}** identifies a simple rectangular region. Similarly, Rtyp^{i} may identify an oriented elliptical refinement only if $i > 1$ and **Rtyp^{i-1}** identifies a simple elliptical region. The **Rtyp^{i}** field is encoded as an 8 bit integer; allowed values are as follows:

Replace Table M-48 (now M-53) with:

Table M-53 – Allowed Rtyp^{i} values

Value (bits) MSB LSB	Meaning
0000 0000	Aligned rectangular region of interest (rectangle edges parallel to the image edges)

0000 0001	Aligned elliptical region of interest (ellipse diameters parallel to the image edges)
AABB CDD0	Quadrilateral refinement, with bit fields: A (0-3), B (0-2), C (0-1), D (1-3)
0000 0011	Oriented elliptical refinement
other values	Reserved for future use.

Replace the definition of **Rlcx^i** *with*:

Rlcx^i: Region of Interest horizontal location. In the case of an Aligned Rectangular Region of Interest or a Quadrilateral Refinement, this is the horizontal position of the top left corner of the relevant rectangle. In the case of an Aligned Elliptical Region of Interest this is the horizontal position of the centre point. For an Oriented Elliptical Refinement, this is the horizontal position at which the ellipse touches the upper edge of its bounding box, as identified by the immediate preceding region of interest. This value is stored as a 4-byte big endian unsigned integer.

Replace the definition of **Rlcy^i** *with*:

Rlcy^i: Region of Interest vertical location. In the case of an Aligned Rectangular Region of Interest or a Quadrilateral Refinement, this is the vertical position of the top left corner of the relevant rectangle. In the case of a Simple Elliptical Region of Interest this is the vertical position of the centre point. For an Oriented Elliptical Refinement, this is the vertical position at which the ellipse touches the left edge of its bounding box, as identified by the immediate preceding region of interest. This value is stored as a 4-byte big endian unsigned integer.

Replace the definition of **Rwdt^i** *with*:

Rwdt^i: Region of Interest width. In the case of an Aligned Rectangular Region of Interest or a Quadrilateral Refinement, this is the width of the relevant rectangle. In the case of an Aligned Elliptical Region of Interest this is the amount by which the ellipse extends to the left and to the right of its centre. For an Oriented Elliptical Refinement, this parameter shall be 1. This value is stored as a 4-byte big endian unsigned integer.

Replace the definition of **Rhth^i** *with*:

Rhth^i: Region of Interest height. In the case of an Aligned Rectangular Region of Interest or a Quadrilateral Refinement, this is the height of the relevant rectangle. In the case of an Aligned Elliptical Region of Interest this is the amount by which the ellipse extends above and below its centre. For an Oriented Elliptical Refinement, this parameter shall be 1. This value is stored as a 4-byte big endian unsigned integer.

Replace Table M-49 (now M-56) with:

Table M-56 – Format of the contents of the ROI Description box

Parameter	Size (bits)	Value
Nroi	8	0 – 255
R^i	8	0 – 255
$Rtyp^i$	8	0 – 255
$Rsig^i$	8	0 – 255
$Rlcx^i$	32	$0 – (2^{32}-1)$
$Rlcy^i$	32	$0 – (2^{32}-1)$
$Rwdt^i$	32	$1 – (2^{32}-1)$ if $Rtyp^i \neq 3$ 1 if $Ryp^i = 3$

$Rhth^i$	32	$1 - (2^{32}-1)$ if $Rtyp^i \neq 3$ 1 if $Ryp^i = 3$

Append the following text to subclause M.11.16:

If **Rtyp^i** identifies an Aligned Rectangular Region of Interest, the locations (x,y) which are considered to be included within the region are those which satisfy:

$Rlcx^i \leq x < Rlcx^i + Rwdt^i$ and $Rlcy^i \leq y < Rlcy^i + Rhth^i$.

If $Rtyp^i$ identifies an Aligned Elliptical Region of Interest, the locations (x,y) which are considered to be included within the region are those which satisfy:

$$(x - Rlcx^i)^2 / (Rwdt^i)^2 + (y - Rlcy^i)^2 / (Rhth^i)^2 \leq 1,$$

where floating-point arithmetic is employed, as opposed to integer division.

If **Rtyp^i** identifies a Quadrilateral Refinement, **Rtyp^{i-1}** shall identify an Aligned Rectangular Region of Interest, and the values of **Rlcx^{i-1}**, **Rlcy^{i-1}**, **Rwdt^{i-1}** and **Rhth^{i-1}** shall describe a bounding rectangle that is refined into a quadrilateral, by using the **A**, **B**, **C** and **D** bit-fields identified in Table M-56, together with the values of **Rlcx^i**, **Rlcy^i**, **Rwdt^i** and **Rhth^i**. Together, these values supply four horizontal and four vertical coordinates, identified as:

$X[0] = Rlcx^{i-1}$, $X[1] = Rlcx^i$, $X[2] = Rlcx^i + Rwdt^i - 1$, $X[3] = Rlcx^{i-1} + Rwdt^{i-1} - 1$ and

$Y[0] = Rlcy^{i-1}$, $Y[1] = Rlcy^i$, $Y[2] = Rlcy^i + Rhth^i - 1$, $Y[3] = Rlcy^{i-1} + Rhth^{i-1} - 1$

These values shall satisfy the following constraints:

$X[0] \leq X[1] \leq X[2] \leq X[3]$ and $Y[0] \leq Y[1] \leq Y[2] \leq Y[3]$.

The quadilateral is defined by four vertices V1, V2, V3 and V4, having horizontal and vertical coordinates (V1x,V1y), (V2x,V2y), (V3x,V3y) and (V4x,V4y). These vertices and associated edges are considered to be included within the region. The vertical coordinates of the vertices shall be obtained as follows:

V1y = Y[0]; V2y = Y[1]; V3y = Y[2]; and V4y = Y[3].

The horizontal coordinates of the vertices shall be obtained using the following equations:

V1x = X[A], where A is the 2 bit field identified in Table M-56, taking values in the range 0 to 3;

V2x = X[(A+1+B) mod 4], where B is the 2 bit field identified in Table M-56, taking values in the range 0 to 2;

V3x = X[(A+1+((B+1+C) mod 3)) mod 4], where C is the 1 bit field identified in Table M-56; and

V4x = X[(A+1+((B+2-C) mod 3)) mod 4].

The connectivity of the vertices is determined by the 2 bit field D identified in Table M-56, which takes values in the range 1 to 3. Table M-58 identifies the three possible connectivity cases and the associated values of D.

Insert the following as table M-57 and relabel Tables M-50 through M-52 and references thereto to Tables M-58 through M-60:

Table M-57 – Interpreting the 2 bit D field of Rtyp^i for quadrilateral refinements

DD	Connectivity of quadrilateral vertices
01	V1 → V2 → V3 → V4 → V1
10	V1 → V3 → V4 → V2 → V1
11	V1 → V4 → V2 → V3 → V1

The vertices obtained by following the above procedure shall correspond to those of a well-formed quadrilateral region, in which opposite edges do not intersect, except possibly at their end-points.

NOTE – Any or all edges may be degenerate, in the sense that the two vertices that define any edge may coincide. In this way, quadrilateral refinements may be used to describe triangular regions, arbitrarily oriented lines (of width 1) or even isolated points.

If **Rtyp^i** identifies an Oriented Elliptical Refinement, **Rtyp^{i-1}** shall identify an Aligned Elliptical Region of Interest, and the values of **Rlcx^{i-1}**, **Rlcy^{i-1}**, **Rwdt^{i-1}** and **Rhth^{i-1}** shall describe the centre, left/right extent and above/below extent of a bounding rectangle for the oriented elliptical region. In this case, the values of **Rwdt^i** and **Rhth^i** shall be equal to 1 and

the values of $Rlcx^i$ and $Rlcy^i$ shall correspond to the horizontal location at which the ellipse touches the upper boundary and the vertical location at which the ellipse touches the left boundary of this bounding rectangle.

The oriented elliptical region may be obtained using either of the horizontal or vertical skewing procedures described below; these procedures are equivalent, up to integer rounding effects. These procedures employ the following quantities:

$W_O = Rwdt^{i-1}$; $H_O = Rhth^{i-1}$; $X_C = Rlcx^{i-1}$; $Y_C = Rlcy^{i-1}$; $\alpha = (Rlcx^i - X_C) / H_O$; $\beta = (Y_C - Rlcy^i) / W_O$;

$W_I = W_O * \text{sqrt}(1 - \alpha*\beta)$; and $H_I = H_O * \text{sqrt}(1 - \alpha*\beta)$.

Horizontal Skewing Procedure*:* Starting with an elliptical region centred at the location (X_C, Y_C), extending horizontally by $\pm W_I$ and vertically by $\pm H_O$, shift each location (x,y) within this initial ellipse horizontally by an amount $\alpha*(Y_C-y)$. Location (x,y) belongs to the oriented ellipse if and only if it satisfies:
$(x - X_C - \alpha(Y_C-y))^2 / W_I^2 + ((y - Y_C)^2 / H_O^2 \le 1$.

Vertical Skewing Procedure*:* Starting with an elliptical region centred at the location (X_C, Y_C), extending horizontally by $\pm W_O$ and vertically by $\pm H_I$, shift each location (x,y) within this initial ellipse vertically by an amount $\beta*(x-X_C)$. Location (x,y) belongs to the oriented ellipse if and only if it satisfies:
$(y - Y_C - \beta (x-X_C))^2 / H_I^2 + ((x - X_C)^2 / W_O^2 \le 1$.

The values of $\mathbf{Rlcx^i}$ and $\mathbf{Rlcy^i}$ shall satisfy:

$X_C - W_O < Rlcx^i < X_C + W_O$; $Y_C - H_O < Rlcy^i < Y_C + H_O$.

Moreover, there shall be a γ in the range -1 to 1, such that $\mathbf{Rlcx^i}$ and $\mathbf{Rlcy^i}$ satisfy the following compatibility constraint:

$\gamma W_O - ½ \le (Rlcx^i - X_C) \le \gamma W_O + ½$; and $\gamma H_O - ½ \le (Y_C - Rlcy^i) \le \gamma H_O + ½$.

NOTE – The compatibility constraint serves to ensure compatibility between the horizontal and vertical skewing procedures.

38) New subclause M.11.21 – Compositing Layer Extensions box

Introduce the following new subclause, including associated tables and figures:

M.11.21 Compositing Layer Extensions box

If a JPX file involves a large number of compositing layers, whose headers are constructed according to a repeating pattern, it may be possible to describe them compactly using one or more Compositing Layer Extensions boxes. Compositing Layer Extensions boxes also allow a single animation to be separated into multiple alternate presentation threads.

For example, a long sequence of multispectral images may be represented by codestreams, each of which uses the Multicomponent Transformation extensions described in Annex J to provide four different three-channel renditions and one panchromatic channel, each involving different linear combinations of the underlying multispectral component data. These five separate renditions can be assigned colourspaces, for display and other rendering purposes, via distinct compositing layers. The Compositing Layer Extensions box allows each of these five types compositing layers to be assigned a separate presentation thread, with its own timing for composition purposes. Moreover, the Compositing Layer Extensions box allows a long succession of such compositing layers, all formed in essentially the same way, from a succession of codestreams, to be represented together in a compact form, along with any associated metadata.

Compositing Layer Extensions boxes may be found only at the top-level of the file (not within any superbox). If the file contains any Compositing Layer Extensions box, then it must also contain a Compositions box, at least one Codestream Header box and at least one Compositing Layer Header box. Moreover, all such boxes must precede any Compositing Layer Extensions boxes within the file.

The Compositing Layer Extensions box is a superbox. It may contain zero or more Codestream Header boxes (as immediate sub-boxes only), one or more Compositing Layer Header boxes (as immediate sub-boxes only), zero or more Instruction Set boxes (as immediate sub-boxes only), together with zero or more auxiliary metadata boxes (e.g., Label boxes, XML boxes and Association boxes). The order in which these boxes appear within the Compositing Layer Extensions box is important and is explained below.

The type of the Compositing Layer Extensions box shall be 'jclx' (0x6A63 6C78) and its contents shall be as follows:

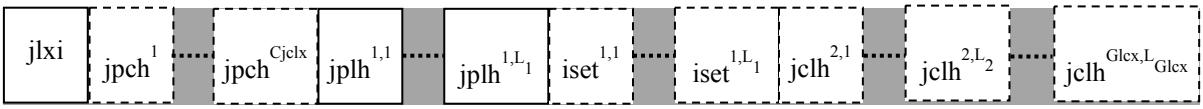

Figure M-42 – Organization of the contents of a Compositing Layer Extensions box

Cjclx: Number of Codestream Header boxes in the Compositing Layer Extensions box. Cjclx may be zero.

jpch^{c}: Codestream Header box c, where c runs from 1 to **Cjclx**. The Codestream Header boxes, if any, shall appear before any other boxes.

Gjclx: Number of compositing groups in the Compositing Layer Extensions box. **Gjclx** shall be at least 1. Compositing groups shall appear consecutively, immediately following any Codestream Header boxes.

Lg: Number of Compositing Layer Header boxes in compositing group g, where g runs from 1 to **Gjclx**. **Lg** shall be at least 1.

jplh^{g,j}: Compositing Layer Header box j, within compositing group g, where j runs from 1 to **Lg**. The Compositing Layer Header boxes within each compositing group g shall precede any Instruction Set boxes within compositing group g.

Ig: Number of Instruction Set boxes in compositing group g, where g runs from 1 to **Gjclx**. **Ig** may be 0 only in the last group – i.e., when g=**Gjclx**. All other **Ig** values must be at least 1.

iset^{g,i}: Instruction Set box i, within compositing group g, where i runs from 1 to **Ig**. The Instruction Set boxes, if any, within compositing group g shall immediately follow the Compositing Layer Header boxes for compositing group g.

Ljclx: Total number of Compositing Layer Header boxes found within the Compositing Layer Extensions box. This value shall equal the sum of the Lg values for g=1 to g=**Gjclx**.

Tjclx: Number of presentation threads defined by the Compositing Layer Extensions box. This value shall be equal to **Gjclx** if all compositing groups contain Instruction Set boxes. Otherwise, the last compositing group contains no Instruction Set boxes and the value of **Tjclx** shall be equal to **Gjclx**-1. If there are no Instruction Set boxes at all, the value of **Tjclx** shall be 0.

Lref^{g}: Number of compositing layers implicitly referenced by the Instruction Set boxes **iset**^{g,1} to **iset**^{g,Ig}, where g runs from 1 to **Tjclx**. The procedure for calculating this value is given below.

Fjclx: Number of composited frames associated with each presentation thread defined by the Compositing Layer Extensions box. This value is supplied by the Compositing Layer Extensions Info box ("jlxi"), except possibly for the last Composition Layer Extensions box in the file; that box's Composition Layer Extensions Info box may hold the value 0 in place of an **Fjclx** value. Presentation threads defined by the final Composition Layer Extensions box may different numbers of frames, as determined by the thread's compositing instructions and compositing layers.

Mjclx: Repetition factor for the Compositing Layer Extensions box. The Compositing Layer Extensions box defines a total of **Mjclx** × **Cjclx** codestream headers and **Mjclx** × **Ljclx** compositing layers. The value of Mjclx is supplied by the Compositing Layer Extensions Info box ("jlxi"), except possibly for the last Composition Layer Extensions box in the file; that box's Composition Layer Extensions Info box may hold the value 0 in place of the **Mjclx** value, but only if **Cjclx** is non-zero. In this case the value of Mjclx shall be determined in the manner described below.

For the last Compositing Layer Extensions box in the file, if **Cjclx** is non-zero the value of **Mjclx** need not be provided within the Compositing Layer Extensions Info box; in this case, the Compositing Layer Extensions Info box supplies a value of 0 in place of the **Mjclx** value and **Mjclx** is defined implicitly through the number, **Ctotal**, of actual Contiguous Codestream boxes and Fragment Table boxes that appear either at the top-level of the file or within Multiple Codestream boxes. Specifically, in the case of the final Compositing Layer Extensions box in the file, unless the file contains no Codestream Header boxes at all, the value of **Mjclx** shall satisfy

$$\mathbf{Ctotal} = \mathbf{Cprev} + \mathbf{Mjclx} \times \mathbf{Cjclx},$$

where **Cprev** is equal to the number of top-level Codestream Header boxes, added to the total number of additional codestream headers appearing within all preceding Compositing Layer Extensions boxes. This equation implicitly defines the value for **Mjclx** when **Cjclx** is non-zero.

NOTE – In cases where a file is being generated dynamically, adding codestreams as they become available (e.g., from a camera), it can be useful to provide 0 in place of the Mjclx value within the final Compositing Layer Info box.

If Instruction Set boxes are present, **Tjclx** > 0 and each of the initial **Tjclx** compositing groups defines a separate presentation thread. Each presentation thread provides an alternate sequence of composited frames, that extends the sequence of frames created by the Composition box. The presentation thread inherits any composited frames created by following the instructions in the Composition box, as explained in M.5.3.2.1. However, the instructions found in the Composition box do not relate to any of the compositing layers associated with Composition Layer Extensions boxes, nor do any persistent composited frames created by the instructions found within a presentation thread persist into other presentation threads, found in the same or a different Compositing Layer Extensions box.

It is possible that the file contains multiple Compositing Layer Extensions boxes, each of which provides a different number of presentation threads, some possibly having no presentation threads at all. Together with the Composition box, these offer T global presentation threads to renderers, where T is the maximum of the **Tjclx** values amongst all Compositing Layer Extension boxes. For each t in the range 1 to T, global presentation thread t consists of the sequence of composited frames offered by the Composition box, followed by the composited frames defined by compositing group g = min{t,**Tjclx**} of each successive Compositing Layer Extensions box for which Tjclx is non-zero.

The set of codestream headers defined by a Compositing Layer Extensions box shall be formed by replicating the entire set of **Cjclx** Codestream Header boxes **Mjclx** times and numbering the codestream header produced by the r^{th} replica of Codestream Header box $jpch^{c}$, starting from r=0, as

$$\mathbf{Cprev} + r \times \mathbf{Cjclx} + c.$$

The set of compositing layers and associated headers defined by a Compositing Layer Extensions box shall be formed by replicating the entire set of **Ljclx** Compositing Layer Header boxes within the box **Mjclx** times and numbering the compositing layer associated with the r^{th} replica of Compositing Layer header box $\mathbf{jplh}^{g,j}$ as

$$\mathbf{Lprev} + r \times \mathbf{Ljclx} + \sum_{1 \le n < g} \mathbf{Ln} + j,$$

where **Lprev** is equal to the number of top-level Compositing Layer Header boxes, added to the total number of additional compositing layers defined by all preceding Compositing Layer Extensions boxes.

For each g from 1 to **Tjclx**, the composited frames associated with presentation thread g are formed by applying the compositing instructions found within Instruction Set boxes $\mathbf{iset}^{g,1}$ to $\mathbf{iset}^{g,Ig}$ to the compositing layers formed from the **Mjclx** replicas of Compositing Layer Header boxes $\mathbf{jclh}^{g,1}$ to $\mathbf{jclh}^{g,Lg}$.
The last Compositing Layer Extensions box in a file is not required to provide a value for **Fjclx**. Apart from this case, all presentation threads in the Compositing Layer Header box shall have the same number of frames, **Fjclx**.

The Compositing Layer Extensions box may also contain other metadata boxes, such as Label boxes, XML boxes, Association boxes and Cross-Reference boxes. These additional boxes are not replicated, unless they are found within a replicated Codestream Header box or Compositing Layer Header box.

If Compositing Layer Header boxes do not contain Codestream Registration boxes, the compositing layer that is numbered with index n uses the codestream that is numbered with index n. If Codestream Registration boxes are used, they shall be found in all Compositing Layer Header boxes, both at the top-level of the file and in all Compositing Layer Extensions boxes. In this case, the codestream indices contained in the Codestream Registration boxes that are found within a Compositing Layer Header box shall be remapped, according to the following procedure. An original codestream index c, found within a Codestream Registration box that is found within a Compositing Layer Header box shall satisfy either:

$$0 \le c < \mathbf{Ctop}, \text{ or else}$$
$$\mathbf{Cprev} \le c < \mathbf{Cprev}+\mathbf{Cjclx},$$

where **Ctop** is equal to the number of top-level Codestream Header boxes in the file. Moreover, in the r^{th} replica of the Compositing Layer Header box, this original codestream index c shall be mapped to Cmap(r,c), where

$$\text{Cmap}(r,c) = c, \text{ if } c < \mathbf{Ctop}, \text{ else}$$
$$\text{Cmap}(r,c) = \mathbf{Cprev}-\mathbf{Ctop} + r \times \mathbf{Cjclx} + c, \text{ if } c \ge \mathbf{Ctop}$$

If a Number List box is found anywhere within either a Compositing Layer Header box or a Codestream Header box that is inside a Compositing Layer Extensions box, any codestream index c that is found within the Number List box shall also be remapped to Cmap(r,c). Any compositing layer index n that is found within such a Number List shall be remapped to Lmap(r,n), where

$$\text{Lmap}(r,n) = n, \text{ if } n < \mathbf{Ltop}, \text{ else}$$

Lmap(r,n) = **Lprev-Ltop** + r × **Ljclx** + n, if n ≥ **Ltop**

and **Ltop** is equal to the number of top-level Compositing Layer Header boxes in the file.

If a Number List box is found anywhere within a Compositing Layer Extensions box that is not within a Compositing Layer Header box or a Codestream Header box, codestream indices c and compositing layer indices n that are found within the Number List box shall be interpreted as references to all codestreams Cmap(0,c) through Cmap(Mjclx-1,c) and all compositing layers Lmap(0,n) through Lmap(Mjclx-1,n), respectively.

39) Subclause M.11.22 -- Compositing Layer Extensions Info box

Introduce the following new subclause, including associated tables and figures:

M.11.22 Compositing Layer Extensions Info box

The Compositing Layer Extensions Info box provides global information concerning repetition, compositing layers and composited frames associated with a Compositing Layer Extensions box. This box shall only be found as the first box inside a Compositing Layer Extensions box.

The type of the Compositing Layer Extensions Info box shall be 'jlxi' (0x6A6C7869) and it shall have the following contents:

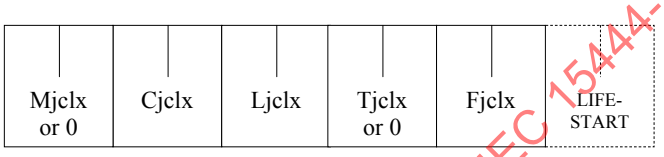

Figure M-43: Organization of the Compositing Layer Extensions Info Box

Mjclx or 0: Repetition factor for the Compositing Layer Extensions box, represented by a 4-byte unsigned big endian integer. This value may be 0 only in the last Compositing Layer Extensions box of the file, and then only if Cjclx is non-zero; in this case the repetition factor Mjclx is determined using the value of **Ctotal**, as described in M.11.21.

Cjclx: Number of Codestream Header boxes in the Compositing Layer Extensions box represented by a 4-byte unsigned big endian integer.

Ljclx: Total number of Compositing Layer Header boxes found within the Compositing Layer Extensions box represented by a 4-byte unsigned big endian integer.

Tjclx: Number of presentation threads defined by the Compositing Layer Extensions box represented by a 4-byte unsigned big endian integer.

Fjclx or 0: Number of composited frames associated with each presentation thread defined by the Compositing Layer Extensions box represented by a 4-byte unsigned big endian integer. This value may be 0 only if **Tjclx** is 0 or within the last Compositing Layer Extensions box in the file. In the latter case, the number of frames in each presentation thread is determined by the compositing instructions themselves, together with the number of available compositing layers.

LIFE-START:The start time for the first composited frame in each presentation thread defined by the Compositing Layer Extensions box, measured in milliseconds, relative to the start of the first composited frame defined by the Composition (comp) box. This field shall appear only if **Tjclx** is non-zero and is represented by a 4-byte unsigned big endian integer.

If **Tjclx** is non-zero, the duration of composited frames associated with the Composition box or previous Compositing Layer Extensions boxes shall be truncated, as required, in order to ensure that they are not presented beyond the point defined by LIFE-START.

40) New subclause M.11.23 – Multiple Codestream box

Introduce the following new subclause, including associated tables and figures:

M.11.23 Multiple Codestream box

If a JPX file contains a large number of codestreams, it may be useful to encapsulate them within one or more Multiple Codestream boxes. Each Multiple Codestream box identifies the number of codestreams which it contains, together with information that may allow readers to efficiently recover specific codestreams of interest. A Multiple Codestream box may contain contiguous codestreams, fragmented codestreams or both.

Multiple Codestream boxes may be found only at the top-level of the file or as immediate sub-boxes of other Multiple Codestream boxes. Moreover, all top-level Contiguous Codestream boxes and Fragment Table boxes in a JPX file must precede any Multiple Codestream boxes.

If the file's compositing layers involve Codestream Registration boxes, the codestreams referenced by such boxes within all top-level Compositing Layer Header boxes shall appear at the top-level of the file (not within Multiple Codestream boxes). If the file's compositing layers do not involve Codestream Registration boxes, there shall be at least one top-level codestream for each top-level compositing layer (i.e., for each compositing layer not defined by a Compositing Layer Extensions box). In any event, the file shall contain at least one top-level codestream that is not found within a Multiple Codestream box.

NOTE – For most purposes, these restrictions mean that the codestreams represented by Multiple Codestream boxes are likely to be those that are associated with Codestream Header boxes found within Compositing Layer Extensions boxes. However, this need not always be the case, and there is no implied correspondence between Multiple Codestream boxes and Compositing Layer Extensions boxes.

The type of the Multiple Codestream box shall be 'j2cx' (0x6A32 6378) and it shall have the following contents:

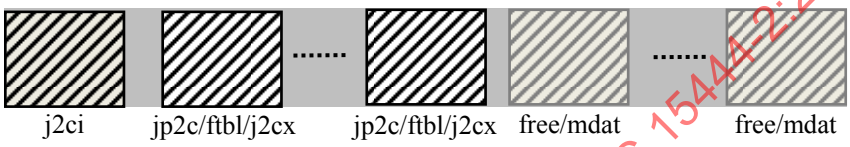

Figure M-44 – Organization of the contents of a Multiple Codestream box

j2ci: A Multiple Codestream Info box, specifying the total number of codestreams contained within this Multiple Codestream box.

jp2c/ftbl/j2cx:A Contiguous Codestream box, a Fragment Table box or another Multiple Codestream box. The total number of codestreams found within these boxes shall agree with the value identified via the Multiple Codestream Info box.

mdat/mfree: Zero or more Media Data or Free boxes; these shall appear after all Contiguous Codestream, Fragment Table and Multiple Codestream boxes that appear within this Multiple Codestream box.

NOTE - It is advisable for file writers to construct Multiple Codestream boxes from sub-boxes that all have the same type, all have the same length and all represent the same number of codestreams, except possibly the last sub-box. In particular, this means that it is advisable to embed Fragment Table boxes rather than Contiguous Codestream boxes within a Multiple Codestream box. This is especially useful in cases where the Multiple Codestream box contains a large number of codestreams, since it allows a reader to use the information found in the Multiple Codestream Info box to efficiently locate codestreams of interest. The actual contents of the codestreams referenced by Fragment Table boxes might be written to Media Data boxes within the same file, or else they might be found in other files.

NOTE – In some cases, it may be advantageous for a writer to allocate space to accommodate a fixed number N of Fragment Table boxes within a Multiple Codestream boxes, even if the actual number of codestreams is not definitely known. If the number of codestreams turns out to be smaller than N, the Ncs value within the Multiple Codestream Info sub-box may be rewritten and the unused Fragment Table boxes may be rewritten as Free boxes, without affecting other boxes that may have been written to the file. If the number of codestreams turns out to be larger than N, additional Multiple Codestream boxes may be written.

Table M-61 Format of the contents of the Multiple Codestream box

Parameter	Size (bits)	Value
j2ci	128	varies
jp2c/ftbl/j2cx	varies	varies
mdat	varies	varies

41) Subclause M.11.24 – Multiple Codestream Info box

Introduce the following new subclause, including associated tables and figures:

M.11.24 Multiple Codestream Info box

The Multiple Codestream Info box shall only be found as the first box inside a Multiple Codestream box. The Multiple Codestream Info box provides the total number of codestreams contained or referenced from within this Multiple Codestream Box, along with auxiliary information that may allow readers to efficiently recover specific codestreams of interest.

The type of the Multiple Codestream box shall be 'j2ci' (0x6A32 6369) and it shall have the following contents:

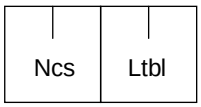

Figure M-45 – Organization of the contents of a Multiple Codestream Offsets box

Ncs: Total number of codestreams represented by the Multiple Codestream box, within which this box occupies the first position. This value is represented by a 4-byte unsigned big endian integer.

Ltbl: This field is represented as a 4-byte unsigned big endian integer. This value is either 0 or else it represents the common size and common number of codestreams that are represented by all of the Contiguous Codestream, Fragment Table or Multiple Codestream boxes that follow this box, except possibly the last one. Specifically, if B is the number of Contiguous Codestream, Fragment Table or Multiple Codestream boxes that follow this box as immediate sub-boxes of the containing Multiple Codestream box, and if Ltbl is not equal to zero, then at least the first B-1 of these Contiguous Codestream, Fragment Table or Multiple Codestream sub-boxes each have a length of L bytes, including the box header, and each represent a total of 2^R codestreams, where $R < 64$, $L < 2^{26}$, and $Ltbl = R \times 2^{26} + L$.

Table M-62 Format of the contents of the Multiple Codestream Info box

Parameter	Size (bits)	Value
Ncs	32	$1 — (2^{32}-1)$
Ltbl	32	$0 — (2^{32}-1)$

42) New subclause M.11.25 – Grouping box

Introduce the following new subclause:

M.11.25 Grouping Box

The Grouping box provides a mechanism to group other boxes within a container without specifying specific associations as in the Association box. The Grouping box is a superbox.

The type of the Grouping box shall be 'grp\040' (0x6772 7020) and it shall have the following contents:

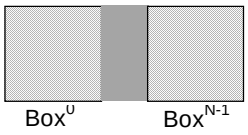

Figure M-46 – Organization of the contents of a Grouping box

The Grouping box shall not be the first box within an Association box. Logically, all boxes found within the Grouping box shall be interpreted as if they were found at the level of the grouping box. However, boxes that are required to be sub-boxes of a particular super-box, according to the remainder of this specification, shall not be found within Grouping boxes.

NOTE - Grouping boxes could be expected to appear in places where Association boxes might appear. The sub-boxes of a Grouping box would typically be boxes that could be expected to appear within an Association box. Examples of such sub-boxes inlcude (but are not limited to) Label boxes, Number List boxes, Region of Interest Description boxes, XML boxes, Association boxes, other Grouping boxes and Cross Reference boxes that provide references to such boxes. Because sub-boxes of a Grouping box would have the same meaning if found at the same level as the Grouping box, the use of Grouping boxes does not have any semantic implications for the contained metadata. However, the use of Grouping boxes may facilitate the selective delivery of metadata of interest by a