INTERNATIONAL STANDARD

ISO/IEC 15444-2

Third edition
2023-11

Information technology — JPEG 2000 image coding system —

Part 2: Extensions

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

Partie 2: Extensions

Reference number
ISO/IEC 15444-2:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted.

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by ITU-T (as Rec. ITU-T T.801) and drafted in accordance with its editorial rules, in collaboration with Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This third edition cancels and replaces the second edition (ISO/IEC 15444-2:2021), which has been technically revised.

The main changes are as follows:

— Support for progression order extensions.

A list of all parts in the ISO/IEC 15444 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

CONTENTS

Page

Page

List of Tables

Page

Page

Page

Page

List of Figures

Page

Page

Page

Page

Page

Page

Page

Introduction

This Recommendation | International Standard defines a set of lossless (bit-preserving) and lossy compression methods for coding continuous-tone, bi-level, grey-scale, colour digital still images, or multi-component images.

This Recommendation | International Standard:

- specifies extended decoding processes for converting compressed image data to reconstructed image data;
- specifies an extended codestream syntax containing information for interpreting the compressed image data;
- specifies an extended file format;
- specifies a container to store image metadata;
- defines a standard set of image metadata;
- provides guidance on extended encoding processes for converting source image data to compressed image data;
- provides guidance on how to implement these processes in practice.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: Extensions

1 Scope

This Recommendation | International Standard defines a set of lossless (bit-preserving) and lossy compression methods for coding continuous-tone, bi-level, grey-scale, colour digital still images, or multi-component images.

This Recommendation | International Standard:

- specifies extended decoding processes for converting compressed image data to reconstructed image data;
- specifies an extended codestream syntax containing information for interpreting the compressed image data;
- specifies an extended file format;
- specifies a container to store image metadata;
- defines a standard set of image metadata;
- provides guidance on extended encoding processes for converting source image data to compressed image data;
- provides guidance on how to implement these processes in practice.

2 Normative references

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

2.1 Identical Recommendations | International Standards

- Recommendation ITU-T T.81 (1992) | ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*.
- Recommendation ITU-T T.82 (1993) | ISO/IEC 11544:1993, *Information technology – Coded representation of picture and audio information – Progressive bi-level image compression.*
- Recommendation ITU-T T.84 (1996) | ISO/IEC 10918-3:1997, *Information technology – Digital compression and coding of continuous-tone still images: Extensions.*, including Rec. ITU-T T.84 (1996)/Amd.1 (1999) | ISO/IEC 10918-3:1997/Amd.1:1999, *Information technology – Digital compression and coding of continuous-tone still images: Extensions – Amendment 1: Provisions to allow registration of new compression types and versions in the SPIFF header*.
- Recommendation ITU-T T.800 (2019) | ISO/IEC 15444-1:2019, *Information technology – JPEG 2000 image coding system: Core coding system*.
- Recommendation ITU-T T.805 | ISO/IEC 15444-6, *Information technology – JPEG 2000 image coding system: Compound image file format*.
- Recommendation ITU-T T.814 (2019) | ISO/IEC 15444-15:2019, *Information technology – JPEG 2000 image coding system: High-throughput JPEG 2000*.
- Recommendation ITU-T T.832 (2019) | ISO/IEC 29199-2:2020, *Information technology – JPEG XR image coding system – Image coding specification*.

2.2 Paired Recommendations | International Standards

- Recommendation ITU-T H.273 (in force), *Coding-independent code points for video signal type identification.*

ISO/IEC 23091-2: (in force), *Information technology – Coding-independent code points: Part 2: Video*.

2.3 Additional references

- Recommendation ITU-T T.42 (2003), *Continuous-tone colour representation method for facsimile*.
- Recommendation ITU-T T.45 (2000), *Run-length Colour Encoding*.
- IEC 61966-2-1 (in force), *Multimedia systems and equipment – Colour measurement and management: Part 2-1: Colour management – Default RGB colour space – sRGB*.
- IEC 61966-2-2:2003, *Multimedia systems and equipment – Colour measurement and management – Part 2-2: Colour management – Extended RGB colourspace – scRGB*.
- IETF RFC 1321 (1992), *The MD5 Message-Digest Algorithm*.
- IETF RFC 2630 (1999), *Cryptographic Message Syntax*.
- ISO 3166-1:2020, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*.
- ISO 3166-2:2020, *Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code*.
- ISO 10126-2:1991, *Banking – Procedures for message encipherment (wholesale) – Part 2: DEA algorithm*.
- ISO 22028-2:2013, *Photography and graphic technology — Extended colour encodings for digital image storage, manipulation and interchange — Part 2: Reference output medium metric RGB colour image encoding (ROMM RGB)*.
- ISO 15076-1:2010, *Image technology colour management – Architecture, profile format and data structure – Part 1: Based on ICC.1:2010*.
- ISO/IEC 23001-1:2006, *Information technology – MPEG systems technologies – Part 1: Binary MPEG format for XML*.
- ISO/IEC 21122-1:2019, *Information technology – JPEG XS low-latency lightweight image coding system – Part 1: Core coding system*.
- ISO/IEC 60559:2020, *Information technology — Microprocessor Systems — Floating-Point arithmetic*.
- ANSI X9.30.2:1997, *Public Key Cryptography using Irreversible Algorithms – Part 2: The Secure Hash Algorithm (SHA-1)*.
  https://infostore.saiglobal.com/en-us/standards/ANSI-X9-30-2-1997-1934_SAIG_ABA_ABA_5034/#
- Federal Information Processing Standard Publication (FIPS PUB) 186-4 (2013), *Digital Signature Standard (DSS)*.
  https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
- PIMA 7667:2001, *Photography - Electronics still picture imaging - Extended sRGB color encoding e-sRGB*.
  https://www.imaging.org/ItemDetail?iProductCode=PIMA7667-2001&Category=DOWNLD_PDF&WebsiteKey=6d978a6f-475d-46cc-bcf2-7a9e3d5f8f82
- W3C Recommendation. *Extensible Markup Language (XML 1.0)*, fifth edition (26 November 2008).
  https://www.w3.org/TR/xml/
- W3C Recommendation. *Namespaces in XML*, (14 January 1999).
  https://www.w3.org/TR/1999/REC-xml-names-19990114/
- W3C Recommendation. *XML Schema Part 1: Structures*, second edition (28 October 2004).
  http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
- W3C Recommendation. *XML Schema Part 2: Datatypes*, second edition (28 October 2004).
  http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. The definitions defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause 3 also apply to this Recommendation | International Standard, except for the terms decomposition level, sub-band and resolution, which are redefined in this clause.

3.1 **attribute**: XML construct that is a name-value pair extending or qualifying the meaning of an element.

3.2 **cell**: Optional subdivision of a tile used for low-memory encoding and decoding.

3.3 **component collection**: A subset of intermediate components used as inputs to a multiple component transformation stage, and a subset of intermediate components obtained as outputs from a multiple component transformation stage, where the subset's constituent components can occur in an arbitrary order, i.e., permuted with respect to their order of appearance in the set of input or output intermediate components.

3.4 **component reconstruction array**: General term that refers to any of the following; decorrelation transformation array, dependency transformation array or offset array.

3.5 **compositing**: Act of combining two compositing layers into a single, non-redundant set of image channels.

3.6 **compositing layer**: Set of non-redundant channels drawn from one or more codestreams that shall be treated as a group.

3.7 **deadzone**: Interval within which all sub-band coefficients are quantized to 0.

3.8 **decomposition level**: Collection of sub-bands where each coefficient has the same spatial impact or span with respect to the original samples, including the LL, LH, HL, HH, LX, HX, XL and XH sub-band splits out of decomposition sublevels.

3.9 **decomposition sub-level**: Collection of sub-bands that result from splits of a sub-band from a lower decomposition sub-level or splits of either LL, LX or XL sub-bands from a higher decomposition level.

3.10 **decorrelation transformation array**: Array of coefficients that maps the input components of a component collection to the output components of the collection via a multiple component decorrelation transformation.

3.11 **dependency transformation array**: Array of coefficients that maps the input components of a component collection to the output components of the collection via a multiple component dependency transformation.

3.12 **element**: XML construct that consists of a start tag and an end tag with data enclosed within.

3.13 **HX sub-band**: Sub-band obtained by forward horizontal high-pass analysis filtering and no vertical analysis filtering, and which contributes to reconstruction with inverse horizontal high-pass synthesis filtering and no vertical synthesis filtering.

3.14 **intermediate component**: Single two-dimensional array of data involved in a stage of a multiple component transformation.

3.15 **JPX baseline**: Subset of the JPX file format that uses codestream conforming to Rec. ITU-T T.800 | ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2 only.

3.16 **JPX baseline reader**: Application that correctly interprets all files that conform to the definition of a JPX baseline file.

3.17 **JPX file**: Name of file in the file format described in this Recommendation | International Standard. Structurally, a JPX file is a contiguous sequence of boxes.

3.18 **LX sub-band**: Sub-band obtained by forward horizontal low-pass analysis filtering and no vertical analysis filtering, and which contributes to reconstruction with inverse horizontal low-pass synthesis filtering and no vertical synthesis filtering.

3.19 **metadata**: Additional data associated with the image data beyond the image data**.**

3.20 **namespace**: Collection of names, identified by a URI, that allows XML documents of different sources to use the same element names within a single document to avoid element name conflicts**.**

3.21 **offset array**: Array of coefficients containing offsets which are added to intermediate components during multiple component transformation of a component collection.

3.22 **reconstructed image component**: Set of output intermediate components from the final transformation stage in the inverse multiple component transformation process.

3.23 **rendered result**: Result generated by combining the compositing layers in the JPX file, either by composition or animation.

3.24 **resolution**: Spatial relation of samples to a physical space.

NOTE – In this Recommendation | International Standard, the decomposition levels of the wavelet transformation create resolutions that differ by powers of two in either just horizontal, just vertical or both horizontal and vertical directions. The last (highest) decomposition level includes either an LL, LX or XL sub-band which is considered to be a lower resolution. Therefore, there is one more resolution level than decomposition levels.

3.25 **sub-band**: Group of transformation coefficients resulting from the sequence of low-pass and high-pass filtering operations, either just horizontally, just vertically or both horizontally and vertically.

3.26 **spatially reconstructed component**: Component which has been extracted from the codestream and passed through the decoding and inverse wavelet transformation process as specified by this Recommendation | International Standard, such that the set of spatially reconstructed components is the set of input components to the first transformation stage in the inverse multiple component transformation process.

3.27 **transformation stage**: Set of component collections and associated multiple component transformations.

3.28 **visual masking**: Mechanism where artefacts are masked by the image acting as a background signal.

3.29 **XH sub-band**: Sub-band obtained by no forward horizontal analysis filtering and vertical high-pass analysis filtering, and which contributes to reconstruction with vertical high-pass synthesis filtering and no inverse horizontal synthesis filtering.

3.30 **XL sub-band**: Sub-band obtained by no forward horizontal analysis filtering and vertical low-pass analysis filtering, and which contributes to reconstruction with vertical low-pass synthesis filtering and no inverse horizontal synthesis filtering.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply. The abbreviations defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause 4 also apply to this Recommendation | International Standard.

ADS	Arbitrary Decomposition Styles
ATK	Arbitrary Transformation Kernels
CBD	Component Bit depth Definition
DCO	variable DC Offset
DFS	Downsample Factor Styles
DPI	Dots Per Inch
IPR	Intellectual Property Rights
MCC	Multiple Component transformation Collection
MCO	Multiple Component transformation Ordering
MCT	Multiple Component Transformation definition
NLT	Non-Linearity point Transformation
UUID	Universal Unique Identifier
VMS	Visual Masking

5 Conventions

In this Recommendation | International Standard, the word "shall" is used to express mandatory requirements for conformance to this Specification. When used to express a mandatory constraint on the values of syntax elements or the values of variables derived from these syntax elements, it is the responsibility of the encoder to ensure that the constraint is fulfilled. The word "may" is used to refer to behaviour that is allowed, but not necessarily required. The word "should" is used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Specification.

For historical reasons, in this Specification, the convention of listing all informative annexes after the last normative annex is not followed strictly to minimize cross referencing issues in external texts.

6 General description

The purpose of this clause is to give an overview of this Recommendation | International Standard. Terms defined in previous clauses in this Recommendation | International Standard will also be introduced. (Terms defined in clauses 3 and 4 in Rec. ITU-T T.800 | ISO/IEC 15444-1 continue to apply in this Recommendation | International Standard.)

This Recommendation | International Standard defines a set of lossless (bit-preserving) and lossy compression methods for coding continuous-tone, bi-level, grey-scale, colour digital still images, or multi-component images. This set of methods

extends the elements in the core coding system described in Rec. ITU-T T.800 | ISO/IEC 15444-1. Extensions which pertain to encoding and decoding are defined as procedures which may be used in combination with the encoding and decoding processes described in Rec. ITU-T T.800 | ISO/IEC 15444-1. Each encoding or decoding extension shall only be used in combination with particular coding processes and only in accordance with the requirements set forth herein. These extensions are backward compatible in the sense that decoders which implement these extensions will also support configuration subsets that are currently defined by Rec. ITU-T T.800 | ISO/IEC 15444-1. This Recommendation | International Standard also defines extensions to the compressed data format, i.e., interchange format and the abbreviated formats.

6.1 Extensions specified by this Recommendation | International Standard

The following extensions are specified in this Recommendation | International Standard.

6.1.1 Syntax

An extension of the code stream syntax is described in Annex A. This extension provides all the codestream signalling in this Recommendation | International Standard. Further, it anticipates signalling needed for future specifications that include this Recommendation | International Standard as a normative reference. In addition to the codestream syntax defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, the following capabilities are supported: variable DC offset, variable scalar quantization, trellis coded quantization, visual masking, arbitrary decomposition, arbitrary transformation kernels, single sample overlap, multiple component transformations, non-linear transformation, arbitrary regions of interest. These extended markers conform to the same rules as the syntax in Rec. ITU-T T.800 | ISO/IEC 15444-1.

6.1.2 Variable DC offset

An extension which provides for variable DC offset is described in Annex B. Variable DC offset may be used to generate a better data distribution for input to the ICT or RCT multi component transformation, defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, and/or the wavelet transformation. Images with very skewed sample distributions may benefit from a non-default DC offset.

6.1.3 Variable scalar quantization

An extension that provides for variable scalar quantization is described in Annex C. This extension allows smaller or larger deadzones to be used with the scalar quantizer. This technique may improve visual appearance of low level texture.

6.1.4 Trellis coded quantization

An extension of the quantization is described in Annex D. This extension provides for trellis coded quantization (TCQ). The TCQ algorithm applies spatial-varying scalar quantization to its input sequence by choosing one of four scalar quantizers for each sample. Quantizer indices from supersets of these quantizers along with quantizer transitions in the form of a trellis provide all information necessary to reconstruct TCQ encoded wavelet coefficients.

6.1.5 Visual masking

An extension which provides for visual masking is described in Annex E. Visual masking is a mechanism where artefacts are masked by the image acting as a background signal. The main goal is to improve the image quality, especially for displays. The first effect of this technique is to improve the image quality, where the improvement becomes greater as the image becomes more complex. The second main effect of this technique is that for a given fixed bit-rate, the image quality is more robust against variations in image complexity. This is accomplished at the encoder via an extended non-linearity interposed between the transformation stage and the quantization stage.

6.1.6 Arbitrary decomposition

An extension providing for arbitrary decomposition of the tile component is described in Annex F. This extension can control the bandpass extent of wavelet sub-bands and thus provide control over the decorrelation process in order to tune compression performance. This extension also allows for transcoding of other wavelet based compression algorithms into codestreams of this Recommendation | International Standard.

6.1.7 Arbitrary wavelet transformation

Extensions that provide for transformation of image tile components using user defined wavelet filters are described in Annexes G and H. Annex G describes whole sample filters while Annex H describes arbitrary filters.

6.1.8 Single sample overlap discrete wavelet transformations

Extensions providing for block based wavelet transformations are described in Annex I. These extensions consist of one method for tile based wavelet transformation without tiling artefacts and one method for cell-based wavelet transformation.

6.1.9 Multiple component transformations

An extension which provides for multiple component transformations is described in Annex J. This extension specifies two types of multiple component transformations:

1) A specification of a multiple component transformation which uses linear transformations of bands to reduce the correlation of each band. This is similar to the most common colour transformations.
2) A specification of a wavelet transformation along the component direction.

6.1.10 Non-linear transformation

Annex K specifies three non-linear point transformations that are used after decoding processes and inverse multiple component transformations to map reconstructed values back to their proper range. These transformations may be employed by encoders prior to multiple component transformation and encoding to increase compression efficiency. The first two transformations, gamma and look-up table (LUT) style non-linearities, can be used to perceptually flatten a scanner or sensor with a linear response, from 12 bits to 8 bits precision prior to compression. The third, which maps sign-magnitude numbers into a two's complement representation, can be used to represent floating point numbers.

6.1.11 Region of interest

An extension which provides for Region of interest coding using the scaling based method is described in Annex L. The scaling based method provides for having different scaling values for different regions of interest. The extension also specifies how to generate the masks in the wavelet domain that describe the set of wavelet coefficient belonging to each Region of interest.

6.1.12 File format

An extension of the file format is described in Annex M. This extension provides for the exchange of compressed image files between application environments. This extension is an optional file format, called JPX, that applications may choose to use to contain JPEG 2000 compressed image data. JPX is an extension to the JP2 file format defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex I. The format:

1) specifies a binary container for both image and metadata;
2) specifies a mechanism to indicate image properties, such as the tone-scale or colour space of the image;
3) specifies a mechanism by which readers may recognize the existence of intellectual property rights information in the file;
4) specifies a mechanism by which metadata (including vendor specific information) can be included in files specified by this Recommendation | International Standard;
5) specifies a mechanism by which multiple codestreams can be combined into a single work, by methods such as compositing and animation.

6.1.13 Metadata definitions

Metadata definitions are described in Annex N. Metadata is additional information that is associated with the primary data (the image). In the context of this Specification, it is "additional data linked with the image data beyond the pixels which define the image". Metadata, to be most valuable for the owner(s) and user(s) of an image, needs to be consistently maintained throughout the image lifecycle. In today's environment of image editing applications, rapid transmission via the Internet, and high quality photographic printers, the lifecycle of a digital image may be very long as well as complex.

6.1.14 Progression order extensions

Beyond the five progression orders specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, this Recommendation | International Standard introduces a position-resolution level-component-layer progression order, that is specified in Annex I.

6.2 Relation between extensions

The relations, at decoder side, between the extensions listed above are given in Figure 6-1. Technologies described in Rec. ITU-T T.800 | ISO/IEC 15444-1 are indicated in the boxes.

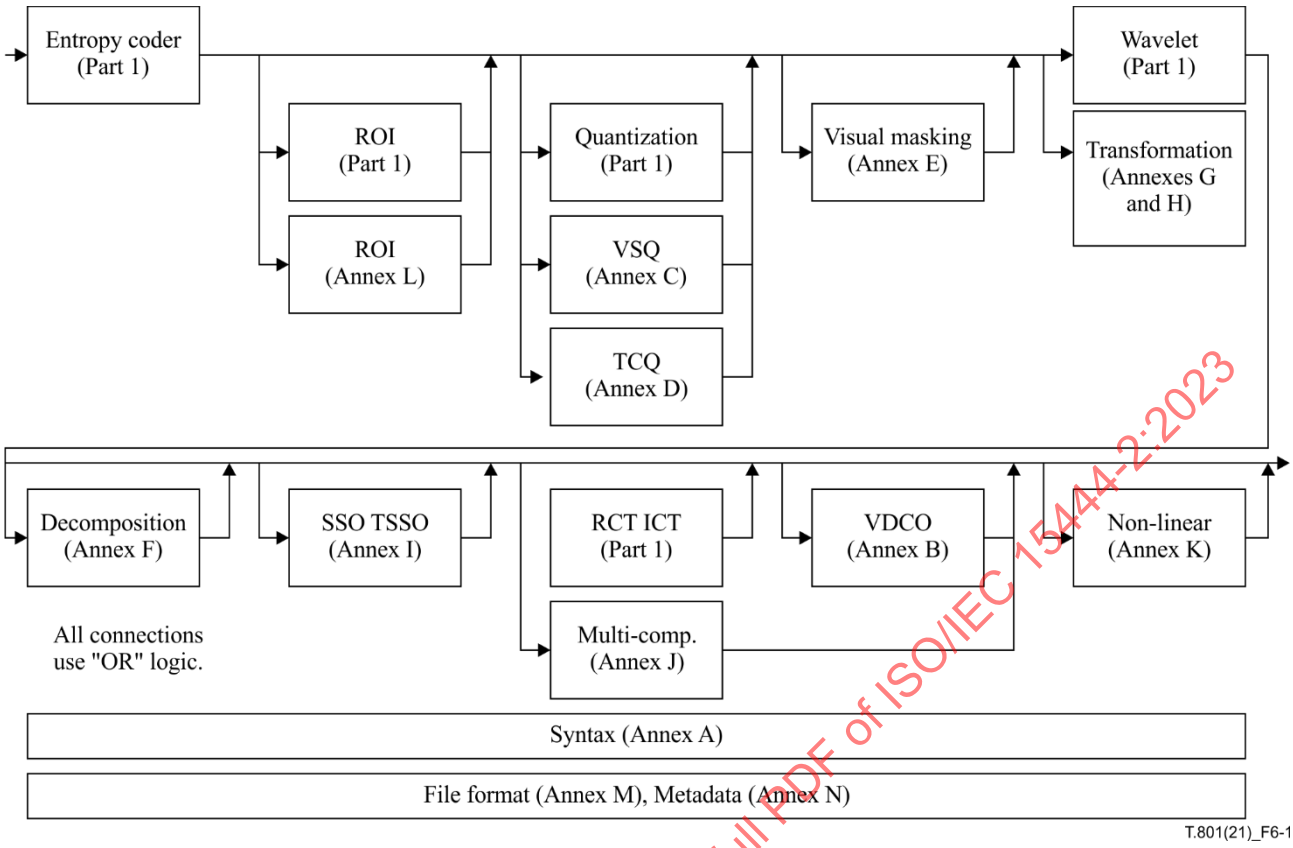

Figure 6-1 – Decoder block diagram

Annex A

Compressed data syntax, extension

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they define an output that alternative implementations shall duplicate.

This annex specifies the marker and marker syntax extensions to the Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A syntax. These markers provide all the codestream signalling in this Recommendation | International Standard. Further, it anticipates signalling needed for future specifications that include this Recommendation | International Standard as a normative reference.

All positive (unsigned) integer values of parameters are placed in the codestream as unsigned integers. All other (signed) integers are expressed in two's complement. Unless otherwise indicated, all values are in big endian order.

In the tables of this annex, the symbol "r" denotes bits that are reserved, and the symbol "x" denotes bits whose value can be either 0 or 1.

For codestreams conforming to this Recommendation | International Standard alone, the value of each bit denoted with an "r" shall be 0.

NOTE – The behaviour of implementations that conform to this Recommendation | International Standard is left unspecified when processing a codestream where the value of any bit denoted with an "r" is not 0.

A.1 Extended capabilities

The syntax in this annex supports the extensions in this Recommendation | International Standard. These marker segments conform to the same rules as the syntax in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A. The addition of parameter values to some marker segments in Rec. ITU-T T.800 | ISO/IEC 15444-1 and the addition of new marker segments signals the information specific to the extensions in this Recommendation | International Standard. In every marker segment the first two bytes after the marker shall be an unsigned value that denotes the length in bytes of the marker segment parameters (including the two bytes of this length parameter but not the two bytes of the marker itself). When a marker segment that is not specified in this Recommendation | International Standard or in Rec. ITU-T T.800 | ISO/IEC 15444-1 is encountered in a codestream, the decoder shall use the length parameter to discard the marker segment. Table A.1 shows the marker segments affected by this Recommendation | International Standard.

Table A.1 – Syntax support for extensions

Extension	Extended Rec. ITU-T T.800 | ISO/IEC 15444-1 marker segments	New marker segments
All extensions	SIZ	–
Variable DC offset	–	DCO
Variable scalar quantization	QCD, QCC, SOT	QPD, QPC
Trellis coded quantization	QCD, QCC, SOT	QPD, QPC
Visual masking	–	VMS
Single sample offset transform	SIZ, COD, COC	–
Arbitrary decomposition styles	COD, COC	DFS, ADS
Arbitrary transformation kernels	COD, COC	ATK
Multiple component transformation	COD	CBD, MCT, MCC, MCO
Non-linearity point transformation	–	NLT
Arbitrary shaped region of interest	RGN	–
Precinct length, tile-part header	–	RLT
Progression order extensions	COD, POC	–

A.2 Extensions to Rec. ITU-T T.800 | ISO/IEC 15444-1 marker segment parameters

This clause describes the extensions to marker segments defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A.

A.2.1 Image and tile size (SIZ), extended

The capability parameter Rsiz of Rec. ITU-T T.800 | ISO/IEC 15444-1 denotes capabilities that a decoder needs to properly decode a codestream. Table A.2 defines Rsiz values of this parameter specific to this Recommendation | International Standard.

Table A.2 – Capability Rsiz parameter, extended

Value (bits) MSB LSB	Capability
1x00 xxxx xxxx xxxx	At least one of the extended capabilities specified in this Recommendation | International Standard is present
1x00 xxx0 xxxx xxx1	Variable DC offset capability is required to decode this codestream[a) b)]
1x00 xxxx xxxx xx1x	Variable scalar quantization capability is required to decode this codestream[a)]
1x00 xxxx xxxx x1xx	Trellis coded quantization capability is useful to decode this codestream[c)]
1x00 xxxx xxxx 1xxx	Visual masking capability is useful to decode this codestream[c)]
1x00 xxxx xxx1 xxxx	Single sample overlap capability is required to decode this codestream[a)]
1x00 xxxx xx1x xxxx	Arbitrary decomposition style capability is required to decode this codestream[a)]
1x00 xxxx x1xx xxxx	Arbitrary transformation kernel capability is required to decode this codestream[a)]
1x00 xxxx 1xxx xxxx	Whole sample symmetric transformation kernel capability is required to decode this codestream[a)]
1x00 xxx1 xxxx xxxx	Multiple component transformation capability is required to decode this codestream[a)]
1x00 xx1x xxxx xxxx	Non-linear point transformation capability is useful to decode this codestream[c)]
1x00 x1xx xxxx xxxx	Arbitrary shaped region of interest capability is required to decode this codestream[a)]
1x00 1xxx xxxx xxxx	Precinct-dependent quantization is required to decode this codestream[a)]
	All other values reserved

a) "Required to decode" implies that no useful data or image can be reconstructed without the use of this capability.

b) Shall not be used with the multiple component transformation.

c) "Useful to decode" implies that use of this capability would improve the quality of the reconstructed data or image; however, the data or image may be decoded without its use.

d) The 2 MSBs of Rsiz are used as follows:
10 – Rec. ITU-T T.801 | ISO/IEC 15444-2 capabilities
11 – Rec. ITU-T T.801 | ISO/IEC 15444-2 capabilities extended via a CAP marker segment.

A.2.2 Start of tile-part (SOT) extended

If Rsiz indicates that the precinct-dependent quantization capability is used, then the SOT marker segment from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A, is extended to allow 1-65535 tile-parts. Table A.3 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.5, and Table A.4 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.6.

Table A.3 – Start of tile-part parameter values, extended

Parameter	Size (bits)	Values
SOT	16	0xFF90
Lsot	16	10
Isot	16	0 to 65534
Psot	32	0, or 14-(2^{32}–1)
TPsot	16	0 to 65535
TNsot	16	Table A.4

Table A.4 – Number of tile-parts, TNsot, parameter value, extended

Value	Number of tile-parts
0	Number of tile-parts of this tile in the codestream is not defined in this header
1-65 535	Number of tile-parts of this tile in the codestream

A.2.3 Coding style (COD, COC), extended

Geometric manipulation is enabled with two bits in the Scod parameter shown in Table A.5.

Block coder extensions are enabled with 1 bit in the Scod parameter, as shown in Table A.5. When this bit is set, the COD marker segment is extended to include a 16-bit parameter SXcod that follows SPcod as shown in Figure A.1bis. The meaning of SXcod is defined by Table A.11bis.

Available progression orders are extended as shown in Table A.7bis, which replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.16.

If the position-resolution level-component-layer progression order is used, bit-14 of Ccap[2] shall be set (see Table A.49).

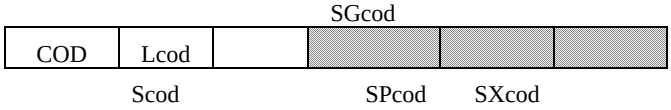

Figure A.1bis – Coding style default syntax

If the Rsiz field of the SIZ marker segment indicates that the multiple component transformations are used, then Table A.8 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.17.

If the Rsiz field of the SIZ marker segment indicates that the arbitrary transformation kernels are used, then Table A.10 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.20.

If the Rsiz field of the SIZ marker segment indicates that the single sample overlap transformation capability is necessary, then an extra 8 bit field is added to Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.13 after the transformation field as shown in Table A.7. The SSO values are found in Table A.11.

If the Rsiz field of the SIZ marker segment indicates that the arbitrary decomposition styles are used then the maximum number of decomposition levels field definitions are found in Table A.9 rather than in Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.13. This is shown in Table A.7.

Table A.5 – Coding style parameter values for the Scod parameter

Values (bits) MSB LSB	Coding style
rrxx xxx0	Entropy coder, precincts with PPx = 15 and PPy = 15
rrxx xxx1	Entropy coder with precincts defined below
rrxx xx0x	No SOP marker segments used
rrxx xx1x	SOP marker segments may be used
rrxx x0xx	No EPH marker used
rrxx x1xx	EPH marker shall be used
rrxx 0xxx rrxx 1xxx	Offset in the horizontal dimension, $z_x = 0$ (CBAP) Offset in the horizontal dimension, $z_x = 1$
rrx0 xxxx rrx1 xxxx	Offset in the vertical dimension, $z_y = 0$ (CBAP) Offset in the vertical dimension, $z_y = 1$
rr0x xxxx	Block coder extensions not used
rr1x xxxx	Block coder extensions defined by SXcod
	All other values reserved

Table A.6 – Coding style parameter values of the SGcod parameter

Parameters (in order)	Size (bits)	Values	Meaning of SGcod values
Progression order	8	Table A.7bis	Progression order
Number of layers	16	1 to 65535	Number of layers
Multiple component transformation	8	Table A.8	Multiple component transformation usage

Table A.7 – Coding style parameter values of the SPcod and SPcoc parameters, extended

Parameters (in order)	Size (bits)	Values (bits) MSB LSB	Meaning of SPcod values
Maximum number of decomposition levels	8	Table A.9	Decomposition mapping and levels
Code-block width	8	Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.18	Code-block width exponent offset value, *xcb*
Code-block height	8	Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.18	Code-block height exponent offset value, *ycb*
Code-block style	8	Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.19	Style of the code-block coding passes
Transform	8	Table A.10	Wavelet transformation used
SSO overlap	16	Table A.11	SSO overlap values
Precinct size	variable	Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.21	If Scod or Scoc = xxxx xxx0, this parameter is not present, otherwise this indicates precinct width and height. The first parameter (8 bits) corresponds to the N_LLL, N_LLX, or N_LXL sub-band. Each successive parameter corresponds to each successive resolution in order.

Table A.7bis – Progression order for the SGcod and Ppoc parameters

Values (bits) MSB LSB	Multiple component transformation type
0000 0000	Layer-resolution level-component-position progression
0000 0001	Resolution level-layer-component-position progression
0000 0010	Resolution level-position-component-layer progression
0000 0011	Position-component-resolution level-layer progression
0000 0100	Component-position-resolution level-layer progression
0000 0101	Position-resolution level-component-layer progression
	All other values reserved.

Table A.8 – Multiple component transformation for the SGcod parameters

Values (bits) MSB LSB	Multiple component transformation type
0000 0000	No multiple component transformation specified.
0000 0001	Component transformation used on components 0, 1, 2 for coding efficiency. Irreversible component transformation used with irreversible filters. Reversible component transformation used with reversible filters.
0000 0x10	Array-based multiple component transformation is used. May be combined with wavelet-based multiple component transformation.
0000 01x0	Wavelet-based multiple component transformation is used. May be combined with array-based multiple component transformation.
	All other values reserved.

Table A.9 – Decomposition for the SPcod and SPcoc parameters, extended

Values (bits) MSB LSB	Decomposition type
0000 0000 to 0010 0000	Number of levels of wavelet decomposition, dyadic decomposition, N_L (Zero implies no transform.).
1000 0001 to 1111 1111	If in the main header: Downsampling factor style index value (1 to 127). (See clause A.3.3.) If in tile-part header: Arbitrary decomposition style index value (1 to 127). (See clause A.3.4.)
	All other values reserved

Table A.10 – Transformation for the SPcod and SPcoc parameters, extended

Values (bits) MSB LSB	Transformation type
0000 0000	Rec. ITU-T T.800 | ISO/IEC 15444-1 9-7 irreversible wavelet transform.
0000 0001	Rec. ITU-T T.800 | ISO/IEC 15444-1 5-3 reversible wavelet transform.
0000 0010 to 1111 1111	Arbitrary transformation kernel definition index value (2 to 255). Definitions are found in the appropriate ATK marker segment (see clause A.3.5).

Table A.11 – SSO parameters, extended

Values (bits) MSB LSB	SSO size (see Annex I)
0xxx xxxx xxxx xxxx 1xxx xxxx xxxx xxxx	SSO not used SSO used
x0xx xxxx xxxx xxxx x1xx xxxx xxxx xxxx	TSSO not used for every tile in the image (shall be the same in every COD marker segment) TSSO used for every tile in the image (shall be the same in every COD marker segment)
xx0x xxxx xxxx xxxx xx1x xxxx xxxx xxxx	Vovlp = 0 (shall be the same in every COD marker segment) Vovlp = 1 (shall be the same in every COD marker segment)
xxx0 xxxx xxxx xxxx xxx1 xxxx xxxx xxxx	Hovlp = 0 (shall be the same in every COD marker segment) Hovlp = 1 (shall be the same in every COD marker segment)
xxxx 0xxx xxxx xxxx x1xx 1xxx xxxx xxxx	TBDWT is not used (shall be the same in every COD marker segment) TBDWT is used (shall be the same in every COD marker segment)
xxxx xxxx xxxx 0000 to xxxx xxxx xxxx 1111	Cell width exponent value, XC = 2^{value}
xxxx xxxx 0000 xxxx to xxxx xxxx 1111 xxxx	Cell height exponent value, YC = 2^{value}
	All other values reserved

Table A.11bis – SXcod parameter

Values (bits) MSB LSB	Coding style
rrrr rrrr rrrr rr00	No modifications to selective arithmetic coder bypass mode.
rrrr rrrr rrrr rr01	Selective arithmetic coder bypass modified to start from the 2nd significant bit-plane if bit 0 of SPcod/SPcoc set, as specified in Annex P.
rrrr rrrr rrrr rr10	Selective arithmetic coder bypass modified to start from the 3rd significant bit-plane if bit 0 of SPcod/SPcoc set, as specified in Annex P.
rrrr rrrr rrrr rr11	Selective arithmetic coder bypass modified to start from the 4th significant bit-plane if bit 0 of SPcod/SPcoc set, as specified in Annex P.
	All other values reserved.

A.2.3bis Progression order change (POC), extended

Table A.11ter replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.32.

Available progression orders are extended as shown in Table A.7bis, which replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.16.

If the position-resolution level-component-layer progression order is used, bit-14 of $Ccap^2$ shall be set (see Table A.49).

Table A.11ter – Progression order change, tile parameter values

Parameter	Size (bits)	Values
POC	16	0xFF5F
Lpoc	16	9 to 65535
$RSpoc^i$	8	0 to 33
$CSpoc^i$	8 16	0 to 255; if Csiz < 257 0 to 16383; Csiz ≥ 257
$LYEpoc^i$	16	1 to 65535
$REpoc^i$	8	$(RSpoc^i + 1) – 33$
$CEpoc^i$	8 16	$(CSpoc^i + 1)$ to 255, 0; if Csiz < 257 $(CSpoc^i + 1)$ to 16384, 0; Csiz ≥ 257 (0 is interpreted as 256)
$Ppoc^i$	8	Table A.7bis

A.2.4 Quantization (QCD, QCC), extended

If Rsiz indicates that the variable scalar quantization (see Annex C) capability is used, then the deadzone adjustment is signalled in modified QCD and QCC marker segments from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A. If Rsiz indicates that trellis coded quantization is used, then these values are also signalled via the extended QCD and QCC marker segments from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A. This shall only be used with irreversible transformations.

Table A.12 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.28, and Table A.13 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.30.

Table A.12 – Quantization default values for the Sqcd, Sqcc, Sqpd, and Sqpc parameters, extended

Values (bits) MSB LSB	Quantization style	SPqxx size (bits)	SPqxx usage
xxx0 0000	No quantization	8	Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A
xxx0 0001	Scalar derived (values signalled for N_LLL sub-band only). Use Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-5.	16	Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A
xxx0 0010	Scalar expounded (values signalled for each sub-band). There are as many step sizes signalled as there are sub-bands.	16	Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A
xxx0 0011	Variable deadzone and scalar derived (values signalled for N_LLL sub-band only). Use Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-5.	32	Table A.13
xxx0 0100	Variable deadzone derived and scalar expounded (values signalled for each sub-band). There are as many step sizes signalled as there are sub-bands.	16	Table A.14 then Table A.15
xxx0 0101	Variable deadzone and scalar expounded (values signalled for each sub-band). There are as many step sizes signalled as there are sub-bands.	32	Table A.13
xxx0 1001	Trellis coded quantization derived (values signalled for N_LLL sub-band only). Use Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-5.	16	Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A
xxx0 1010	Trellis coded quantization expounded (values signalled for each sub-band). There are as many step sizes signalled as there are sub-bands.	16	Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A
000x xxxx to 111x xxxx	Number of guard bits 0-7.		
	All other values reserved.		

Table A.13 – Quantization values (irreversible transformation only), extended

Values (bits) MSB LSB	Deadzone adjustment and quantization step size values
0000 0000 0000 0000 xxxx xxxx xxxx xxxx to 1111 1111 1111 1111 xxxx xxxx xxxx xxxx	Variable deadzone, num_nz$_b$, value –32768 to 32767 (see Equation C-1)
xxxx xxxx xxxx xxxx xxxx x000 0000 0000 to xxxx xxxx xxxx xxxx xxxx x111 1111 1111	Mantissa, μ_b, of the quantization step size value 0 to 2047 (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-3)
xxxx xxxx xxxx xxxx 0000 0xxx xxxx xxxx to xxxx xxxx xxxx xxxx 1111 1xxx xxxx xxxx	Exponent, ε_b, of the quantization step size value 0 to 31 (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-3)

Table A.14 – SPqcd, SPqcc, SPqpd, and SPqpc parameters (irreversible transformation only), extended

Values (bits) MSB LSB	Deadzone adjustment values (one for each sub-band)
0000 0000 0000 0000 to 1111 1111 1111 1111	First two bytes of SPqcx are the deadzone adjustment, num_nz$_b$, value –32768 to 32767 (see Equation C-1)

Table A.15 – SPqcd, SPqcc, SPqpd, and SPqpc parameters (irreversible transformation only), extended

Values (bits) MSB LSB	Quantization step size values
xxxx x000 0000 0000 to xxxx x111 1111 1111	After first two bytes of SPqcx are the mantissa, μ_b, of the quantization step size value 0 to 2047 (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-3)
0000 0xxx xxxx xxxx to 1111 1xxx xxxx xxxx	After first two bytes of SPqcx are the exponent, ε_b, of the quantization step size value 0 to 31 (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-3)

A.2.5 Region of interest marker (RGN), extended

If Rsiz indicates that an arbitrary region of interest is used (see Annex L), then the description of a coefficient shift and a mask are signalled in a modified RGN marker segment from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A. Table A.16 replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.25. If there is RGN marker segment in the main header with a Srgn = 0, there shall not be any RGN marker segment anywhere in the codestream with a non-zero Srgn value for the component given by the corresponding Crgn value. Likewise, if there is RGN marker segment in the main header with a non-zero Srgn value, there shall not be any RGN marker segment anywhere in the codestream with a Srgn = 0 for the component given by the corresponding Crgn value.

When used in both the main header and the first tile-part header, the RGN in the first tile part header overrides the main for that tile. Also, an RGN specifying a single component (Crgn ≠ 65 535) overrides on specifying all components (Crgn = 65 535). Thus, the order of precedence is the following:

Tile-part RGN (Crgn ≠ 65 535) > Tile-part RGN (Crgn = 65 535) > Main RGN (Crgn ≠ 65 535) > Main RGN (Crgn = 65 535)

where the "greater than" sign, >, means that the greater overrides the lesser marker segment.

Table A.16 – Region-of-interest parameter values for the Srgn parameter

Values	ROI style (Srgn)	SPrgn usage
0	Implicit ROI (maximum shift)	Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.26
1	Arbitrary region of interest, rectangle	Table A.18
2	Arbitrary region of interest, ellipse	Table A.18
	All other values reserved	

Table A.17 – Component index parameter value for the Crgn parameter

Parameter	Size (bits)	Values	Components index parameter
Component	16	0 to 16383 16394 to 65354 65535	Specifies component to which these region of interest descriptions apply Reserved Region of interest descriptions apply to all components

Table A.18 – Region-of-interest values from SPrgn parameter (Srgn = 1 or Srgn = 2)

Parameter	Size (bits)	Values	Meaning of SPrgn parameter
Binary shift	8	0 to 255	Binary shifting of coefficients in the region of interest above the background.
XArgn (left)	32	0 to (2^{32} –1)	Horizontal reference grid point from the origin of the first point. (In the case of the ellipse, Srgn = 2, this value shall not exceed the width of the image.)
YArgn (top)	32	0 to (2^{32} –1)	Vertical reference grid point from the origin of the first point. (In the case of the ellipse, Srgn = 2, this value shall not exceed the height of the image.)
XBrgn (right)	32	0 to (2^{32} –1)	Horizontal reference grid point from the origin of the second point.
YBrgn (bottom)	32	0 to (2^{32} –1)	Vertical reference grid point from the origin of the second point.

A.3 Extended marker segments

Table A.19 lists the markers specified in this Recommendation | International Standard.

Table A.19 – List of markers and marker segments

	Symbol	Code	Main header[a)]	Tile-part header[a)]
Variable DC offset	DCO	0xFF70	optional	optional
Visual masking	VMS	0xFF71	optional	optional
Downsampling factor style	DFS	0xFF72	optional	optional
Arbitrary decomposition style	ADS	0xFF73	optional	optional
Arbitrary transformation kernels	ATK	0xFF79	optional	optional
Component bit depth	CBD	0xFF78	optional	optional
Multiple component transformation definition	MCT	0xFF74	optional	optional
Multiple component transformation collection	MCC	0xFF75	optional	optional
Multiple component transformation ordering	MCO	0xFF77	optional	optional
Non-linearity point transformation	NLT	0xFF76	optional	optional
Quantization default, precinct	QPD	0xFF5A	optional	optional
Quantization component, precinct	QPC	0xFF5B	optional	optional
Precinct length, tile-part header	RLT	0xFF95	no	optional
a) "optional" means it may be used in the header if this extension is used. "no" means it shall never be present in the header.				

A.3.1 Variable DC offset (DCO)

Function: Describes the variable DC offset for every component.

Usage: Present only if the variable DC offset capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main and first tile-part header of a given tile. Optional in both the main and tile-part headers. No more than one shall appear in any header. If present in the main header, it describes the variable DC offset for every component in every tile. If present in the first tile-part header of a given tile, it describes the variable DC offset for every component in that tile only. When used in both the main header and the first tile-part header, the DCO in the first tile part header overrides the main for that tile. Thus, the order of precedence is the following:

Tile-part DCO > Main DCO

where the "greater than" sign, >, means that the greater overrides the lesser marker segment.

Shall not be used with the multiple component transformation.

Length: Variable depending on the number of components. The syntax is depicted in Figure A.1.

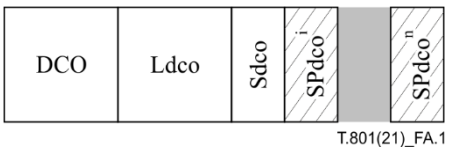

Figure A.1 – Variable DC offset syntax

DCO: Marker code. Table A.20 shows the size and parameter values for coding style component marker segment.

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

$$Ldco = \begin{cases} 3 + Csiz & Sdco = 0 \\ 3 + 2 \cdot Csiz & Sdco = 1 \\ 3 + 4 \cdot Csiz & Sdco = 2 \\ 3 + 8 \cdot Csiz & Sdco = 3 \end{cases} \quad \text{(A-1)}$$

where Csiz is from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A.

NOTE – If Ldco were to be larger than 65 535, then the DCO marker segment cannot be used. Instead, multiple component transformation functionality could be used.

Sdco: Variable DC offset type definition.

SPdco^{i}: Variable DC offset for the ith component. There is one SPdco parameter for every component in the image.

Table A.20 – Variable DC offset parameter values

Parameter	Size (bits)	Values
DCO	16	0xFF70
Ldco	16	5 to 32770
Sdco	8	Table A.21
SPdco^{i}	variable	Table A.21

Table A.21 – Variable DC offset parameter values for the Sdco parameter

Values (bits) MSB LSB	Offset type definition
0000 0000	Offsets are 8 bit unsigned integers
0000 0001	Offsets are 16 bit signed integers
0000 0010	Offsets are 32-bit binary floating point (ISO/IEC/IEEE 60559)
0000 0011	Offsets are 64-bit binary floating point (ISO/IEC/IEEE 60559)
	All other values reserved

IEEE numbers as used in the Rec. ITU-T T.801 | ISO/IEC 15444-2 codestream have to be written in big-endian order, using the following bit-assignments for encoding the floating point numbers:

SEEE EEEE EMMM MMMM MMMM MMMM MMMM MMMM

for single precision IEEE numbers, where S = sign bit, E = exponent bits, M = mantissa bits

SEEE EEEE EEEE MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM MMMM

for double precision IEEE numbers, where S = sign bit, E = exponent bits, M = mantissa bits

NOTE 2 – This encoding requires typically an endian-swap on little-endian machines, and is typically the native encoding for big-endian machines.

A.3.2 Visual masking (VMS)

Function: Describes the visual masking for all tile-components in the image or tile.

Usage: Present only if the visual masking capability bit in the Rsiz parameter (see clause A.2.1) has the value one. Optionally used in the main and/or the first tile-part header of a given tile. No more than one VMS marker segment for a component shall appear in any header. When used in both the main header and the first tile-part header, the VMS marker segment in the first tile part header overrides the one in the main header for that tile. A VMS marker segment specifying a single component (Cvms ≠ 65 535) overrides on specifying all components (Cvms = 65 535). Thus, the order of precedence is the following:

Tile-part VMS (Cvms ≠ 65 535) > Tile-part VMS (Cvms = 65 535) > Main VMS (Cvms ≠ 65 535) > Main VMS (Cvms = 65 535)

where the "greater than" sign, >, means that the greater overrides the lesser marker segment.

Length: Fixed. The syntax is depicted in Figure A.2.

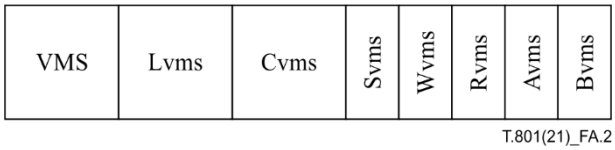

Figure A.2 – Visual masking syntax

VMS: Marker code. Table A.22 shows the size and parameter values for coding style, default marker segment.

Lvms: Length of marker segment in bytes (not including the marker). Fixed at 7 bytes.

Cvms: The index of the component to which this marker segment applies. Could be all components.

Svms: Minimal resolution level and respect block boundaries flag.

Wvms: Window width variable, *win_width* (see E.6).

Rvms: Bits retained variable, *bits_retained* (see E.6).

Avms: Value of the numerator of the α parameter, α = *Avms*/128 (see E.6).

Bvms: Value of the numerator of the β parameter, β = *Bvms*/128 (see E.6).

Table A.22 – Visual masking parameter values

Parameter	Size (bits)	Values
VMS	16	0xFF71
Lvms	16	9
Cvms	16	Table A.23
Svms	8	Table A.24
Wvms	8	0 to 8
Rvms	8	0 to 255
Avms	8	0 to 255
Bvms	8	0 to 255

Table A.23 – Component parameter value for the Cvms parameter

Values	Component index parameter
0 to 16383 16394 to 65354 65535	Specifies component to which these region of interest descriptions apply Reserved Region of interest descriptions apply to all components

Table A.24 – Visual masking for the Svms parameters

Values (bits) MSB LSB	Visual masking parameters
x000 0000 to x001 0000	Minimum resolution level value, *minlevel* (0 to 32) (see E.6)
0xxx xxxx 1xxx xxxx	Variable *respect_block_boundaries* = 0 (see E.6) Variable *respect_block_boundaries* = 1 (see E.6)
	All other values reserved

A.3.3 Downsampling factor styles (DFS)

Function: Describes the arbitrary decomposition pattern for the lowest resolution sub-band for all tiles of a given component.

Usage: Present only if the custom decomposition style bit in the Rsiz parameter (see clause A.2.1) is a one value. Main header. Assigned to a component by an index in the main header COD or COC markers.

Length: Variable. The syntax is depicted in Figure A.3.

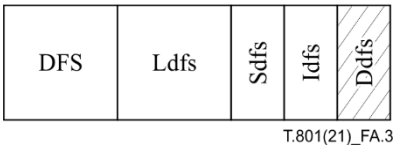

Figure A.3 – Downsampling factor styles syntax

DFS: Marker code. Table A.26 shows the size and values of the symbol and parameters for coding style, default marker segment.

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

$$Ldfs = 4 + \left\lceil \frac{Idfs}{4} \right\rceil \quad \text{(A-2)}$$

Sdfs: The index of this DFS marker segment. This marker segment is associated with a component via the parameter in the COD or COC marker segments found in the main header.

Idfs: Number of elements in the string defining the number of decomposition sub-levels.

Ddfs: String defining the number of decomposition sub-levels. The two bit elements are packed into bytes in big endian order. The final byte is padded to a byte boundary.

Table A.25 – Downsampling factor styles parameter values

Parameter	Size (bits)	Values
DFS	16	0xFF72
Ldfs	16	5 to 65535
Sdfs	16	0 to 15
Idfs	8	0 to 255
Ddfs	variable	String of elements

A.3.4 Arbitrary decomposition styles (ADS)

Function: Describes the arbitrary decomposition pattern for a tile-component or all tile-components within a single tile.

Usage: Present only if the custom decomposition style capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Shall not be used to describe the decomposition described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex F. Main and first tile-part header of a given tile. There may be up to 127 such marker segments with unique index values. If an index value is found in a tile-part header, then it is used instead of an ADS marker segment in the main header with the same index value. These are assigned to a particular tile-component via the parameter in the COD or COC marker segments found only in a specific tile-part header.

Length: Variable. The syntax is depicted in Figure A.4.

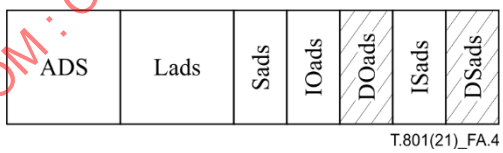

Figure A.4 – Arbitrary decomposition styles syntax

ADS: Marker code. Table A.26 shows the size and values of the symbol and parameters for coding style, default marker segment.

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

$$Lads = 5 + \left\lceil \frac{IOads+ISads}{4} \right\rceil \quad \text{(A-3)}$$

Sads: The index of this ADS marker segment. This marker segment is associated with a component via the parameter in the COD or COC marker segments found in that tile-part header.

IOads: Number of elements in the string defining the number of decomposition sub-levels.

DOads: String defining the number of decomposition sub-levels. The two bit elements are packed into bytes in big endian order. The final byte is padded to a byte boundary.

ISads: Number of elements in the string defining the arbitrary decomposition structure.

DSads: String defining the arbitrary decomposition structure. The two bit elements are packed into bytes in big endian order. The final byte is padded to a byte boundary.

Table A.26 – Arbitrary decomposition styles parameter values

Parameter	Size (bits)	Values
ADS	16	0xFF73
Lads	16	3 to 65535
Sads	8	1 to 127
IOads	8	0 to 255
DOads	variable	String of elements
ISads	8	0 to 255
DSads	variable	String of elements

A.3.5 Arbitrary transformation kernels (ATK)

Function: Describes a transformation kernel and an index that allows assignment to tile-components.

Usage: Present only if the arbitrary transformation kernel capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main and first tile-part header of a given tile. May be up to 254 marker segments in any header. A marker segment in the tile-part header with the same index as one in the main header overrides the main header marker segment. Table A.27 describes the arbitrary transformation parameter values.

Length: Variable. The syntax is depicted in Figure A.5.

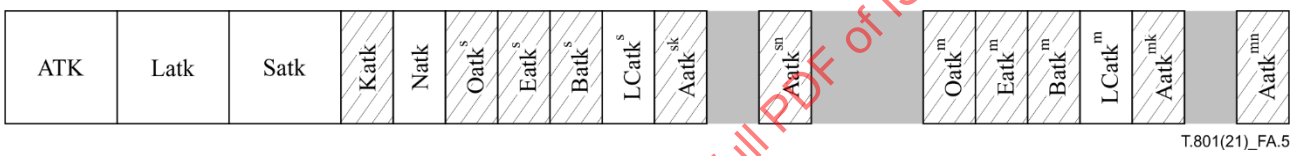

Figure A.5 – Arbitrary transformation default syntax

ATK: Marker code. Figure A.5 shows the size and values of the symbol and parameters for arbitrary transformation marker segment.

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

$$Latk = \begin{cases} 5 + 2Natk + sizeof(Coeff\_Typ)\left(1 + \sum_{s=0}^{Natk-1} LCatk^s\right) & WT\_Typ = IRR, Filt\_Cat = ARB \\ 5 + Natk + sizeof(Coeff\_Typ)\left(1 + \sum_{s=0}^{Natk-1} LCatk^s\right) & WT\_Typ = IRR, Filt\_Cat = WS \\ 5 + 3Natk + sizeof(Coeff\_Typ)\left(Natk + \sum_{s=0}^{Natk-1} LCatk^s\right) & WT\_Typ = REV, Filt\_Cat = ARB \\ 5 + 2Natk + sizeof(Coeff\_Typ)\left(Natk + \sum_{s=0}^{Natk-1} LCatk^s\right) & WT\_Typ = REV, Filt\_Cat = WS \end{cases} \quad \text{(A-4)}$$

where *sizeof(Coeff_Typ)* is the size (in bytes) of the Satk parameter that takes values of the type *Coeff_Typ*.

Satk: Index of the ATK marker segment; the type, *Coeff_Typ*, of the scaling factor and lifting step parameters; the wavelet filter category, *Filt_Cat*; wavelet transformation type, *WT_Typ*, the initial odd or even subsequence, m_{init}.

Katk: The scaling factor, *K*. Present for irreversible transformation only, *WT_Typ = IRR*.

Natk: Number of lifting steps, N_{LS}.

Oatk^s: Offset for lifting step *s*, off_s. The index, *s*, ranges from *s* = 0 to Natk – 1. Present only if *Filt_Cat = ARB*.

Eatk^s: The base two scaling exponent for lifting step *s*, ε_s. Present only with reversible transformation, *WT_Typ = REV*. The index, *s*, ranges from *s* = 0 to Natk – 1.

Batk^{s}: Additive residue for lifting step, *s*. Present for reversible transformations (*WT_Typ* = *REV*) only. The index, *s*, ranges from *s* = 0 to Natk – 1.

LCatk^{s}: Number of lifting coefficients signalled for lifting step *s*. Provides the range, *k*, for Aatk^{sk}. The index, *s*, ranges from *s* = 0 to Natk – 1.

Aatk^{sk}: The *k*th lifting coefficient for the lifting step *s*, $\alpha_{s,k}$. The index, *s*, ranges from *s* = 0 to Natk – 1. The index, *k*, ranges from *k* = 0 to LCatk – 1.

Table A.27 – Arbitrary transformation parameter values

Parameter	Size (bits)	Values
ATK	16	0xFF79
Latk	16	9 to 65535
Satk	16	Table A.28
Katk	0 8 16 32 64 128	*WT_Typ* = *REV* *WT_Typ* = *IRR*, *Coeff_Typ* = 0 *WT_Typ* = *IRR*, *Coeff_Typ* = 1 *WT_Typ* = *IRR*, *Coeff_Typ* = 2 *WT_Typ* = *IRR*, *Coeff_Typ* = 3 *WT_Typ* = *IRR*, *Coeff_Typ* = 4
Natk	8	0 to 255
Oatk^{s}	0 8	Filt_Cat = WS –128 to 127: *Filt_Cat* = *ARB*
Eatk^{s}	0 8	*WT_Typ* = *IRR* 0 to 255: *WT_Typ* = *REV*
Batk^{s}	0 8 16 32 64 128	*WT_Typ* = *IRR* *Coeff_Typ* = 0 *Coeff_Typ* = 1 *Coeff_Typ* = 2 *Coeff_Typ* = 3 *Coeff_Typ* = 4
LCatk^{s}	8	0 to 255
Aatk^{sk}	8 16 32 64 128	*Coeff_Typ* = 0 *Coeff_Typ* = 1 *Coeff_Typ* = 2 *Coeff_Typ* = 3 *Coeff_Typ* = 4

Table A.28 – Arbitrary transformation values for the Satk parameter

Values (bits) MSB LSB	Index
xxxx xxxx 0000 000x	Not available NOTE – The indices "0000 0000" and "0000 0001" are not available having been assigned to the 9-7 irreversible wavelet filter and the 5-3 reversible wavelet filter respectively in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A.
rxxx xxxx 0000 0010 to rxxx xxxx 1111 1111	Index of this marker segment (2 to 255)
rxxx x000 xxxx xxxx rxxx x001 xxxx xxxx rxxx x010 xxxx xxxx rxxx x011 xxxx xxxx rxxx x100 xxxx xxxx	Parameters 8-bit signed integer, *Coeff_Typ* = 0 Parameters 16-bit signed integer, *Coeff_Typ* = 1 Parameters 32-bit binary floating point (ISO/IEC/IEEE 60559), *Coeff_Typ* = 2 Parameters 64-bit binary floating point (ISO/IEC/IEEE 60559), *Coeff_Typ* = 3 Parameters 128-bit binary floating point (ISO/IEC/IEEE 60559), *Coeff_Typ* = 4
rxxx 0xxx xxxx xxxx r1xx 1xxx xxxx xxxx	Arbitrary filters, *Filt_Cat* = *ARB* WS filters, *Filt_Cat* = *WS*
rxx0 xxxx xxxx xxxx rxx1 xxxx xxxx xxxx	Irreversible filter, *WT_Typ* = *IRR* Reversible filter, *WT_Typ* = *REV*
rx0x xxxx xxxx xxxx rx1x xxxx xxxx xxxx	Modify even-indexed subsequence in first reconstruction step, m_{init} = 0 Modify odd-indexed subsequence in first reconstruction step, m_{init} = 1
r0xx 0xxx xxxx xxxx r1xx xxxx xxxx xxxx	Boundary extension method used in lifting steps is constant, *Exten* = *CON* Boundary extension method used in lifting steps is whole-sample symmetric, *Exten* = *WS*
	All other values reserved

A.3.6 Component bit depth definition (CBD)

Function: Defines the bit depth of reconstructed image components coming out of any multiple component transformation process.

Usage: Present only if the multiple component transformation capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main header. The CBD marker segment is required if the multiple component transformation processes are used. At most there can be one CBD in the main header.

The presence of a CBD marker segment in a codestream alters the procedures used to determine the precision of output image components and the interpretation of the SIZ marker. See Annex J for further details.

Length: Variable depending on the number of reconstructed image component bit depths signalled. The syntax is depicted in Figure A.6.

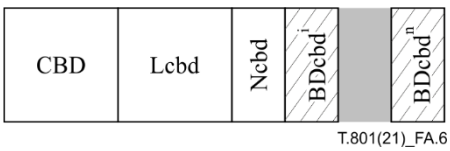

Figure A.6 – Component bit depth definition syntax

CBD: Marker code. Table A.29 shows the size and parameter values for component bit depth definition syntax.

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

$$Lmct = \begin{cases} 5 & \text{Ncbd} = 1\text{xxxxxxxxxxxxxxx} \\ 4 + \text{Ncbd} & \text{Ncbd} = 1\text{xxxxxxxxxxxxxxx} \end{cases} \quad \text{(A-5)}$$

Ncbd: Number of component bit depths included in marker segment. Table A.30 shows the value for the Ncbd parameter.

BDcbd^i: Bit depth and sign of the reconstructed image components in the order in which they are created as determined by the MCC and MCO marker segments. Either one value is signalled for all components (see Table A.30) or an individual bit depth is given for each component.

Table A.29 –Component bit depth definition parameter values

Parameter	Size (bits)	Values
CBD	16	0xFF78
Lcbd	16	5 to 16388
Ncbd	16	Table A.30
$BDcbd^i$	8	Table A.31

Table A.30 – Component bit depth definition values for the Ncbd parameter

Values (bits) MSB LSB	Number of reconstructed image component bit depths in marker
x000 0000 0000 0001 to x100 0000 0000 0000	Number of reconstructed image components (1 to 16384)
0xxx xxxx xxxx xxxx 1xxx xxxx xxxx xxxx	Bit depths included, one per reconstructed image component One bit depth included, applies to *all* reconstructed image components
	All other values reserved

Table A.31 – Component bit depth definition values for the $BDcbd^i$ parameter

Values (bits) MSB LSB	Reconstructed image component bit depths
x000 0000 to x010 0101	Component sample bit depth = value + 1. From 1 bit deep through 38 bits deep respectively.
0xxx xxxx	Component sample values are unsigned values
1xxx xxxx	Component sample values are signed values
	All other values reserved

A.3.7 Multiple component transformation definition (MCT)

Function: Defines one multiple component transformation array per marker segment. The type and index of the array defined in this marker distinguishes it from other MCT marker segments in a given header. This array can be assigned to a collection of components within the MCC marker segment.

Usage: Present only if the multiple component transformation capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main and first tile-part header of a given tile. An MCT marker segment in a tile-part header overrides a main header MCT segment for that tile if and only if the ten low-order bits of the Imct fields of both marker segments are identical.

A series of MCT marker segments (defined as having the same Imct value in the same header and a Ymct > 0) shall all appear in the same header in order of (consecutive) Zmct parameter values.

To apply the transformation array included in an MCT marker segment, an MCC marker segment shall exist that associates the MCT marker segment with a component collection. This association is made through the array definition index of the MCT marker segment and the $Tmcc^i$ fields of MCC marker segments. If no such MCC marker segment exists, then the transformation array included in the MCT marker segment shall not be used in the decoding process.

Length: Variable depending on the size of the array. The syntax is depicted in Figure A.7.

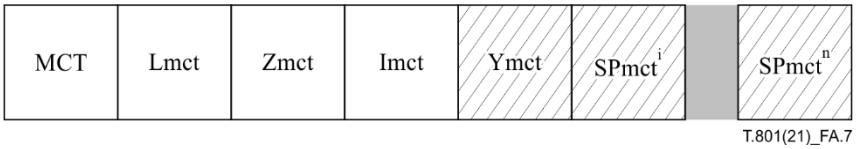

Figure A.7 – Multiple component transformation definition syntax

MCT: Marker code. Table A.32 shows the size and parameter values for multiple component transformation definition marker segment.

Lmct: Length of marker segment in bytes (not including the marker).

Zmct: Index of this marker segment in a series of MCT marker segments. All the marker segments in the series have the same Imct parameter value present in this header. The data in each subsequent MCT marker segment shall be appended, in order, to make on stream of $SPmct^i$ parameter values. The Ymct parameter values are present only in the first marker segment in the series (Zmct = 0).

Imct: Multiple component transformation index value, array type, and parameter size. An MCT marker segment, or series, with a given Imct value in the tile-part header overrides a main header MCT marker segment, or series, if and only if the ten low-order bits (index, and transformation type) of the Imct values of both markers are identical.

Ymct: Index of the last number of MCT marker segment in the series. For every series of MCT marker segments (i.e., MCT marker segments in this header with the same Imct parameter value), there shall be MCT marker segment with Zmct parameter values of 0 to Ymct. The last MCT marker segment will have Zmct = Ymct. This value is present only in the first marker segment in the series (Zmct = 0).

$SPmct^i$: Parameters for the multiple component transformation definition. One parameter value for each element in the array. See J.2 to determine the number of array elements and their order in the marker segment. The number of elements in a row and the number of rows (elements in a column) are determined by the type of array and the number of the input and output components to which it is assigned.

Table A.32 – Multiple component transformation definition parameter values

Parameter	Size (bits)	Values
MCT	16	0xFF74
Lmct	16	6 to 65535
Zmct	16	0 to 65535
Imct	16	Table A.33
Ymct	0 16	If Zmct > 0 0 to 65535
$SPmct^i$	variable	Variable, Array of types as indicated in Table A.33

Table A.33 – Multiple component transformation definition values for the Imct parameter

Values (bits) MSB LSB	Index of the array definition, type and parameter type
rrrr xxxx 0000 0001 to rrrr xxxx 1111 1111	Index of the array definition, 1 to 255
rrrr xx00 xxxx xxxx	Dependency transformation array type
rrrr xx01 xxxx xxxx	Decorrelation transformation array type
rrrr xx10 xxxx xxxx	Offset array type
rrrr 00xx xxxx xxxx	Array elements are 16 bit signed integers
rrrr 01xx xxxx xxxx	Array elements are 32 bit signed integers
rrrr 10xx xxxx xxxx	Array elements are 32-bit binary floating point (ISO/IEC/IEEE 60559)
rrrr 11xx xxxx xxxx	Array elements are 64-bit binary floating point (ISO/IEC/IEEE 60559)
	All other values reserved

A.3.8 Multiple component transformation collection (MCC)

Function: Describes the collection of input intermediate components, the collection of output intermediate components, and the associated wavelets or arrays for a multiple component transformation. This marker segment can appear in the main header and can be referred to or overridden by an MCC marker in a tile-part header.

Usage: Present only if the multiple component transformation capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main and first tile-part header of a given tile. There may be up to 255 MCC marker segments, or series of marker segments, in the main header. There may be up to 255 MCC marker segments, or series of marker segments, in any tile-part header. An MCC marker segment in a tile-part header with the same index (Imcc) as one in the main header overrides the main header MCC segment for that tile.

A series of MCC marker segments (defined as having the same Imcc value in the same header and a Ymcc > 0) shall all appear in the same header in order of (consecutive) Zmcc parameter values.

Length: Variable depending on the number of component collections. The syntax is depicted in Figure A.8.

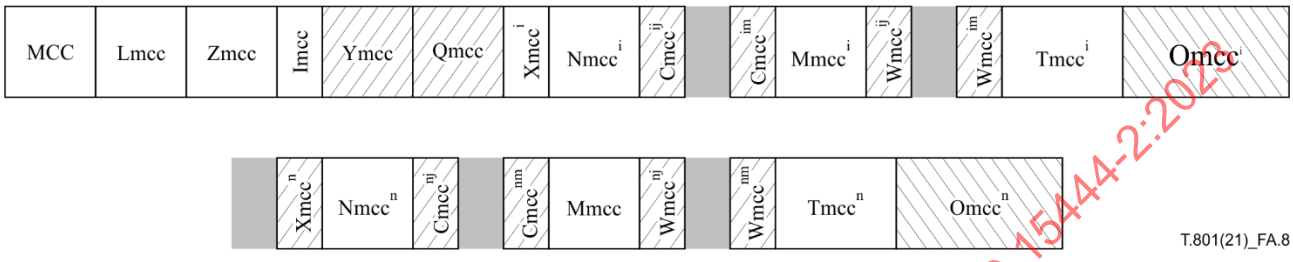

Figure A.8 – Multiple component collection syntax

MCC: Marker code. Table A.34 shows the size and parameter values for multiple component collection marker segment.

Lmcc: Length of marker segment in bytes (not including the marker).

Zmcc: Index of this marker segment in a series of MCC marker segments. All the marker segments in the series have the same Imcc parameter value present in this header. The data in each subsequent MCC marker segment shall be appended, in order, to make one stream of the other parameters. The Ymcc and Qmcc parameter appears only in the first marker segment (Zmcc = 0).

Imcc: Index of this marker segment. An MCC marker segment, or series, with a given Imcc value in the tile-part header overrides a main header MCC marker segment, or series, with the same Imcc value.

Ymcc: Index of the last number of MCC marker segment in the series. For every series of MCC marker segments (i.e., MCC marker segments in this header with the same Imcc parameter value), there shall be MCC marker segment with Zmcc parameter values of 0 to Ymcc. The last MCC marker segment will have Zmcc = Ymcc. This value is present only in the first marker segment in the series (Zmcc = 0).

Qmcc: The number of collections in the MCC marker segment. This value is present only in the first marker segment in the series (Zmcc = 0).

$Xmcc^i$: Indicates type of multiple component transformation used for the ith component collection (wavelet or array-based decorrelation or array-based dependency). Defines the interpretation applied to Tmcc.

$Nmcc^i$: Indicates the number of input components for the ith component collection and defines the number of bits (8 or 16) used to represent the component indices in ith collection.

$Cmcc^{ij}$: Input intermediate component indices included the ith component collection. The number of indices in the ith component collection is $Nmcc^i$. Each index denotes an input intermediate component. The order of the indices defines the ordering applied to the input intermediate components prior to application of the inverse transform.

$Mmcc^i$: Indicates the number of output intermediate components for the ith component collection and defines the number of bits (8 or 16) used to represent the component indices in ith collection. If anything, other than an array-based irreversible decorrelation transform is used, $Mmcc^i$ shall equal $Nmcc^i$.

$Wmcc^{ij}$: Intermediate component indices included the ith output component collection. The number of indices in the ith component collection is $Mmcc^i$. All output intermediate component indices in a given MCC marker segment shall appear only once across all collections in that MCC marker.

$Tmcc^i$: For array-based component collection transforms, $Tmcc^i$ assigns arrays defined in an MCT marker segment to the ith component collection. An MCT marker segment with the right type and index in the first tile-part header of a tile is used before an MCT marker segment with the right type and index in the main header. $Tmcc^i$ also indicates the reversibility of array-based component transforms.

For wavelet-based component collection transforms, $Tmcc^i$ assigns a wavelet kernel defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A or an ATK marker segment and the number of wavelet

decomposition levels for the ith component collection (only the dyadic decomposition of Rec. ITU-T T.800 | ISO/IEC 15444-1 is supported). An ATK marker segment with the proper index in the first tile-part header of a tile is used before an ATK marker segment with the proper index in the main header. $Tmcc^i$ also contains the index of an MCT marker segment that contains component additive offsets.

$Omcc^i$: Present in the MCC marker segment only for those component collections that use a wavelet-based transform. $Omcc^i$ indicates the reference grid offset to apply in the component dimension for the ith component collection (see J.2.2).

Table A.34 – Multiple component collection parameter values

Parameter	Size (bits)	Values
MCC	16	0xFF75
Lmcc	16	5 to 65535
Zmcc	16	0 to 65535
Imcc	8	0 to 255
Ymcc	0 16	If Zmcc > 0 0 to 65535
Qmcc	0 16	If Zmcc > 0 0 to 16383
$Xmcc^i$	8	Table A.35
$Nmcc^i$	16	Table A.36
$Cmcc^{ij}$	8 16	0 to 255 0 to 16383
$Mmcc^i$	16	Table A.37
$Wmcc^{ij}$	8 16	0 to 255 0 to 16383
$Tmcc^i$	24	Table A.38 or Table A.39
$Omcc^i$	32	0 to 4294967295

Table A.35 – Multiple component collection values for the $Xmcc^i$ parameter

Values (bits) MSB LSB	Coding style	$Tmcc^i$ parameter
rrrr rr00 rrrr rr01 rrrr rr11	Component collection transform is array-based dependency transform Component collection transform is array-based decorrelation transform Component collection transform is wavelet-based transform	Table A.38 Table A.38 Table A.39
	All other values reserved	

Table A.36 – Multiple component collection values for the $Nmcc^i$ parameter

Values (bits) MSB LSB	Coding style
0xxx xxxx xxxx xxxx to 1xxx xxxx xxxx xxxx	Input component collection indices ($Cmcc^i$) are 8 bit integers Input component collection indices ($Cmcc^i$) are 16 bit integers
x000 0000 0000 0001 to x100 0000 0000 0000	Number of input components in ith component collection (1 to 16384)

Table A.37 – Multiple component collection values for the $Mmcc^i$ parameter

Values (bits) MSB LSB	Coding style
0xxx xxxx xxxx xxxx to 1xxx xxxx xxxx xxxx	Output component collection indices ($Wmcc^i$) are 8 bit integers Output component collection indices ($Wmcc^i$) are 16 bit integers
x000 0000 0000 0001 to x100 0000 0000 0000	Number of output components in ith component collection (1 to 16384)

Table A.38 – Multiple component collection values for the $Tmcc^i$ parameter (array-based)

Values (bits) MSB LSB	Coding style
rrrr rrrx xxxx xxxx 0000 0000 rrrr rrrx xxxx xxxx 0000 0001 to rrrr rrrx xxxx xxxx 1111 1111	No assigned transform array (NULL array) Decorrelation or dependency transform array index
rrrr rrrx 0000 0000 xxxx xxxx rrrr rrrx 0000 0001 xxxx xxxx to rrrr rrrx 1111 1111 xxxx xxxx	No assigned offset array (NULL array) Offset array index
rrrr rrr0 xxxx xxxx xxxx xxxx rrrr rrr1 xxxx xxxx xxxx xxxx	Decorrelation or dependency transform is irreversible Decorrelation or dependency transform is reversible
	All other values reserved

Table A.39 – Multiple component collection values for the $Tmcc^i$ parameter (wavelet-based)

Values (bits) MSB LSB	Coding style
xxxx xxxx xxxx xxxx 0000 0000 xxxx xxxx xxxx xxxx 0000 0001 xxxx xxxx xxxx xxxx 0000 0010 to xxxx xxxx xxxx xxxx 1111 1111	Rec. ITU-T T.800 | ISO/IEC 15444-1 9-7 irreversible filter Rec. ITU-T T.800 | ISO/IEC 15444-1 5-3 reversible filter Index of the ATK marker segment containing the wavelet kernel for component collection
xxxx xxxx 0000 0000 xxxx xxxx xxxx xxxx 1111 1111 xxxx xxxx	Index of MCT marker segment containing additive offsets for component collection
xx00 0000 xxxx xxxx xxxx xxxx xx10 0000 xxxx xxxx xxxx xxxx	Number of dyadic wavelet decomposition levels used in the component collection (0 to 32)
	All other values reserved

A.3.9 Multiple component transformation ordering (MCO)

Function: Describes the order in which multiple component transformations are applied during inverse multiple component transformation processing.

Usage: Present only if the multiple component transformation capability bit in the Rsiz parameter (see clause A.2.1) is a one value. At most one MCO marker segment in main and first tile-part header of a given tile. If used in the main header, this marker segment defines the default ordering of multiple component transformation stages for all tiles. If used in the first tile-part header, then the component transformation order established by the MCO marker segment overrides any default ordering defined by a main header MCO marker segment.

Length: Variable depending on the number of multiple component transformation stages used. The syntax is depicted in Figure A.9.

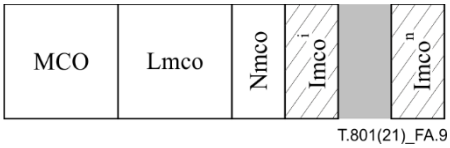

Figure A.9 – Multiple component transformation ordering syntax

MCO: Marker code. Table A.40 shows the size and parameter values for the multiple component transformation ordering marker segment.

Lmco: Length of marker segment in bytes (not including the marker). The length is given by the following expression:

$$Lmco = 3 + Nmco \quad \text{(A-6)}$$

Nmco: Number of multiple component transformation stages specified for inverse transform processing. If Nmco = 0, then no multiple component transformation processing is used for the current tile and no $Imco^i$ parameters shall appear. Otherwise, Nmco specifies the number of MCC marker segment identifiers that will follow.

$Imco^i$: Index of the MCC marker segment containing the component collection information for the i^{th} inverse multiple component transformation stage (see clause A.3.8).

Table A.40 – Multiple component intermediate collection parameter values

Parameter	Size (bits)	Values
MCO	16	0xFF77
Lmco	16	3 to 258
Nmco	8	0 to 255
$Imco^i$	8	0 to 255

A.3.10 Non-linearity point transformation (NLT)

Function: Describes either a gamma or LUT non-linearity to be applied to a single component or all components.

Usage: Present only if the non-linearity point transformation capability bit in the Rsiz parameter (see clause A.2.1) is a one value. Main and first tile-part header of a given tile. There may be no more than one marker segment per component plus one default in any header.

When used in the main header, the defined non-linearity can be established as a default for all components or established as a default for a single component. When used in a tile-part header, it can be used to establish a default for all components in the tile or to set the non-linearity transformation for a single component in that tile. Thus, the order of precedence is the following:

Tile-part NLT > Tile-part NLT default > Main NLT > Main NLT default

where the "greater than" sign, >, means that the greater overrides the lesser marker segment.

Length: Variable depending on the value of Tnlt. The syntax is depicted in Figure A.10.

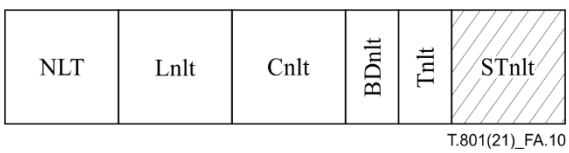

Figure A.10 – Non-linearity point transformation syntax

NLT: Marker code. Table A.41 shows the size and values of the symbol and parameters for non-linearity point transformation marker segment.

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

$$Lnlt = 6 + \begin{cases} 0 & Tnlt = 0 \text{ or } Tnlt = 3 \\ 15 & Tnlt = 1 \\ 11 + (N_{points} \cdot \Psi_{Tval}) & Tnlt = 2 \end{cases}, \quad \text{(A-7)}$$

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

NOTE – Specifying a non linearity transform with Tnlt = 0 allows a transformation to be explicitly disabled.

Cnlt: The index of the component to which this marker segment relates. The components are indexed 0, 1, 2, etc. If this value is 65 535, then this marker segment applies to all components. Table A.42 shows the value for the Cnlt parameter.

BDnlt: Bit depth and sign of the decoded image component, Z_i, after processing of the *i*th reconstructed image component by the non-linearity. If Cnlt = 65 535, then this value applies to all components. Table A.43 shows the values for the BDnlt parameter.

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

STnlt: Parameter values associated with the non-linearity as controlled by the Tnlt field.

Table A.41 – Non-linearity transformation parameter values

Parameter	Size (bits)	Values
NLT	16	0xFF76
Lnlt	16	12 to 65535
Cnlt	16	Table A.42
BDnlt	8	Table A.43
Tnlt	8	Table A.44
STnlt	variable	Table A.44

Table A.42 – Non-linearity transformation parameter values for the Cnlt parameter

Values	Components index parameter
0-16 383	Defines component to which these non-linearity transformation descriptions in this marker segment apply
65 535	Non-linearity transformation descriptions in this marker segment apply to all components
	All other values reserved

Table A.43 – Decoded image component bit depth parameter values for the BDnlt parameter

Values (bits) MSB LSB	Decoded image component bit depth
x000 0000 to x010 0101	Component sample bit depth = value + 1. From 1 bit deep through 38 bits deep respectively.
0xxx xxxx	Component sample values are unsigned values
1xxx xxxx	Component sample values are signed values
	All other values reserved

Table A.44 – Non-linearity transformation parameter values of the Tnlt parameter

Values (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	–

Table A.45 – Non-linearity transformation parameter values of the STnlt parameter (Tnlt = 1)

Parameter (in order)	Size (bits)	Values	Meaning of STnlt values
E	24	$\{0, \ldots, 255\} + \frac{\{0, \ldots, 65535\}}{65535}$	Non-linearity exponent (8-bit integer + 16-bit fraction)
S	24	$\{0, \ldots, 255\} + \frac{\{0, \ldots, 65535\}}{65535}$	Non-linearity toe slope (8-bit integer + 16-bit fraction)
T	24	$\{0, \ldots, 255\} + \frac{\{0, \ldots, 65535\}}{65535}$	Non-linearity threshold (8-bit integer + 16-bit fraction)
A	24	$\{0, \ldots, 255\} + \frac{\{0, \ldots, 65535\}}{65535}$	Non-linearity continuity parameter A (8-bit integer + 16-bit fraction)
B	24	$\{0, \ldots, 255\} + \frac{\{0, \ldots, 65535\}}{65535}$	Non-linearity continuity parameter B (8-bit integer + 16-bit fraction)
NOTE – E, S, and A parameters shall not be zero as the non linearity transformation defined by Equation K-2 is otherwise not well-defined.			

Table A.46 – Non-linearity transformation parameter values of the STnlt parameter (Tnlt = 2)

Parameters (in order)	Size (bits)	Values	Meaning of STnlt values
Npoints	16	1 to 8191	(Number of points – 1) in the LUT-style non-linearity definition (all other values reserved)
Dmin	32	0 to (2^{32}–1)	Dmin = parameter value / (2^{32}–1)
Dmax	32	1 to (2^{32}–1)	Dmax = parameter value / (2^{32}–1)
PTval	8	0000 0001 to 0010 0000	Precision of Tvalue parameter in bits (1-32). This also implies how many bytes are used to express the Tvalue (all other values reserved)
Tvalue	8, PTval ≤ 8; 16, 9 ≤ PTval ≤ 16 32, PTval > 16	variable	Run of table values for the LUT-style non-linearity. The (Npoints + 1) parameters are unsigned integers. The actual value of Tvalue is Tvalue = parameter value / (2^{PTval} – 1)

A.3.11 Quantization default, precinct (QPD)

Function: Describes the quantization default used for compressing all components of a particular resolution level and precinct. The parameter values can be overridden for an individual component, resolution level, and precinct by a QPC marker segment which, if present, shall appear in a tile-part header prior to any packets for that component, resolution level, and precinct.

Usage: Main and any tile-part header. Several QPD marker segments may appear in any tile-part header, but only one for each resolution level and precinct. If a QPD is used in a tile-part header it overrides the quantization characteristics defined by either QCD or QCC marker segments for all components of the resolution level and precinct indexed by the QPD within the scope of the particular tile. Thus, the quantization characteristics of a particular resolution level, precinct pair is determined by the presence of QCD, QCC, QPD or QPC markers in the following order of precedence:

Any tile-part QPC > Any tile-part QPD > First tile-part QCC > First tile-part QCD >
Main QPC > Main QPD > Main QCC > Main QCD

When QPD marker segments are used, they shall appear in tile-part headers before any packets are found for the indexed resolution level and precinct.

Length: Variable depending on the number of quantized sub-bands within the resolution level indexed. The syntax is depicted in Figure A.11.

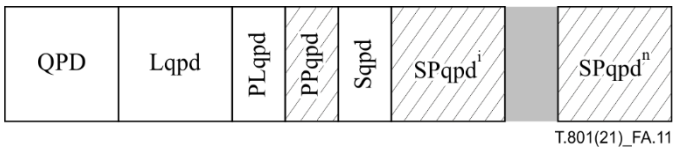

Figure A.11 – Quantization default, precinct syntax

QPD: Marker code. Table A.47 shows the size and values of the symbol and parameters for quantization default, precinct marker segment.

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

$$Lqpd = \begin{cases} 5 + \text{num\_subbands\_lev} & \text{no\_quant AND PLqpd} < 128 \\ 7 & \text{quant\_derived AND PLqpd} < 128 \\ 5 + 2 \cdot \text{num\_subbands\_lev} & \text{quant\_expounded AND PLqpd} < 128 \\ 6 + \text{num\_subbands\_lev} & \text{no\_quant AND PLqpd} \geq 128 \\ 8 & \text{quant\_derived AND PLqpd} \geq 128 \\ 6 + 2 \cdot \text{num\_subbands\_lev} & \text{quant\_expounded AND PLqpd} \geq 128 \end{cases} \quad \text{(A-8)}$$

where *num_sub-bands_lev* can be derived from clause F.2.4 for each resolution level and whether this marker segment has *no_quant*, *quant_derived*, and *quant_expounded* is signalled in the Sqpd parameter.

NOTE – The Lqcd can be used to determine how many quantization step sizes are present in the marker segment. However, there is not necessarily a correspondence with the number of sub-bands present because the sub-bands can be truncated with no requirement to correct this marker segment.

PLqpd: The resolution level index for the quantization values signalled. Equation A-9 shows how this marker segment is constructed based on the resolution level index, *lev*, as well as the precinct index, *prec*.

$$PLqpd = \begin{cases} lev & prec < 256 \\ 128 + lev & prec \geq 256 \end{cases} \quad \text{(A-9)}$$

The resolution level index, *lev*, can range from 0 to N_L, where N_L is the number of decomposition levels defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause A.6.1.

PPqpd: The precinct index for the quantization values signalled. The size of this marker segment parameter will be one byte when the PLqpd parameter is less than 128, but two bytes when PLqpd is greater than or equal to 128. This parameter will then just hold the precinct index, *prec*. The precinct index, *prec*, can range from 0 to *numprecincts* –1, where *numprecincts* is the number of precincts at resolution level *lev* and is also defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause B.6.

Sqpd: Quantization style for all components at the resolution level, *lev*, and precinct, *prec*.

SPqpd: Quantization step size value for the ith sub-band at the resolution level, *lev*, in the order defined for *lev* in F.2.4. The number of parameters is at least as large as the number of sub-bands in the tile-component with the greatest number of sub-bands at resolution level, *lev*.

Table A.47 – Quantization default, precinct parameter values

Parameter	Size (bits)	Values
QPD	16	0xFF5A
Lqpd	16	6 to 101
PLqpd	16	0 to 32 or 128 to 160
PPqpd	8 16	0 to 255, PLqpd < 128 0 to 65535, PLqpd ≥ 128
Sqpd	8	Table A.12
$SPqpd^i$	variable	Table A.12

A.3.12 Quantization precinct component (QPC)

Function: Describes the quantization used for compressing a particular component, resolution level, and precinct.

Usage: Main and any tile-part header. Several QPC marker segments may appear in any tile-part header, but only one for each component, resolution level, and precinct. If a QPC is used in a tile-part header it overrides the quantization characteristics defined by QCD, QCC, or QPD marker segments for the triplet indexed by the QPC within the scope of the particular tile. Thus, the quantization characteristics of a particular component, resolution level, and precinct is determined by the presence of QCD, QCC, QPD or QPC markers in the following order of precedence:

Any tile-part QPC > Any tile-part QPD > First tile-part QCC > First tile-part QCD >
Main QPC > Main QPD > Main QCC > Main QCD

When QPC marker segments are used, they shall appear in tile-part headers before any packets are found for the indexed component, resolution level, and precinct.

Length: Variable depending on the number of quantized sub-bands within the resolution level indexed. The syntax is depicted in Figure A.12.

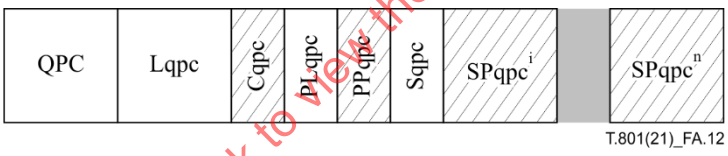

Figure A.12 – Quantization precinct component syntax

QPC: Marker code. Table A.48 shows the size and values of the symbol and parameters for quantization component marker segment.

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

$$Lqpc = \begin{cases} 6 + num\_subbdands\_lev & no\_quant\ AND\ Csiz < 257\ AND\ Plqpc < 128 \\ 8 & quant\_derived\ AND\ Csiz < 257\ AND\ Plqpc < 128 \\ 6 + 2 \cdot num\_subbdands\_lev & quant\_expounded\ AND\ Csiz < 257\ AND\ Plqpc < 128 \\ 7 + num\_subbdands\_lev & no\_quant\ AND\ Csiz \geq 257\ AND\ Plqpc < 128 \\ 9 & quant\_derived\ AND\ Csiz \geq 257\ AND\ Plqpc < 128 \\ 7 + 2 \cdot num\_subbdands\_lev & quant\_expounded\ AND\ Csiz \geq 257\ AND\ Plqpc < 128 \\ 7 + num\_subbdands\_lev & no\_quant\ AND\ Csiz < 257\ AND\ Plqpc \geq 128 \\ 9 & quant\_derived\ AND\ Csiz < 257\ AND\ Plqpc \geq 128 \\ 7 + 2 \cdot num\_subbdands\_lev & quant\_expounded\ AND\ Csiz < 257\ AND\ Plqpc \geq 128 \\ 8 + num\_subbdands\_lev & no\_quant\ AND\ Csiz \geq 257\ AND\ Plqpc \geq 128 \\ 10 & quant\_derived\ AND\ Csiz \geq 257\ AND\ Plqpc \geq 128 \\ 8 + 2 \cdot num\_subbdands\_lev & quant\_expounded\ AND\ Csiz \geq 257\ AND\ Plqpc \geq 128 \end{cases} \quad \text{(A-10)}$$

where *num_sub-bands_lev* can be derived from clause F.2.4 for each resolution level and whether this marker segment has *no_quant*, *quant_derived*, and *quant_expounded* is signalled in the Sqpc parameter.

NOTE – The Lqpc can be used to determine how many step sizes are present in the marker segment. However, there is not necessarily a correspondence with the number of sub-bands present because the sub-bands can be truncated with no requirement to correct this marker segment.

Cqpc: The index of the component to which this marker segment relates. The components are indexed 0, 1, 2, etc. (Either 8 or 16 bits depending on Csiz value.)

PLqpc: The resolution level index for the quantization values signalled. Equation A-11 shows how this marker segment is constructed based on the resolution level index, *lev*, as well as the precinct index, *prec*.

$$PLqpd = \begin{cases} lev & prec < 256 \\ 128 + lev & prec \geq 256 \end{cases} \quad \text{(A-11)}$$

The resolution level index, *lev*, can range from 0 to N_L, where N_L is the number of decomposition levels defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause A.6.1.

PPqpc: The precinct index for the quantization values signalled. The size of this marker segment parameter will be one byte when the PLqpc parameter is less than 128, but two bytes when PLqpc is greater than or equal to 128. This parameter will then just hold the precinct index, *prec*. The precinct index, *prec*, can range from 0 to *numprecincts* –1, where *numprecincts* is the number of precincts at resolution level *lev* and is also defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause B.6.

Sqpc: Quantization style for this component, resolution level, *lev*, and precinct, *prec*.

SPqpc^i: Quantization value for the ith sub-band at resolution level, *lev*, in the defined order for *lev* in clause F.2.4. The number of parameters is at least as large as the number of sub-bands in the relevant tile-component with the greatest number of sub-bands at resolution level, *lev*.

Table A.48 – Quantization precinct component parameter values

Parameter	Size (bits)	Values
QPC	16	0xFF5B
Lqpc	16	5 to 199
Cqpc	8 16	0 to 255; if Csiz < 257 0 to 16383; Csiz ≥ 257
PLqpc	16	0 to 32 or 128 to 160
PPqpc	8 16	0 to 255, PLqpd < 128 0 to 65535, PLqpd ≥ 128
Sqpc	8	Table A.12
SPqpc^i	variable	Table A.12

A.3.13 Extended capabilities (CAP)

Table A.49 defines values for the Ccap^2 field.

Table A.49 – Ccap^2 syntax and semantics

Values (bits) MSB LSB	Part parameter
1xxx xxxx xxxx xxxx	Block coder extensions may be present, as defined in COD or COC marker segments
x1xx xxxx xxxx xxxx	The position-resolution level-component-layer progression order may be present, as specified in COD or POC marker segments
	All other values reserved

A.3.14 Precinct length, tile-part header (RLT)

Function: Provides a means to discover the locations of precincts within the tile-part, for progression orders in which the packets of each precinct appear contiguously.

There may be zero or more RLT marker segments in a tile, serving collectively to provide the lengths of all precincts in the tile. RLT marker segments may appear in multiple tile-part headers of the tile, so long as the precincts whose lengths they provide appear in the same or a later tile-part header.

NOTE – if any RLT marker segment appears in a tile, then the first tile-part header for the tile contains at least one RLT marker segment.

All RLT marker segments within a tile-part header shall appear contiguously.

With the possible exception of the first RLT marker segment within a tile-part, RLT marker segments provide the lengths of precincts from the tile, in sequential order, including all bytes of the precinct's packets, including any SOP marker segments and/or EPH markers that may be found with the precinct's packets.

The first RLT marker segment within a tile-part header may, optionally, provide the lengths and contents of all other RLT marker segments in the same tile-part header.

Usage: May appear in a tile-part header. If present, neither PPM nor PPH marker segments shall be used to contain packet headers for the tile, nor shall the progression order for Spcod, Spcoc and Ppoc parameters of the tile be "Layer-resolution level-component-position progression" or "Resolution level-layer-component-position progression".

Length: Variable. The syntax is depicted in Figure A.13.

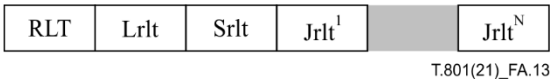

Figure A.13 – Precinct length, tile-part header syntax

RLT: Marker code. Table A.50 lists the size and values of the symbol and parameters for the precinct length tile-part header marker segment.

Lrlt: Length in bytes of the RLT marker segment (not including the marker).

Srlt: Size and type of the $Jrlt^i$ values, as defined in Table A.51. This field is encoded as a big-endian unsigned integer.

$Jrlt^i$: 16-bit or 32-bit unsigned integers that encode precinct lengths or describe other RLT marker segments. This field is encoded as a big-endian unsigned integer.

Table A.50 – Precinct length, tile-part header parameter values

Parameter	Size (bits)	Value
RLT	16	0xFF95
Lrlt	16	8 to 65534
Srlt	32	Table A.51
$Jrlt^i$	16 or 32	0 to 0xFFFFFFFF

Table A.51 – Srlt values and semantics

Srlt Value	Semantics
$[0, 2^{31} – 1]$	Total number of bytes in all subsequent RLT marker segments in the tile-part. The $(i + 1)^{th}$ subsequent RLT marker segment in the tile-part is described by $Jrlt^i$ as specified in Table A.52. Each $Jrlt^i$ is a 16-bit value.
2^{31}	16-bit unsigned $Jrlt^i$ values, providing precinct lengths
$2^{31} + 1$	32-bit unsigned $Jrlt^i$ values, providing precinct lengths

Table A.52 – Semantics of $Jrlt^i$ values when Srlt is in the range $[0, 2^{31} – 1]$

$Jrlt^i$ value	Interpretation
Even	Subsequent RLT marker segment has $\lfloor Jrlt^i/2 \rfloor$ 16-bit precinct lengths
Odd	Subsequent RLT marker segment has $\lfloor Jrlt^i/2 \rfloor$ 32-bit precinct lengths

Annex B

Variable DC offset, extension

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard, except the multiple component transformation in Annex J. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

This annex specifies variable DC offset that converts the signed values resulting from the decoding process to the proper reconstructed samples. It can be applied to with both signed and unsigned component data.

B.1 Variable DC offset flow

DC offset occurs external to any component transformations, i.e., prior to component transformations during encoding and after component transformations during decoding. Figure B.1 shows the flow of DC offset in the system with a multiple component transformation from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex G.

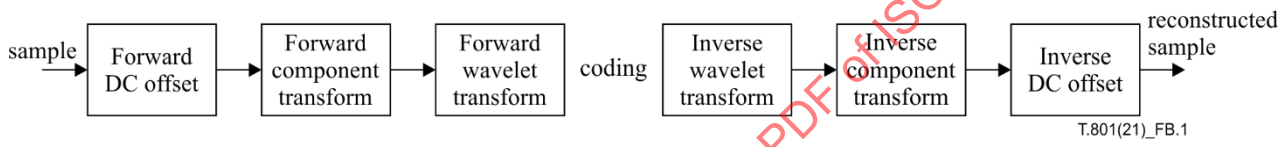

Figure B.1 – Placement of the DC offset with multiple component transformation

Figure B.2 shows the flow of DC offset in the system without a multiple component transformation from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex G.

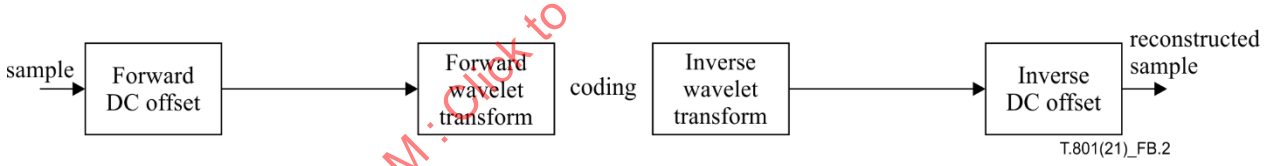

Figure B.2 – Placement of the DC offset without multiple component transformation

B.2 Inverse DC offset

If the DCO marker segment is present in the main or tile-part header (see clause A.3.1), then the DC offset, O_i, is specified by that marker. All samples of a given component are offset by adding the same quantity to each sample as follows:

$$I'(x,y) = I(x,y) + O_i \tag{B-1}$$

If no DCO marker segment is present, then the DC offset is performed as described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex G.

NOTE – If I' is not of the same precision or dynamic range as the output data it can be rounded to the closest precision and clipped in bounds.

B.3 Forward DC offset (informative)

The variable DC offset allows user control of the actual offset value. The offset, O_i, may be chosen as any value, but it is suggested that it be within the dynamic range and precision of the original data. The default value for unsigned data is:

$$O_i = 2^{Ssiz^i} \tag{B-2}$$

where $Ssiz^i$ comes from Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A. For signed data the default in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex G, is to have no offset (O_i = 0). Any other value shall be signalled in a DCO marker segment

in either the main or tile-part header. All input data is then offset by subtracting the fixed offset from all samples in the tile component. When using a variable DC offset in conjunction with a reversible transformation, the offset should be an integer value.

$$I(x,y) = I'(x,y) - O_i \tag{B-3}$$

When a non-default offset is used, care shall be taken to adjust the number of guard bits to account for any potential increase in bit depth of the offset data. If the offset, O_i, is chosen within the dynamic range of the original data, then increasing the number of guard bits, G, by one will be sufficient to handle any potential increase.

For most images the default offset setting gives good compression performance. However, for some images that have sharply peaked histograms, with a small amount of highly contrasting data, significantly improved performance can be obtained if the offset is set nearer the mode of the histogram.

Annex C

Variable scalar quantization, extension

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

Variable scalar quantization extends the default scalar quantization described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex E, to allow deadzones of variable width (up to four times the step size). Variable scalar quantization shall only be used with irreversible filters.

C.1 Variable scalar quantization

All terminology and variables described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex E, remain the same for variable scalar quantization. An additional parameter nz_b is used to convey the adjusted deadzone size. The adjusted deadzone size is $2(1–nz_b)\Delta_b$. When $nz = 0$ this is equivalent to the scalar quantizer in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex E, with a deadzone of twice the stated step size. When $nz_b > 0$ then the deadzone is smaller, and when $nz_b < 0$ then the deadzone is larger. The value of nz_b shall lie in the range [–1, 1)[1].

When $nz = 0$ for all sub-bands, there is no need to transmit adjusted deadzone factors. However, if $nz_b \neq 0$ for at least one sub-band, then extended QCD, QCC, QPD, and/or QPC marker segments shall appear in either the main header or the first tile-part header of a given tile (see clause A.2.4). The value nz_b is represented as:

$$nz_b = \frac{\text{num\_}nz_b}{2^{15}} \tag{C-1}$$

When an extended QCD, QCC, QPD, QPC marker segment appears, the adjustment factors are either signalled for each sub-band explicitly, or else signalled only for the LL sub-band. The former is known as expounded deadzone adjustment and the latter is known as derived deadzone adjustment. In the latter case, all the deadzone adjustment factors nz_b are derived implicitly from the single deadzone adjustment factor nz_0 corresponding to the LL band, according to:

$$nz_b = nz_0 \tag{C-2}$$

C.2 Variable scalar dequantization for irreversible filters

For generalized scalar dequantization with irreversible filters the reconstructed values are computed as:

$$Rq_b(u,v) = \begin{cases} (\bar{q}_b(u,v) + r2^{M_b - N(u,v)} - nz_b)\Delta_b & \bar{q}_b(u,v) > 0 \\ (\bar{q}_b(u,v) - r2^{M_b - N(u,v)} - nz_b)\Delta_b & \bar{q}_b(u,v) < 0 \\ 0 & \bar{q}_b(u,v) = 0 \end{cases} \tag{C-3}$$

where nz_b is the transmitted deadzone adjustment factor from the extended QCD, QCC, QPD, QPC marker segments and all other parameters are as described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex E. If there is no extended QCD, QCC, QPD, QPD marker segment applicable to a component in the codestream, then $nz_b = 0$ for all sub-bands. When $nz_b = 0$ this formula is identical to the scalar dequantization used with irreversible filters in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex E.

The variable scalar quantizer shall only be used with irreversible transforms.

C.3 Variable scalar quantization for irreversible filters (informative)

The quantized coefficients, $q_b(u, v)$, are computed from the unquantized coefficients, $a_b(u, v)$, by:

$$q_b(u,v) = \begin{cases} sign(a_b(u,v)) \left\lfloor \frac{|a_b(u,v)| + nz_b\Delta_b}{\Delta_b} \right\rfloor & |a_b(u,v)| \geq -nz_b\Delta_b \\ 0 & |a_b(u,v)| < -nz_b\Delta_b \end{cases} \tag{C-4}$$

[1] The "[" means the first value is included and the ")" means the last value is excluded.

where Δ_b is the quantization step size included in the QCD, QCC, QPD, QPC marker segments of Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex A, and nz_b is a deadzone adjustment parameter included in the extended QCD, QCC, QPD, QPC marker segments. If $nz_b = 0$ for all sub-bands, then it need not be signalled. If nz_b is identical for all sub-bands, then the derived signalling may be used in the extended QCD, QCC, QPD, QPC marker segments.

Annex D

Trellis coded quantization extensions

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

This annex specifies the trellis coded quantization (TCQ) option for encoding and reconstructing a sequence of wavelet coefficients. TCQ shall only be used with irreversible transformations.

D.1 Introduction to TCQ

The TCQ algorithm applies spatial-varying scalar quantization to its input sequence by choosing one of four scalar quantizers for each sample. Quantizer indices from supersets of these quantizers along with quantizer transitions in the form of a trellis provide all information necessary to reconstruct TCQ encoded wavelet coefficients.

Figure D.1 depicts the four separate scalar quantizers (D_0, D_1, D_2, and D_3) used for this Recommendation | International Standard. Included with this figure is information regarding scalar quantized indices (m_{D_i}), reconstruction levels ($R_{D_i}(m_{D_i})$), and eventual union quantizer indices ($Q_{D_i}(m_{D_i})$) for each scalar quantizer.

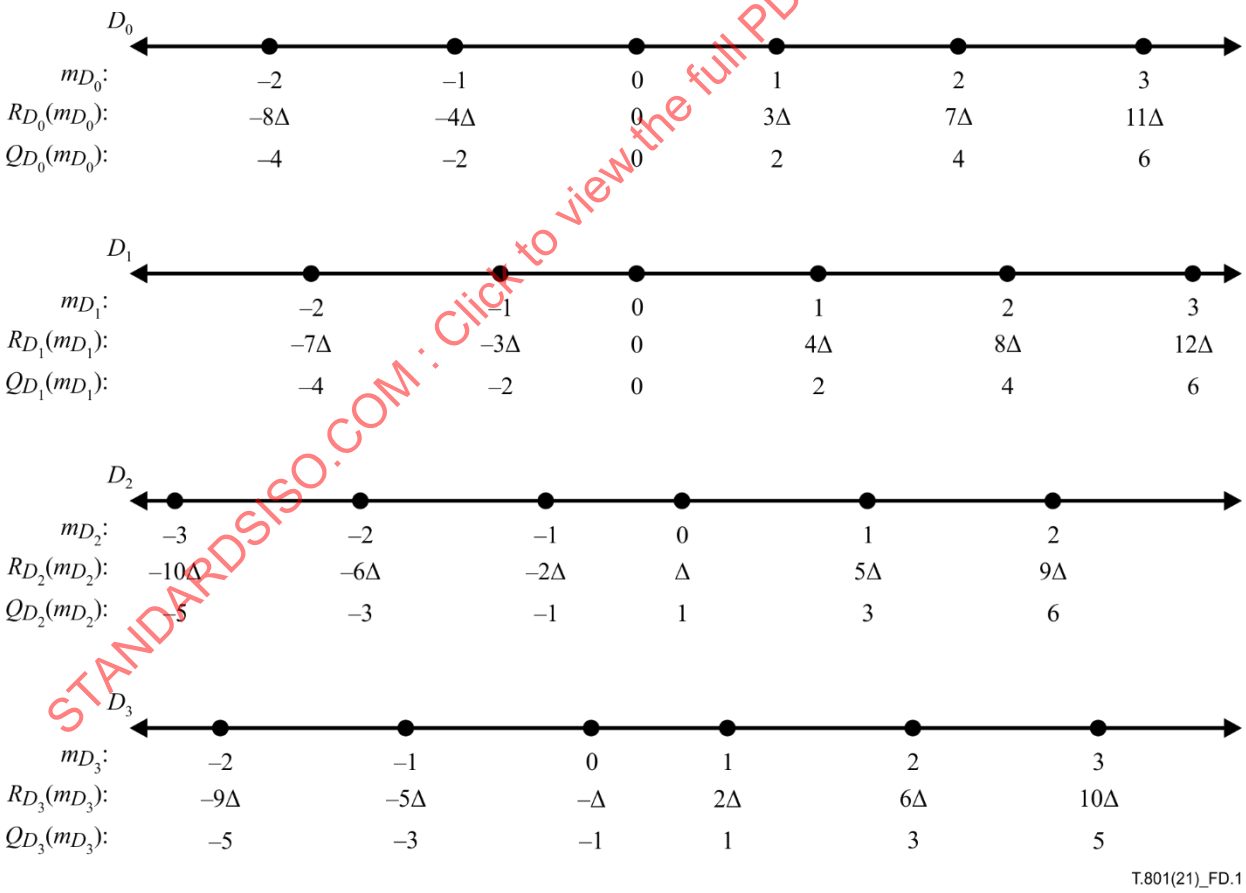

Figure D.1 – Scalar quantizers used for TCQ

Figure D.2 shows the combination of these scalar quantizers into union quantizers, A_0 and A_1, along with the indices available with each quantizer and the corresponding reconstruction levels ($R_{A_i}(m_{A_i})$).

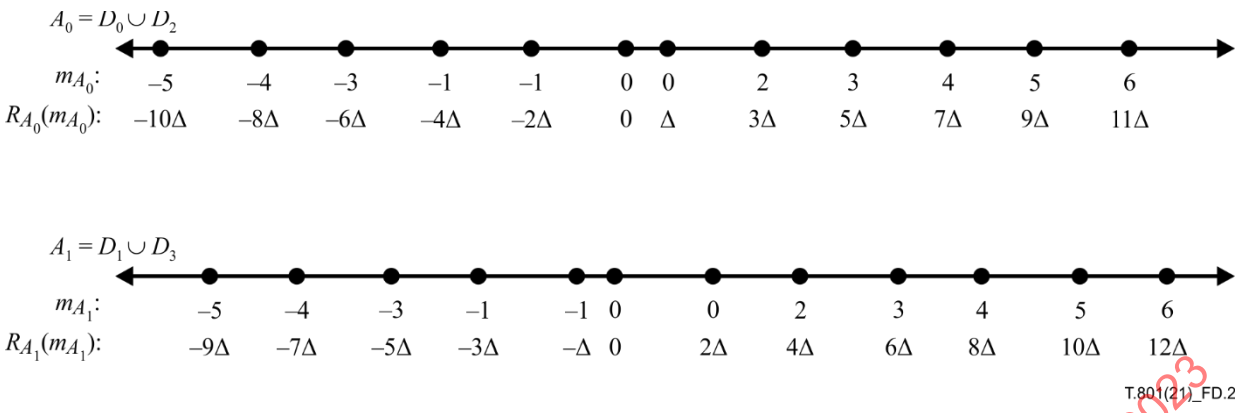

Figure D.2 – Union quantizers for TCQ

The eight state trellis directed graph with possible quantizer transitions is shown in Figure D.3, flowing left to right, where each node represents a possible trellis state. Columns of nodes represent stages which are ordered from left to right. There are exactly K+1 stages if K data points are quantized. Each node in Figure D.3 is labelled as $N_{k,s}$, where k corresponds to the stage index for the node and s is the trellis state of the node.

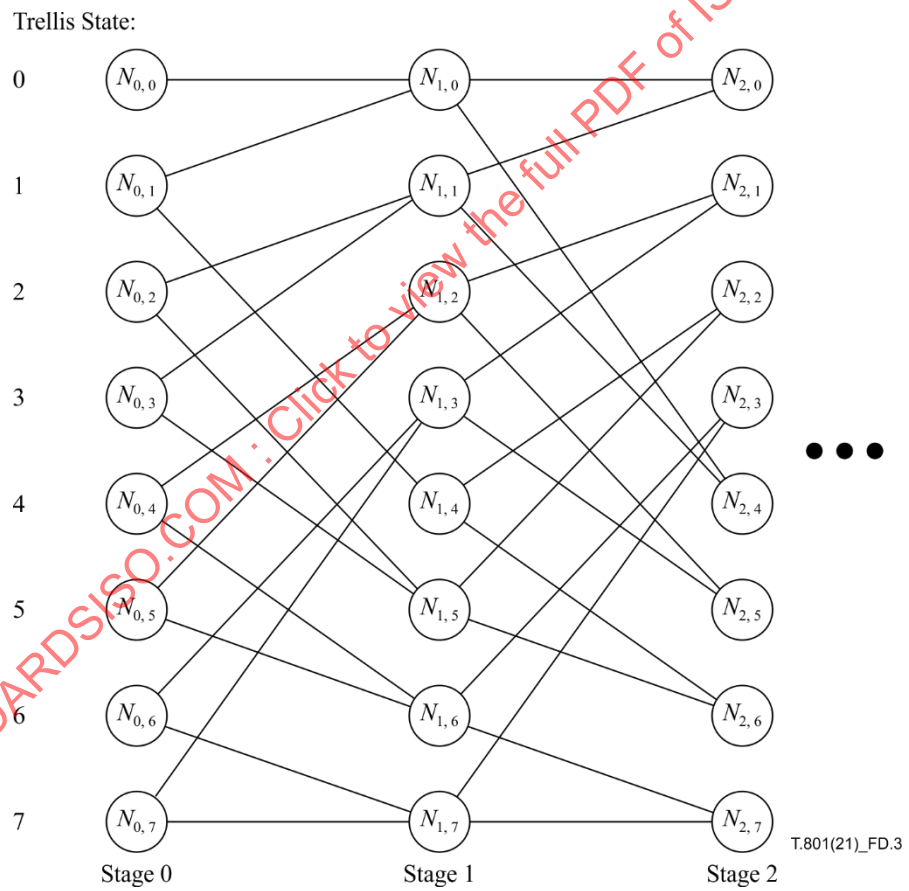

Figure D.3 – Trellis showing node indices

D.2 Sequence definition

TCQ processing is performed independently on each codeblock. The coefficients of a given codeblock are scanned following the order described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause D.1, to form a sequence of coefficients processed by TCQ.

D.3 Forward TCQ quantization (informative)

All TCQ code-block sequences within a particular sub-band b use the same quantization step-size Δ_b. As with the scalar quantization option described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause E.2, no particular selection of step sizes is required for TCQ. In fact, the ε_b and μ_b values which parameterize Δ_b can be derived according to Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-5. A recommended algorithm for choosing the set of Δ_b is that of Lagrangian Rate Allocation (LRA) defined in clause D.5.

Regardless of the algorithm for choosing the Δ_b, parameters ε_b and μ_b are determined to most closely represent the desired Δ_b by the following:

$$\Delta_b = 2^{R_b-\varepsilon_b-1}\left(1+\frac{\mu_b}{2^{11}}\right) \quad \text{(D-1)}$$

where, as described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause E.1, R_b is the dynamic range of sub-band b. The ε_b and μ_b values are then used to set the SPqcd^i, SPqcc^i, SPqpd^i, SPqpc^i parameters in the QCD, QCC, QPD, QPC marker segments (see Rec. ITU-T T.800 | ISO/IEC 15444-1, clauses A.6.4 and A.6.5).

Several look-up-tables (LUTs) are used for purposes of forward quantization. Table D.1 specifies four pre-defined LUTs: $N_0^P(N_{k,s})$ and $N_1^P(N_{k,s})$ define the parent nodes for $N_{k,s}$ in the trellis; $D_0^P(N_{k,s})$ and $D_1^P(N_{k,s})$ define the scalar quantizers that lead to $N_{k,s}$ from its parent nodes. Five other LUTs are maintained during forward quantization.

- $q_D(D_i)$ holds the best quantizer index for each scalar quantizer;
- $d_D(D_i)$ holds the resulting distortion due to each scalar quantizer;
- $d_N(N_{k,s})$ holds the cumulative distortion at node $N_{k,s}$, (survivor distortion);
- $B(N_{k,s})$ holds the parent node which yields lowest survivor distortion at node $N_{k,s}$;
- $q_N(N_{k,s})$ holds the index for the quantizer which leads from parent node $B(N_{k,s})$ to $N_{k,s}$.

The complete TCQ encoding algorithm which determines the sequence q_k of quantized indices for a particular code-block sequence is outlined in Figure D.4 and Table D.2. Additionally, optimum progressive TCQ performance is achieved when the three entropy coder passes (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause D.3) for the last bit-plane of a given code-block are concentrated into a single packet.

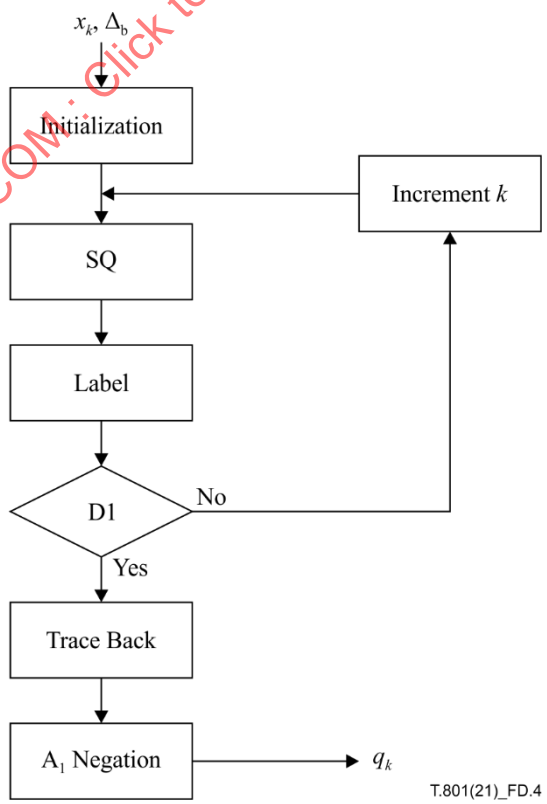

Figure D.4 – Forward TCQ processing

Table D.1 – Parent LUTs for $k > 0$ in the trellis of Figure D.3

Node	Parent nodes		Parent scalar quantizers	
	$N_0^P(N_{k,s})$	$N_1^P(N_{k,s})$	$D_0^P(N_{k,s})$	$D_1^P(N_{k,s})$
$N_{k,\,0}$	$N_{k-1,0}$	$N_{k-1,1}$	D_0	D_2
$N_{k,\,1}$	$N_{k-1,2}$	$N_{k-1,3}$	D_1	D_3
$N_{k,\,2}$	$N_{k-1,4}$	$N_{k-1,5}$	D_2	D_0
$N_{k,\,3}$	$N_{k-1,6}$	$N_{k-1,7}$	D_3	D_1
$N_{k,\,4}$	$N_{k-1,0}$	$N_{k-1,1}$	D_2	D_0
$N_{k,\,5}$	$N_{k-1,2}$	$N_{k-1,3}$	D_3	D_1
$N_{k,\,6}$	$N_{k-1,4}$	$N_{k-1,5}$	D_0	D_2
$N_{k,\,7}$	$N_{k-1,6}$	$N_{k-1,7}$	D_1	D_3

Table D.2 – Description of functional blocks in Figure D.4

Block	Description
Initialization	Set $d_N(N_{0,0}) = 0$ and $d_N(N_{0,s}) = \infty$ for $s = 1, \ldots, 7$. Set $k = 0$.
SQ	For each quantizer D_i ($i = 0, 1, 2, 3$), find the best quantizer index and compute its squared error., i.e., $$d_D(D_i) = \min_m\{(x_k - R_{D_i}(m))^2\}$$ and set $q_D(D_i)$ equal to the minimizing index value m.
Label	For each stage $s = 0, \ldots, 7$: if $(d_N(N_0^P(N_{k+1,s})) + d_D(D_0^P(N_{k+1,s})) \le d_N(N_1^P(N_{k+1,s})) + d_D(D_1^P(N_{k+1,s})))$: $$d_N(N_{k+1,s}) = d_N(N_0^P(N_{k+1,s})) + d_D(D_0^P(N_{k+1,s}))$$ $$q_N(N_{k+1,s}) = q_D(D_0^P(N_{k+1,s}))$$ $$B(N_{k+1,s}) = N_0^P(N_{k+1,s})$$ else: $$d_N(N_{k+1,s}) = d_N(N_1^P(N_{k+1,s})) + d_D(D_1^P(N_{k+1,s}))$$ $$q_N(N_{k+1,s}) = q_D(D_1^P(N_{k+1,s}))$$ $$B(N_{k+1,s}) = N_1^P(N_{k+1,s})$$
D1	At end of data sequence?
Inc k	Increment the sequence index k.
Trace Back	Set $k = K$. Find $s_{\min}$ (out of $s_{\min} = 0, \ldots, 7$) such that $d_N(N_{k,s_{min}}$ is minimum. Set $N = N_{k,s_{min}}$. While ($k > 0$): $$q_{k-1} = q_N(N)$$ $$N = B(N)$$ $$k = k - 1$$
A_1 Negation	If using A_1 union quantizer and if $q_k = 1$ then set $q_k = -1$. If using A_1 union quantizer and if $q_k = -1$ then set $q_k = 1$.

D.4 Inverse quantization (normative)

Decoded TCQ quantized indices may be reconstructed to wavelet coefficients using either the recommended full TCQ dequantization process or an approximate scalar dequantization. The full TCQ dequantization process should not be used, however, when the cleanup pass of the last bit-plane for the current code-block sequence is not fully decoded. This can occur when only part of the codestream has been transmitted or decoded.

D.4.1 Full TCQ dequantization

This approach for reconstructing wavelet coefficients uses the same quantization step size as defined in Equation D-1, where ε_b and μ_b are derived from the SPqcd^i, SPqcc^i, SPqpd^i, SPqpc^i parameters in the QCD, QCC, QPD, QPC marker segments (see Rec. ITU-T T.800 | ISO/IEC 15444-1, clauses A.6.4 and A.6.5.

The full dequantization process is outlined below in Figure D.5 and Tables D.3, D.4 and D.5. Input to this process is the code-block sequence of TCQ indices q_k (equivalent to $\bar{q}_b$ as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-1) and its output is the sequence x_k of reconstructed wavelet coefficients for the current code-block.

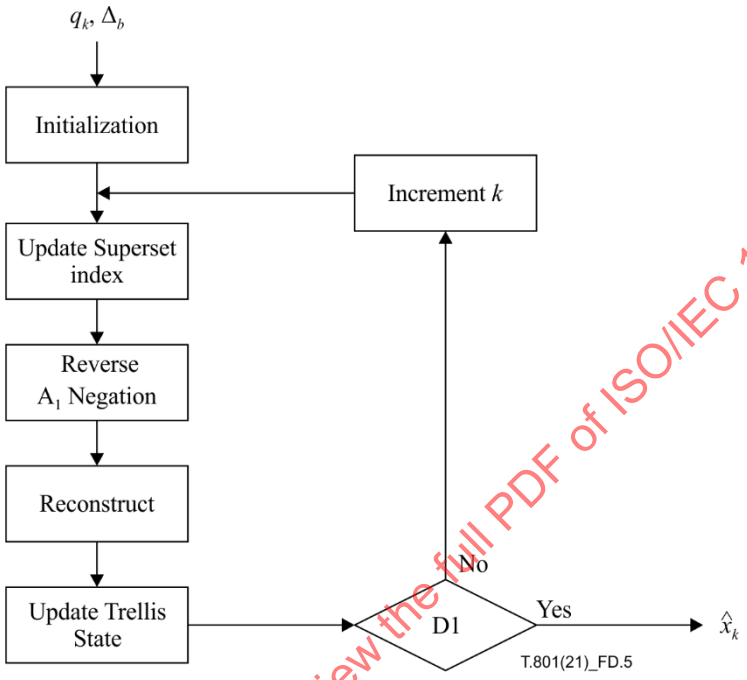

Figure D.5 – Full inverse processing for TCQ indices

Table D.3 – Description of functional blocks in Figure D.5

Block	Description
Initialization	Initialize the trellis state index s and the sequence index k: $s = 0$; $k = 1$.
Update Superset Index	Set the current union quantizer index using Table D.4: $a = A(s)$.
Reverse A_1 Negation	If a = A_1 and if q_k = 1 then set q_k = –1. If a = A_1 and if q_k = –1 then set q_k = 1.
Reconstruct	Form reconstructed coefficient x_k: $$\hat{x}_k = R_a(q_k) + \begin{cases} -r\Delta_b & R_a(q_k) > 0 \\ 0 & R_a(q_k) = 0 \\ r\Delta_b & R_a(q_k) < 0 \end{cases}$$ where $R_{A_i}(m)$ is defined in Figure D.2, and r is a reconstruction parameter. A reasonable setting is r = 01,25 when $\lvert q_k \rvert \leq 2$ and r = 0 otherwise.
Update Trellis State	Update the current trellis state using Table D.5: $s = S(s,q_k)$.
D1	At end of code-block data sequence?
Inc k	Increment the sequence index k.

Table D.4 – Look-up table for $A(s)$

Current state s	$A(s)$
0	A_0
1	A_0
2	A_1
3	A_1
4	A_0
5	A_0
6	A_1
7	A_1

Table D.5 – Look-up table for $S(s,q_k)$

Current state s	q_k Odd/Even	$S(s,q_k)$
0	Even	0
	Odd	4
1	Even	4
	Odd	0
2	Even	1
	Odd	5
3	Even	5
	Odd	1
4	Even	6
	Odd	2
5	Even	2
	Odd	6
6	Even	7
	Odd	3
7	Even	3
	Odd	7

D.4.2 Approximate dequantization

Unlike the full dequantization, this approach uses twice the step sizes defined in Equation D-1. Specifically,

$$\Delta_b = 2^{R_b-\varepsilon_b}\left(1+\frac{\mu_b}{2^{11}}\right) \quad \text{(D-2)}$$

The approximate dequantization process is outlined in both Table D.6 and Figure D.6. Input to this process is the code-block sequence of partially or fully decoded TCQ indices q_k and its output is the sequence of reconstructed wavelet coefficients for the current code-block. This dequantization technique corresponds to the basic scalar dequantization described in Rec. ITU-T T.800 | ISO/IEC 15444-1, and as such, does not require special processing to reasonably decode TCQ indices.

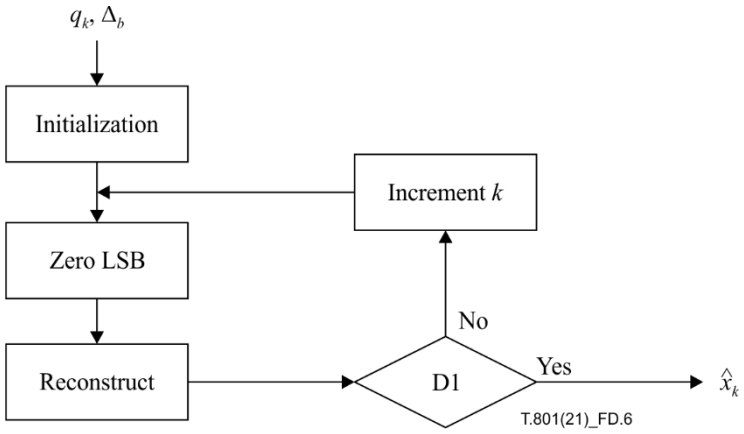

Figure D.6 – Approximate dequantization of TCQ indices

Table D.6 – Description of functional blocks for Figure D.6

Block	Description
Initialization	Initialize the sequence index k: $k = 1$.
Zero LSB	Zero out LSB of q_k.
Reconstruct	Form reconstructed coefficient x_k: $\hat{x}_k = \begin{cases} \left(q_k + r^{2^{M_b - N_b(k)}}\right) \cdot \Delta_b, & q_k > 0 \\ \left(q_k + r^{2^{M_b - N_b(k)}}\right) \cdot \Delta_b, & q_k < 0 \\ 0, & q_k = 0 \end{cases}$ where M_b, $N_b(k)$, and r are as defined for Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-8.
D1	At end of code-block data sequence?
Inc k	Increment the sequence index k.

D.5 Lagrangian rate allocation (informative)

This algorithm uses three sets of parameters to determine all sub-band step sizes. The first set is listed in Table D.7 and provides statistics for each sub-band, including standard deviation (σ_b), size weight (β_b), energy weight (Υ_b), and generalized Gaussian density (GGD) parameter (α_b). The second set of parameters include α_b and quantizer-dependent parameters which are listed in Tables D.8, D.9, D.10 and D.11. The parameters listed in these tables are based on experimentally obtained rate-distortion data for both TCQ and scalar quantization in conjunction with the entropy coder described in Rec. ITU-T T.800 | ISO/IEC 15444-1. Finally, the last parameter set simply provides the free parameter λ which is used during constrained minimization of overall image domain mean-squared error.

Table D.7 – Sub-band statistics required for LRA

Statistic	Description
σ_b	Standard deviation for sub-band *b*.
β_b	Size weight for sub-band *b* (the proportion of the number of coefficients in sub-band *b* to the total number of image pixels).
Υ_b	Energy weight for sub-band *b* (the amount of image domain squared error introduced by a unit error in a single wavelet coefficient for sub-band *b*).
α_b	GGD parameter for sub-band *b*. Found by solving, $$\frac{\sum_k (x_k - \bar{x}_b)^4}{(\sigma_b^2)^2} = \frac{\Gamma\left(\frac{5}{\alpha_b}\right)\Gamma\left(\frac{1}{\alpha_b}\right)}{\Gamma\left(\frac{3}{\alpha_b}\right)^2}$$ where x_k is the sequence of data for sub-band *b* and $\bar{x}_b$is its mean.

Table D.8 – ρ_b parameters for TCQ

	α_b = 0.5	α_b = 0.75	α_b = 1.0	α_b = 1.5	α_b = 2.0
m_h	–1.6610	–1.6610	–1.6610	–1.6610	–1.6610
a_h	–0.2985	0.0765	0.2144	0.3023	0.3186
y_1	–2.3000	3.0000	–0.8239	–0.5229	0.2218
m_l	0.0563	0.0000	–0.1950	–0.3344	–1.4917
a_l	0.1480	0.0000	–0.1240	–0.1526	–0.3311
y_2	–2.3000	–2.2080	–0.8237	–0.5229	–0.2218
a	72.0781	2.2543	70.1885	1.2153	1.3267
ζ	–0.0938	0.0460	0.0487	0.0750	–0.0040
p	283.2414	14.7723	598.0913	32.7548	70.8032
m_c	1.6610	1.6610	1.6610	1.6610	1.6610

Table D.9 – Δ_b parameters for TCQ

	α_b = 0.5	α_b = 0.75	α_b = 1.0	α_b = 1.5	α_b = 2.0
m_h'	0.5000	0.5000	0.5000	0.5000	0.5000
a_h'	0.2250	0.2250	0.2250	0.2250	0.2250
y_1'	–4.0000	–3.3980	–3.0000	–3.0000	–2.3980
m_l'	0.0276	0.0237	0.0311	0.0213	0.0473
a_l'	0.1096	0.0828	0.0925	0.0627	0.1081
y_2'	–4.0000	–3.3980	–3.0000	–3.0000	–2.3980
a′	293.3300	32606000.0000	399.6300	81289000.0000	1806.7000
ζ'	–1.5067	–1.1329	–0.8759	–0.5922	0.5818
p′	855.3700	102500000.0000	1523.9000	321130000.0000	6809.0000
m_c'	0.2117	0.3028	0.3903	0.6518	15.3783

Table D.10 – ρ_b parameters for SQ

	$\alpha_b = 0.5$	$\alpha_b = 0.75$	$\alpha_b = 1.0$	$\alpha_b = 1.5$	$\alpha_b = 2.0$
m_h	–1.6610	–1.6610	–1.6610	–1.6610	–1.6610
a_h	–0.1026	0.2744	0.4249	0.5095	0.5285
y_1	–0.7847	–1.3863	–0.7791	–0.6482	–0.3968
m_l	0.4060	–0.0942	–0.4315	–0.7034	–1.5322
a_l	0.3186	–0.1306	–0.3362	–0.4559	–0.6079
y_2	–0.4191	–0.4115	–0.3435	–0.1282	–0.0599
a	0.5912	0.4934	0.3859	0.1501	0.0350
ζ	0.1721	0.0819	0.0424	0.0227	–0.0249
p	3.2225	4.0915	3.8673	2.5889	1.4163
m_c	1.6610	1.6610	1.6610	1.6610	1.6610

Table D.11 – Δ_b parameters for SQ

	$\alpha_b = 0.5$	$\alpha_b = 0.75$	$\alpha_b = 1.0$	$\alpha_b = 1.5$	$\alpha_b = 2.0$
m_h'	0.5000	0.5000	0.5000	0.5000	0.5000
a_h'	0.4687	0.4687	0.4687	0.4687	0.4687
y_1'	–2.4000	–1.9376	–1.4771	–1.4569	–1.5025
m_l'	0.0498	0.0837	0.0643	0.0439	0.0364
a_l'	0.1196	0.1622	0.0949	0.0640	0.0547
y_2'	–2.4000	–1.9376	–1.4771	–1.4569	–1.5025
a′	3.4746	10.7335	4.8908	3.8051	5.3635
ζ'	–0.6358	–0.5266	–0.6001	–0.5131	–0.0280
p′	16.9615	85.2986	82.0612	32.4838	29.9839
m_c'	0.1851	0.1469	0.2837	0.4051	2.0851

The LRA process is defined in Figure D.7 and Table D.12.

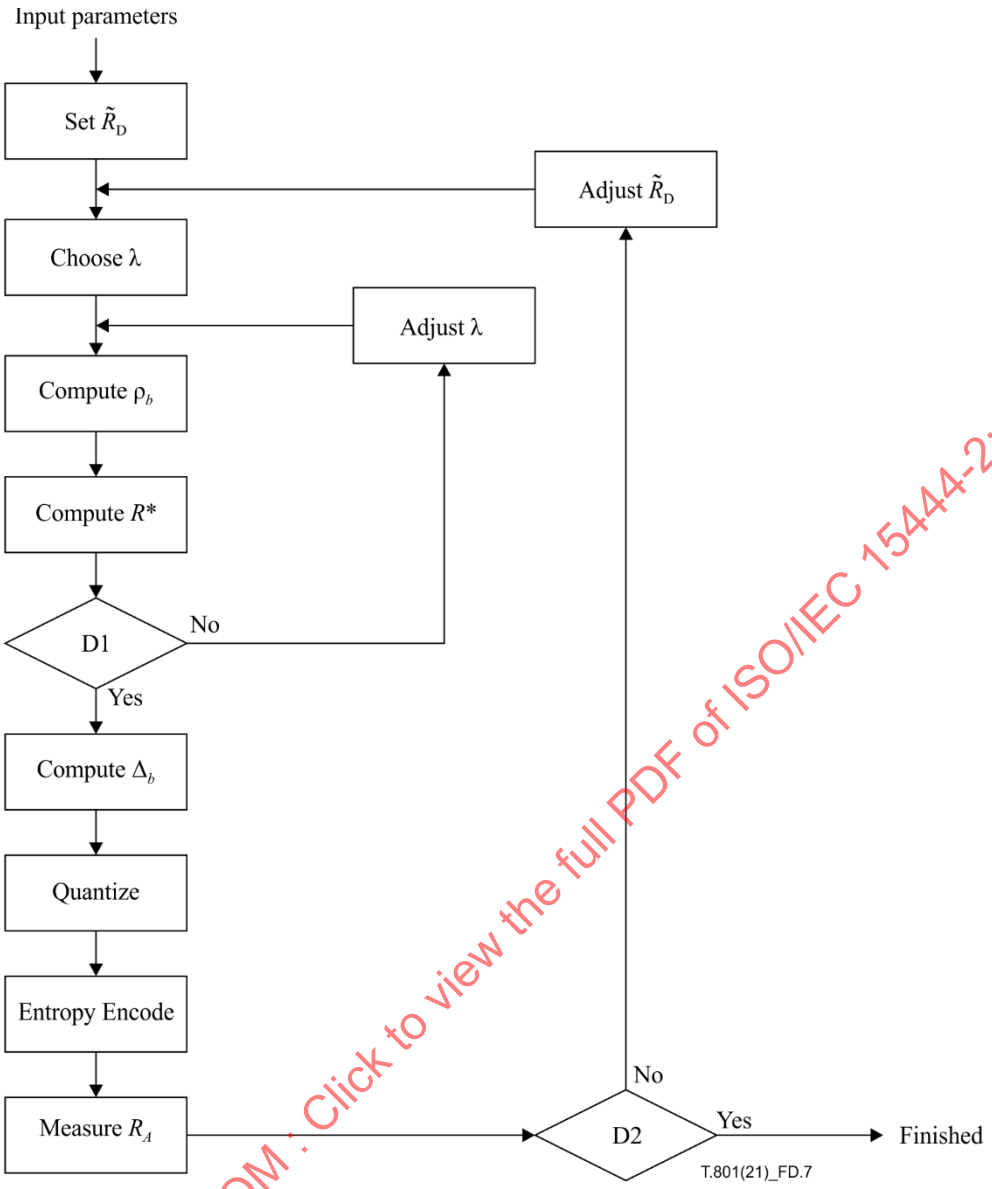

Figure D.7 – Lagrangian rate allocation

Table D.12 – Description of functional blocks in Figure D.7

Block	Description
Set $\tilde{R}_D$	Set $\tilde{R}_D$ equal to desired bit-rate R_D.
Choose λ	Provide initial estimate for Lagrangian multiplier.
Compute ρ*b*	$\rho_b = R_h + R_c + R_l$ for each sub-band b, where $R_h = m_h v + a_h$, $R_c = \begin{cases} m_c a\left\{\left[1+\left(\frac{v+a-\xi}{a}\right)^p\right]^{\frac{1}{p}}-1\right\}, & for\, v > y_2 \\ 0, & \text{otherwise} \end{cases}$ $R_l = \begin{cases} m_l v + a_l, v > y_1 \\ 0, \text{otherwise} \end{cases}$ where $v = log\left(\frac{\lambda}{\gamma_b \sigma_b^2}\right)$
Compute R^*	$R* = \sum_b \beta_b \rho_b$
D1	R^* within tolerance of $\tilde{R}_D$?
Adjust λ	Properly tune λ so $\tilde{R}_D$ and R^* converge within tolerance.
Compute Δ_b	$\Delta_b = 10^{\Delta_h+\Delta_t+\Delta_c}$ for each sub-band b where $\Delta_h = m_n' v + a_h'$ $\Delta_c = \begin{cases} m_c' a' \left\{\left[1+\left(\frac{v+a'-\xi'}{a'}\right)^{p'}\right]^{\frac{1}{p'}}-1\right\}, & for\, v > y_2' \\ 0, & \text{otherwise} \end{cases}$ $\Delta_l = \begin{cases} m_l' v + a_l', v > y_1' \\ 0, \text{otherwise} \end{cases}$ where $v = log\left(\frac{\lambda}{\gamma_b \sigma_b^2}\right)$
Quantize	Use Δ_b to quantize wavelet coefficients.
Entropy encode	Run quantized indices through variable length encoding.
Measure R_A	Measure achieved rate resulting from step sizes.
D2	R_A within tolerance of R_D?
Adjust $\tilde{R}_D$	Properly tune $\tilde{R}_D$ so R_A and R_D converge within tolerance.

Annex E

Visual masking, extensions

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

This annex describes an option that allows the coder to exploit the masking properties of the human visual system.

E.1 Introduction to visual masking (informative)

Visual masking is a mechanism where artefacts are masked by the image acting as a background signal. The main goal is to improve the image quality, especially for low resolution (often measured in dots per inch, or DPI) displays. The first effect of this technique is to improve the image quality, where the improvement becomes greater as the image becomes more complex. The key area of improvement is in low amplitude textures, such as skin. Another area of improvement is in knife-edges (i.e., those with zero transition width) in graphical images created digitally. The second main effect of this technique is that for a given fixed bit-rate, the image quality is more robust against variations in image complexity. This is accomplished at the encoder via an extended non-linearity interposed between the transformation stage and the quantization stage. At the decoder, the inverse non-linearity is applied after dequantization, prior to the inverse wavelet transformation (see Figure E.1).

E.2 Point-wise extended non-linearity (informative)

The extended masking option treats visual masking as a combination of two separate processes, i.e., self-contrast masking and neighbourhood masking. The visual masking effect is therefore exploited in two steps. The first step exploits the self-contrast masking effect by applying a point-wise power function (referred to here as the transducer function) to the original coefficient x_i obtained using an analysis filter with $gain_b$ (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Table E.1), i.e.,

$$x_i \rightarrow y_i = sign(x_i) \left| \frac{x_i}{gain_b} \right|^{\alpha} \cdot gain_b \qquad \text{(E-1)}$$

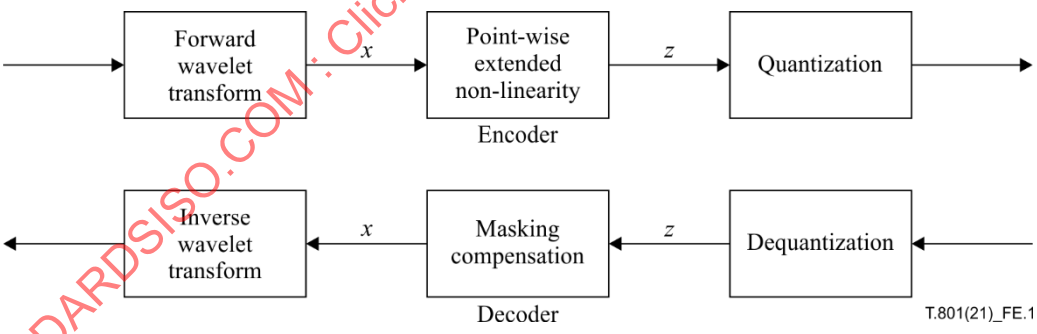

Figure E.1 – System diagram for point-wise extended masking extension

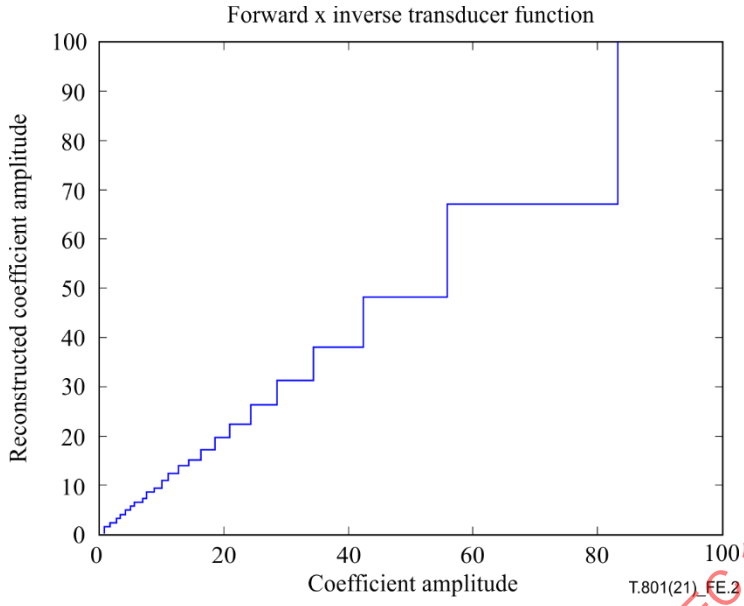

Figure E.2 – Non-uniform quantization for self-contrast masking

Equation E-1 is applied to all sub-bands above some specified resolution using the same value for α. The parameter α assumes a value between 0 and 1. A typical value of α is 0,7. Typically a point-wise extended non-linearity is not applied to the LL sub-band.

If uniform quantization is to be applied to y_i, the resultant quantization step sizes as a function of the coefficient value x_i is shown in Figure E.2. The quantization bins increase as the coefficient amplitude increases. This step assumes that each signal with which a coefficient is associated is lying on a common flat background. Under this assumption, $\{y_i\}$ are perceptually uniform. In a real image, however, this is usually not the case. Each signal is superimposed on other spatially neighbouring signals. There is some masking effect contributed from spatially neighbouring signals due to the phase uncertainty, receptive field sizes, as well as possible longer range effects. To further exploit this neighbourhood masking effect, the second step normalizes y_i by a neighbourhood masking weighting factor w_i which is a function of the amplitudes of the neighbouring signals, i.e.,

$$y_i \rightarrow z_i = \frac{y_i}{w_i} \tag{E-2}$$

where w_i is a function *g(.)* of the causal neighbouring signals as perceived by the decoder $\hat{x}_k$, denoted in a vector form as $N_i(\{\hat{x}_k\})$, i.e.,

$$w_i = g(N_i(\{\hat{x}_k\}))$$

The resultant z_i is then subject to quantization. An advantage of this strategy is its ability to distinguish between large amplitude coefficients that lie in a region of simple edge structure from those in a complex region. This feature will assure the good visual quality of simple edges on a smooth background, which is often critical to the overall perceived visual quality.

Since the neighbourhood weighting factor w_i shall be computed by the decoder as well as the encoder, the decoder perceived signals $\hat{x}_i$ shall be computed to include the effects of quantization and reconstruction. That is:

$$\frac{|\hat{x}_i|}{gain_b} = \left(\frac{Q^{-1}(Q(|z_i|))}{gain_b} g(N_i(\{\hat{x}_k\}))\right)^{1/\alpha} \tag{E-3}$$

where *Q*() denotes a quantization operation. The $\hat{x}_i$ depends on previously computed values in the causal neighbourhood, so $\hat{x}_i$ is always well defined.

This simulation of the quantization process is used by the encoder to ensure that both the encoder and the decoder perform exactly the same operation to calculate w_i. For non-scalable coding, this is a relatively simple problem. The encoder will use the final quantized neighbouring coefficients to calculate the neighbourhood masking factor.

For embedded coding the encoder cannot do the non-linear transformation based on the exact final compressed and quantized version of the coefficient x_k because the "extended non-linearity" is applied prior to scalable compression and

the decoder might receive a truncated bit stream. Nevertheless, this discrepancy in w_i can be completely eliminated or reduced by using the same course quantization value from bit truncation of the neighbouring coefficients to calculate the masking weighting factor w_i at the encoder and the decoder. This is accomplished by maintaining only the *bits_retained* most significant bits of $Q(|z_i|)$ in the above formula (the rest of the bits are replaced with 0). As long as *bits_retained* is small enough (with respect to the lowest interesting bit rate at the decoder), the decoder will be able to obtain exactly the same quantized (bit truncated) version of the neighbouring coefficients. The compromise is a coarser granularity for w_i which may slightly affect the accuracy of the masking model. But experiments have suggested that the performance usually is not very sensitive to which quantized version of the neighbouring coefficients is used.

Visual masking may be applied to all resolution levels that have an index value not less than a particular level *minlevel* which can be specified in the bit stream. For example, if *minlevel* is set to 1, then the extended non-linearity is applied to all sub-bands except the lowest LL band.

E.3 Decoding with visual masking

When the VMS marker (see clause A.3.2) is present for component *c*, then the following shall occur between inverse quantization and inverse DWT. In component *c*, for each sub-band *b* with resolution level *r* not less than *minlevel*, execute the following formula in raster order over sub-band *b*:

$$x_i = sign(z_i)\left[\left|\frac{z_i}{gain_b}\right|\left(1+\left(a\sum_{k\in neighbourhood}\left|\frac{\hat{x}_k}{gain_b}\right|^{\beta}\right)/|\varphi_i|\right)\right]^{1/\alpha}\cdot gain_b \quad \text{(E-4)}$$

where z_i is the coefficient under consideration, and $gain_b$ is the analysis filter gain (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Table E.1). The normalization constant *a* is defined as $(1000/2^{(R_I-1)})^{\beta}$ where R_I is the bit-depth of the spatially reconstructed component samples as specified in the SIZ marker segment (see Rec. ITU-T T.800 | ISO/IEC 15444-1, clause A.5.1). The neighbourhood is a causal neighbourhood of the current coefficient, z_i, having a nominal height of (*win_width*+1) and width of (2*win_width*+1) (see Figure E.3). The neighbourhood excludes the current coefficient, z_i, and shrinks to respect sub-band boundaries. The neighbourhood also shrinks to respect code block boundaries if *respect_block_boundaries* = 1. $|\varphi_i|$is the number of pixels in the neighbourhood of z_i.

The value $\hat{x}_i$ is computed from a *bits_retained* quantized version of z_i denoted $\hat{q}_i$ as shown in Equations E-5 and E-6.

$$\hat{q}_i = \lfloor |z_i|/\Delta_b \rfloor \text{ with all but the } bits\_retained \text{ MSBs replaced by zero} \quad \text{(E-5)}$$

where Δ_b is the quantization step size and $\lfloor |z_i|/\Delta b \rfloor$ is an integer containing M_b unsigned bits. See Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause E.1, for definitions of Δ_b and M_b.

NOTE 1 – For example, if z_i = –36, Δ_b = 5, and M_b = 10, then $\lfloor |z_i|/\Delta b \rfloor$ = 00 0000 0111 (in binary). If *bits_retained* < 8 then, $\hat{q}_i$ = 0. If *bits_retained* = 8, then $\hat{q}_i$ = 4.

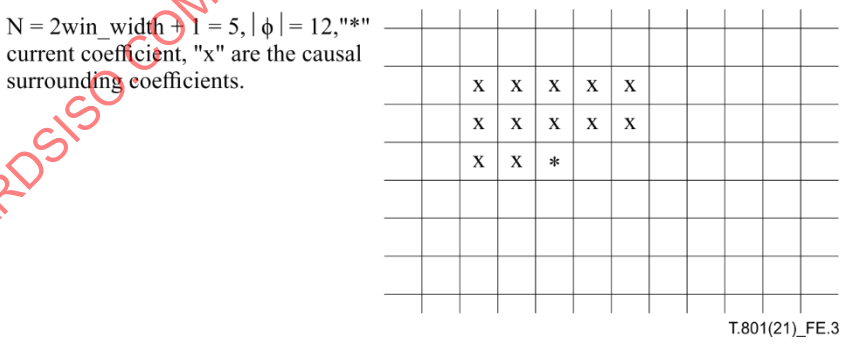

Figure E.3 – Causal neighbourhood

$$\frac{|\hat{x}_i|}{gain_b} = \left[\frac{\hat{q}_i\Delta_b}{gain_b}\left(1+\left(a\sum_{k\in neighbourhood}\left|\frac{\hat{x}_k}{gain_b}\right|^{\beta}\right)/|\varphi_i|\right)\right]^{1/\alpha} \quad \text{(E-6)}$$

The values of *minlevel*, α, β, *win_width*, *respect_block_boundaries*, and *bits_retained*, all of which can differ for each component, are given in the VMS marker segment (see clause A.3.2).

NOTE 2 – When *respect_block_boundaries* = 1, it allows parallel implementation and restricts error propagation, but may sacrifice some performance.

E.4 Encoding with visual masking (informative)

When employing visual masking on component *c*, the following should occur between the DWT and quantization. In component *c*, for each sub-band *b* with resolution *r* not less than *minlevel*, execute the following formula in raster order over sub-band *b*:

$$z_i = \frac{sign(x_i)\left|\frac{x_i}{gain_b}\right|^{\alpha} \cdot gain_b}{1+\left(a\sum_{k\in neighbourhood}\left|\frac{\hat{x}_k}{gain_b}\right|^{\beta}\right)/|\varphi_i|} \quad \text{(E-7)}$$

where x_i is the wavelet coefficient under consideration, and the parameters $gain_b$, *a*, *neighbourhood*, and $|\varphi_i|$ are defined as in clause E.3. The value $\hat{x}_i$ is defined as in Equations E-5 and E-6.

The values of *minlevel*, α, β, *win_width*, *respect_block_boundaries*, and *bits_retained*, all of which can differ for each component, are recorded in the VMS marker segment (see clause A.3.2).

E.5 Setting parameters (informative)

The parameter β assumes a value between 0 and 1, and, together with *win_width*, is used to control the degree of neighbourhood masking. The parameters β and *win_width* play important roles in differentiating coefficients around a simple edge from those in the complex area. The parameter *win_width* controls the degree of averaging; β controls the influence of the amplitude of each neighbouring coefficient. It is important that β assumes a value much smaller than 1. A good value of β is 0,2. This helps to protect coefficients around simple sharp edges, since the coefficients around sharp edges usually have high values. A small value of β suppresses the contribution of a few large coefficients around sharp edges to the masking factor, thus implicitly distinguishing coefficients around sharp edges from coefficients in a complex region.

A special case of the point-wise extended masking approach is the self-contrast masking approach achieved when β is set to 0. The self-contrast masking is referred to as the case where the mask signal is at exactly the same frequency, orientation and location as the distortion signal. This masking approach assumes that the wavelet band structure is used and filters are a good match to the visual system's underlying channels, which is usually not true. Therefore, it may have an over-masking problem at diagonal edges, especially at relatively lower bit rates.

Some parameters that are encoded into the VMS marker segment (see clause A.3.2) are α, β, *bits_retained* (the number of most significant bits to be retained for obtaining quantized neighbouring coefficients), *win_width* (half of the causal neighbourhood window width, i.e., $N = 2win\_width + 1$), and *minlevel* (the lowest frequency level at which masking will start). A good set of values for these parameters are 0,7, 0,2, 9, 6, and 1, respectively. The switch *respect_block_boundaries* is also included in the VMS marker segment.

E.6 Compatibility with other technologies (informative)

The visual masking extension works with both scalar quantization and TCQ (see Annex D) for irreversible filters. It can be combined with visual frequency weighting (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause J.12) to further improve the visual quality. Typically, the transform coefficients are multiplied by CSF weights before they are subjected to the "transducer" function. In some implementations, however, it is more convenient to interchange the operations because the CSF weighting can then be incorporated into the rate distortion optimization. To do this, the CSF weights, originally designed for the *x* domain, are modified so that they can be applied in the *z* domain are raised to the power α.

Annex F

Arbitrary decomposition of tile-components, extensions

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

The extension described in this annex specifies options available for forming wavelet sub-band decompositions. The notational conventions are first introduced followed by updates to various equations, text, decompositions and procedures from Rec. ITU-T T.800 | ISO/IEC 15444-1. Many of these new and updated procedures are defined recursively. Except for variables that are included in a recursive procedure's output parameter list, all variables for recursive procedures are maintained with internal copies that do not change outside of the procedure's scope.

F.1 Wavelet sub-bands

This Recommendation | International Standard provides four tiers of detail for specifying two-dimensional bandpass signals (called sub-bands) at various spatial resolutions. Each tier provides more information in defining finer detail in the sub-band decomposition structure. These tiers are defined below, starting with the tier with the lowest level of detail.

F.1.1 Tier 1: Number of decomposition levels

The first tier in defining sub-band decompositions is the number of wavelet decomposition levels, N_L. This value is signalled for each tile-component in the COD or COC markers as specified in Table A.15 of Rec. ITU-T T.800 | ISO/IEC 15444-1. As with that Recommendation | International Standard, decomposition level indices are 1 for highest resolution sub-bands and N_L for the lowest resolution sub-band. Resolution indices, on the other hand, are labelled with value zero for the lowest resolution and N_L for the highest resolution. A value of zero for N_L indicates no wavelet transformation for the tile-component.

F.1.2 Tier 2: Resolution formation

The various spatial resolutions are obtained through joint/disjoint horizontal and/or vertical downsampling of higher resolutions. As a result, spatial resolutions with subsampling factors from the original image that differ in the horizontal and vertical directions are allowed. As in Rec. ITU-T T.800 | ISO/IEC 15444-1, the orientation of each spatial resolution (or sub-band) is specified with a two-character code, where the first letter indicates horizontal filtering, the second letter indicates vertical filtering, and the letters L and H indicate lowpass or highpass filtering followed by decimation by a factor of two. This annex also provides for the third letter X to indicate no vertical/horizontal filtering and decimation. Since spatial resolutions are not produced with highpass processing and no two spatial resolutions can be the same, there are three possible orientations for each resolution: LL, LX, or HX. The signalling required to specify resolution formation is achieved via the COD, COC, and DFS marker segments (see clauses A.2.3 and A.3.3) as described in clause F.2.5.

F.1.3 Tier 3: Sub-level decompositions

Wavelet sub-bands resulting from the first two tiers of wavelet processing may be further decomposed into new sub-bands of reduced bandpass extent. The concept of decomposition sub-levels is used to help convey this detail tier. An absolute maximum of three decomposition sub-levels may be produced at decomposition level *lev*, with the first sub-level resulting from decomposition of the next highest resolution level. Use of ADS marker segments (see clause A.3.4) for signalling the maximum number of sub-levels, $\theta(lev)$, for each decomposition level is described in clause F.2.

F.1.4 Tier 4: Horizontal and vertical splits to variable sub-level depths

Not all sub-bands have to be decomposed to the maximum sub-level depth. As a result, sets of sub-bands with non-uniform size at the same decomposition level may be produced. The look-up-tables (LUTs) $S()$ and $J()$ defined in clause F.2 show how information in the ADS marker segments (see clause A.3.4) is used to signal the varying sub-level depth throughout complete wavelet decompositions.

Sub-bands can also be split disjointly in the horizontal and vertical directions, thus allowing sub-bands with subsampling factors from the original image that differ in the horizontal and vertical directions. As a result, sub-bands may be further decomposed into three separate sets of new sub-bands, as depicted in Figure F.1. The first set has decomposition sub-

levels with LL, HL, LH and HH orientations that result from joint horizontal and vertical splits as in Rec. ITU-T T.800 | ISO/IEC 15444-1. The second set yields only LX and HX orientations that result from just horizontal splits of a sub-band. The final set provides XL and XH orientations that result from just vertical splits. In addition to indicating sub-level depth, the LUT *S*() provides details necessary to specify joint and disjoint horizontal/vertical processing. Figure F.1 also shows how the elements of *S*() would be assigned for such processing. As a result of disjoint horizontal and vertical processing, sub-bands may be produced during wavelet processing which have different horizontal and vertical downsampling factors from the original image. The LUT *R*(*lev*) is defined in clause F.2 for specifying the level and orientation of decomposition level *lev*.

Each of these three LUTs (*S*(), *J*() and *R*()) are used throughout most of the procedures defined in this annex. However, to avoid clutter, usage of these LUTs in a procedure is not explicitly specified unless these LUTs are modified by that procedure.

F.1.5 Complete sub-band notation

A colon separated notation is used to label the index *b* of sub-band a_b for the four tiered wavelet decomposition approach defined in this annex. This notation (which is a simple extension of that given in Rec. ITU-T T.800 | ISO/IEC 15444-1) begins with an index *lev* corresponding to the decomposition level of the sub-band, followed by the two-letter orientation for the first sub-level decomposition. For sub-bands with greater than one sub-level, a colon follows along with the second sub-level decomposition orientation. A final colon along with the third sub-level decomposition orientation ends the notation for sub-bands with greater than two sub-levels.

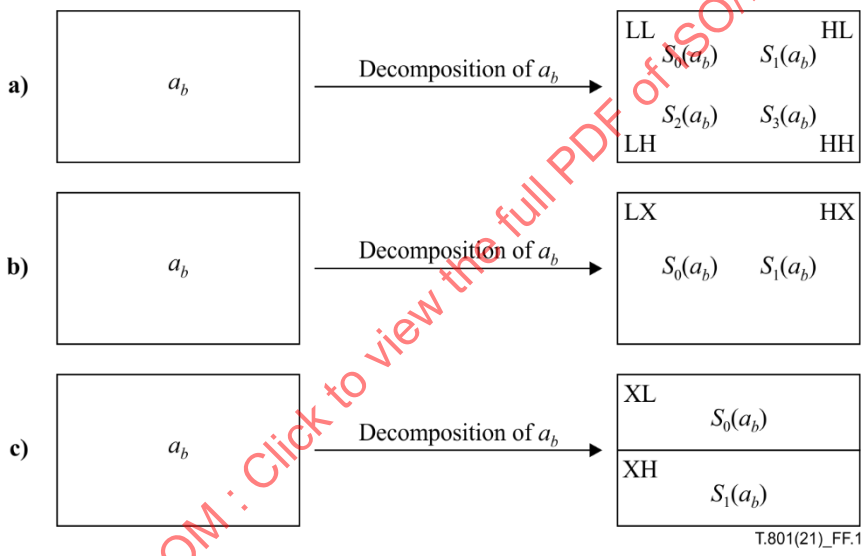

Figure F.1 – Possible splits of sub-bands

F.1.6 HorOrient, VerOrient and PrimeOrient sub-band operators

To aid definitions of procedures in this annex, the operator HorOrient(a_b) refers to the last filtering operation (H, L, or X) which has been applied in the horizontal direction to a given sub-band a_b, while the operator VerOrient(a_b) refers to the last filtering operation (H, L, or X) which has been applied in the vertical direction to a given sub-band a_b. The operator PrimeOrient(a_b) is used to refer to the orientation (LL, LX, XL, HL, LH, HX, XH or HH) of the highest sub-level sub-band which eventually spawns into sub-band a_b within the same decomposition level. For example, for sub-band $a_b = a_{2LH:HX}$, HorOrient(a_b) = H, VerOrient(a_b) = X, PrimeOrient(a_b) = LH.

F.2 Equation, text and decomposition updates

The arbitrary decomposition capability introduced in this annex impacts several clauses of Rec. ITU-T T.800 | ISO/IEC 15444-1 outside of the wavelet transform. These affected clauses are specified below along with updates to allow conformance with this annex.

F.2.1 Updates to N_LLL references

General references to sub-band N_LLL are made throughout Rec. ITU-T T.800 | ISO/IEC 15444-1. To conform to the extension specified in this annex, these references should be updated to any of the N_LLL, N_LLX or N_LXL sub-bands.

F.2.2 Context updates

The wavelet sub-level and horizontal/vertical disjoint processing introduced in this annex require updates to the context propagation shown in Table D.1 of Rec. ITU-T T.800 | ISO/IEC 15444-1. An update for this table is shown in Table F.1, with references made to Figure D.2 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

Table F.1 – Updates to contexts for significance propagation and cleanup coding passes

Sub-bands with primary orientation of LL, LH, LX, XL, or XH			Sub-bands with primary orientation of HL and HX			Sub-bands with primary orientation of HH		Context Label[a)]
$\sum H_i$	$\sum V_i$	$\sum D_i$	$\sum H_i$	$\sum V_i$	$\sum D_i$	$\sum (H_i + V_i)$	$\sum D_i$	
2	x[b)]	x	x	2	x	x	≥ 3	8
1	≥ 1	x	≥ 1	1	x	≥ 1	2	7
1	0	≥ 1	0	1	≥ 1	0	2	6
1	0	0	0	1	0	≥ 2	1	5
0	2	x	2	0	x	1	1	4
0	1	x	1	0	x	0	1	3
0	0	≥ 2	0	0	≥ 2	≥ 2	0	2
0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

a) The context labels are indexed only for identification convenience in this Specification. The actual identifiers used is a matter of implementation.

b) x = do not care.

F.2.3 Extension to Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation B-14

This equation shows how tile-components divide into resolution levels, and should be updated for this annex according to:

$$trx_0 = \left\lceil \frac{tcx_0}{2^{\text{GET\_HOR\_DEPTH}(N_L - r)}} \right\rceil \quad trx_1 = \left\lceil \frac{tcx_1}{2^{\text{GET\_HOR\_DEPTH}(N_L - r)}} \right\rceil$$
$$try_0 = \left\lceil \frac{tcy_0}{2^{\text{GET\_VER\_DEPTH}(N_L - r)}} \right\rceil \quad try_1 = \left\lceil \frac{tcy_1}{2^{\text{GET\_VER\_DEPTH}(N_L - r)}} \right\rceil \tag{F-1}$$

The usage for the procedures GET_HOR_DEPTH and GET_VER_DEPTH are defined in Figure F.2 whereas the definitions of these algorithms are depicted in Figure F.3.

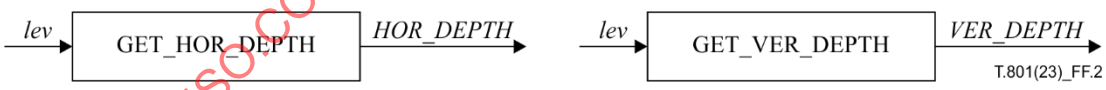

Figure F.2 – Parameters for the GET_HOR_DEPTH and GET_VER_DEPTH procedures

F.2.4 Remaining updates

From Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation B-15 (defining sub-band dimensions tbx_i and tby_i), Equation E-4 (defining sub-band gains $gain_b$) and Equation E-5 (defining the number of sub-band decomposition levels, n_b) as well as the order, $O()$, of compressed sub-band data inside packet bitstreams and the dimensions *xpb* and *ypb* for precincts in each sub-band are defined for this annex by the front end procedure SET_SUBBAND_INFO shown in Figures F.4 and F.5 along with the recursive procedure RECUR_INFO defined by both Figures F.6 and F.7. The SET_SUBBAND_INFO procedure first calls the procedures INIT_θ and INIT_S_R (defined in clause F.2.5) to set up the LUTs $R()$, $S()$ and $J()$ using the N_L, $d_R()$, and I_R information retrieved via the COD, COC, and DFS marker segments and the $d_\theta()$, I_θ, $d_S()$, and I_S information retrieved via the DFS and ADS marker segments (see clauses A.3.3, A.3.4 and F.2.5). The B-15 updates also refer to Table F.2 based on the orientations of various sub-bands. Although the order $O()$ is defined in reverse order by RECUR_INFO and the bulk of SET_SUBBAND_INFO, the last step in SET_SUBBAND_INFO reverses the order for proper output of sub-band information. Also, the front end procedure SET_SUBBAND_INFO calls the RECUR_INFO using parameters *rPPx* and *rPPy* which are the same *PPx* and *PPy* parameters signalled through COD and COC markers (see clause A.2.3) for each resolution r as described in Clause B.6 in Rec. ITU-T T.800 | ISO/IEC 15444-1. To avoid *xpb* and *ypb* values less than one, both of these *rPPx* and *rPPy* values shall also be greater than or equal to $\theta(N_L - r)$ for all resolution levels except resolution $r = 0$. Finally, as with most loops with the index j of length $J(S(a_b))$, the j index in RECUR_INFO is decremented to cause processing to recurse on the next lowest resolution level.

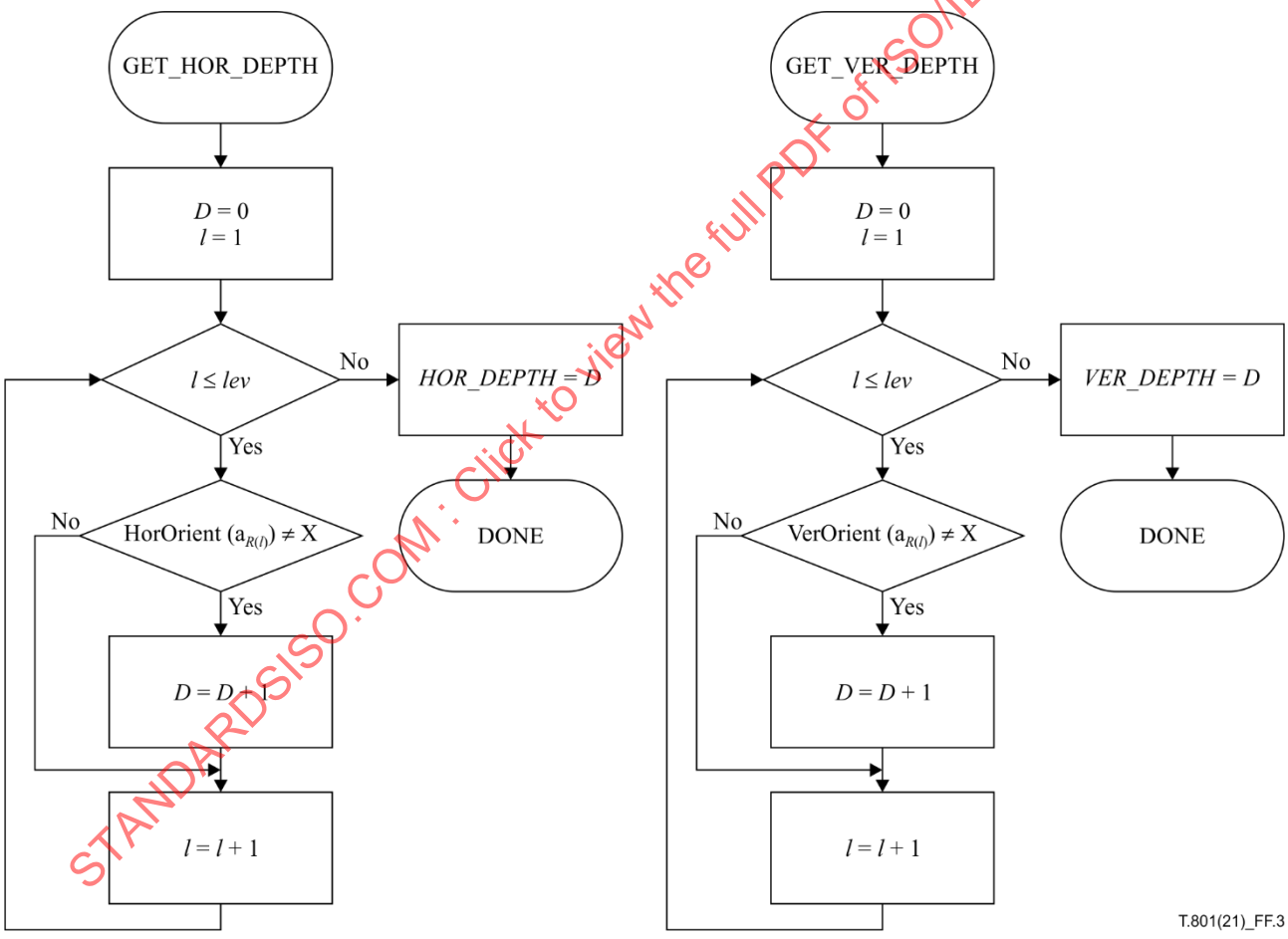

Figure F.3 – The GET_HOR_DEPTH and GET_VER_DEPTH procedures

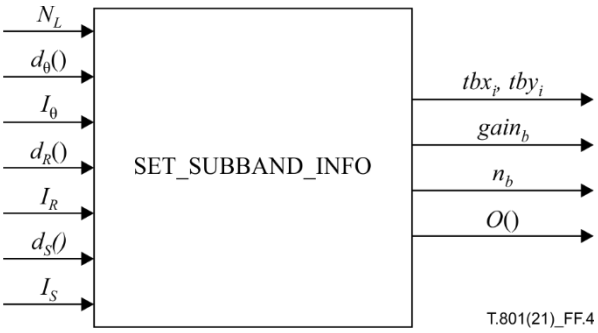

Figure F.4 – Parameters for the SET_SUBBAND_INFO procedure

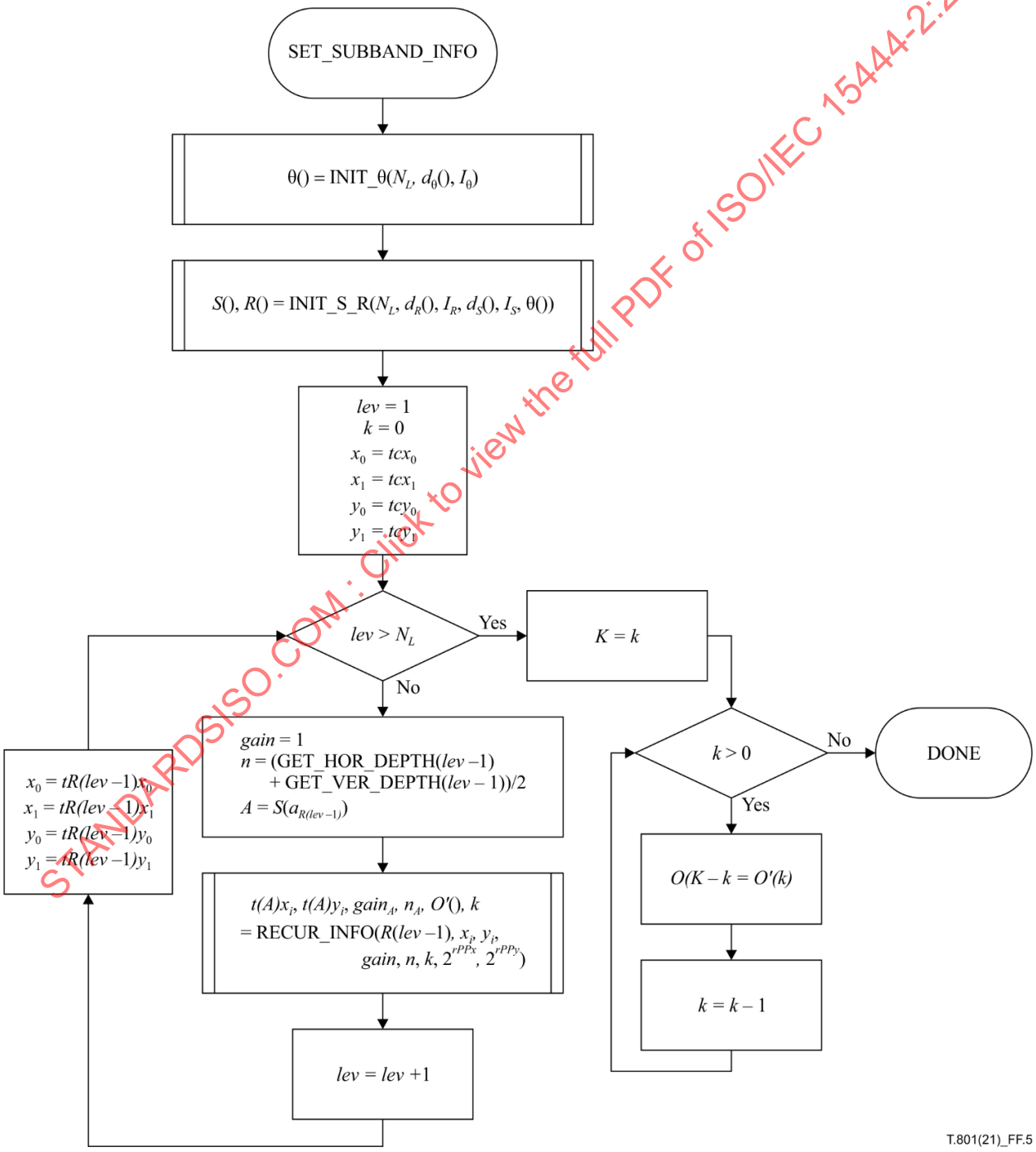

Figure F.5 – The SET_SUBBAND_INFO procedure

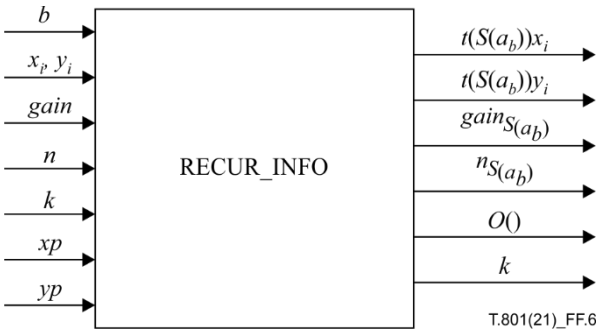

Figure F.6 – Parameters for the RECUR_INFO procedure

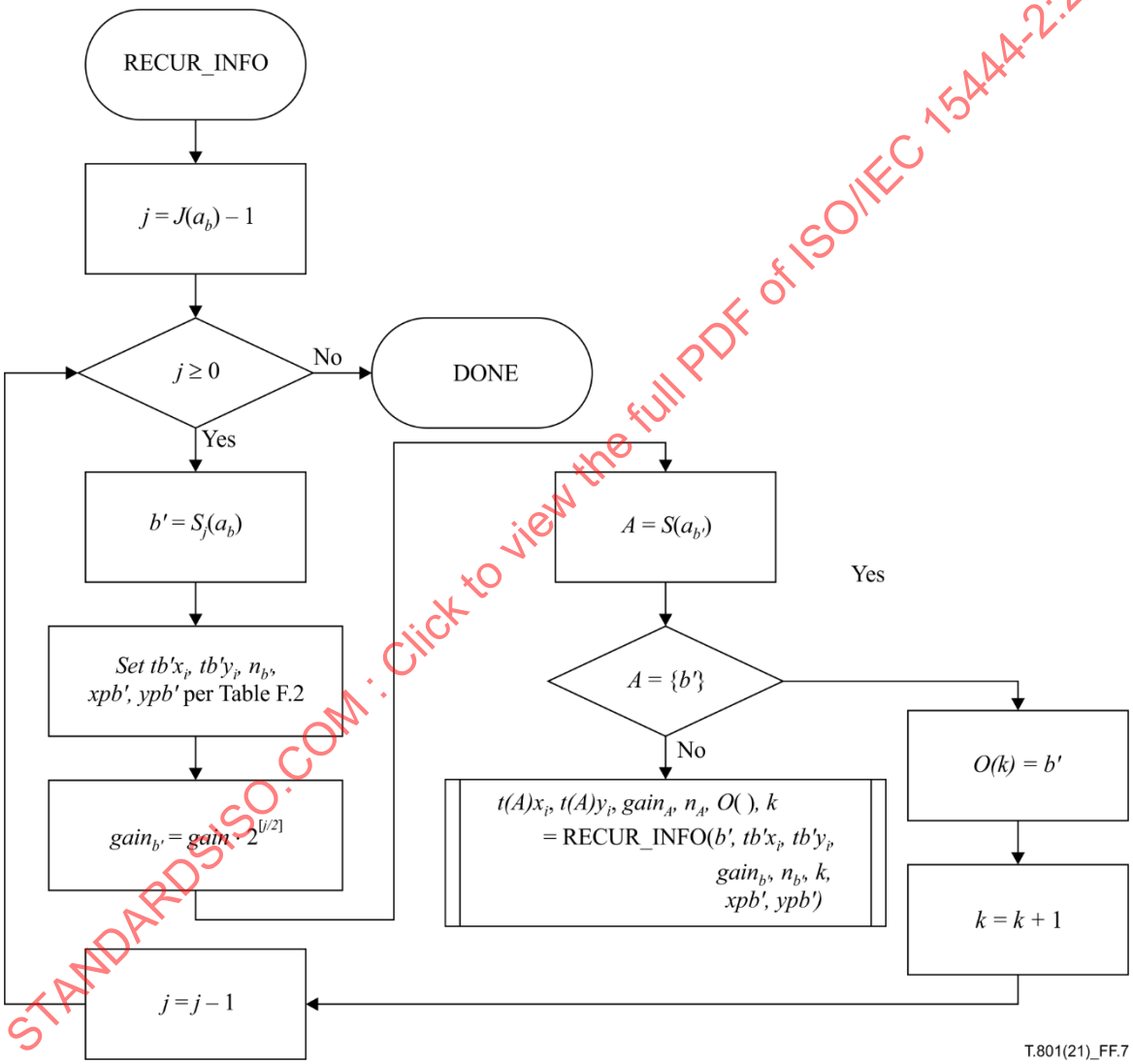

Figure F.7 – The RECUR_INFO procedure

Table F.2 – Quantities for sub-band info calculation

HorOrient($a_{b'}$)	VerOrient($a_{b'}$)	$tb'x_i$	$tb'y_i$	$n_{b'}$	xpb'	ypb'
L	L	$\lceil x_i/2 \rceil$	$\lceil y_i/2 \rceil$	$n + 1$	$xp/2$	$yp/2$
H	L	$\lceil x_i/2 \rceil$	$\lceil y_i/2 \rceil$	$n + 1$	$xp/2$	$yp/2$
L	H	$\lceil x_i/2 \rceil$	$\lceil y_i/2 \rceil$	$n + 1$	$xp/2$	$yp/2$
H	H	$\lceil x_i/2 \rceil$	$\lceil y_i/2 \rceil$	$n + 1$	$xp/2$	$yp/2$
L	X	$\lceil x_i/2 \rceil$	y_i	$n + 1/2$	$xp/2$	yp
H	X	$\lceil x_i/2 \rceil$	y_i	$n + 1/2$	$xp/2$	yp
X	L	x_i	$\lceil y_i/2 \rceil$	$n + 1/2$	xp	$yp/2$
X	H	x_i	$\lceil y_i/2 \rceil$	$n + 1/2$	xp	$yp/2$

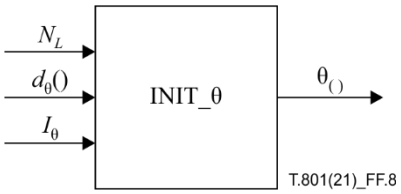

Figure F.8 – Parameters for the INIT_θ procedure

F.2.5 Updates to decomposition structure

Three arrays are used for specifying the decomposition structure for each tile-component. Each array is composed of two-bit values that are signalled in DFS and ADS markers (see clauses A.3.3 and A.3.4). The first array, $d_\theta(i)$, $i = 0, \ldots, I_\theta - 1$, is defined by the DOads and IOads ADS marker segment (see clause A.3.4) and is used by the INIT_θ procedure to determine the maximum number of sub-levels, θ(*lev*), in each decomposition level. Usage for this procedure is shown in Figure F.8 and the procedure itself is defined in Figure F.9. If the ADS marker segment is not defined for the current tile-component, the length I_θ is set to zero.

The values of $d_\theta(i)$ used to set all θ(*lev*) in this procedure should be non-zero and thus equal to 1, 2 or 3. Remaining levels not set before encountering the end of $d_\theta(i)$ are set to the last $d_\theta(i)$ entry.

The second array, $d_R(i)$, $i = 0, \ldots, I_R - 1$, is defined by Ddfs and Idfs DFS marker segments and specifies the dimensionality of each resolution level. The third array, $d_S(i)$, $i = 0, \ldots, I_S - 1$, is defined by the DSads and ISads ADS marker segment (see clause A.3.4) and specifies the sub-level decomposition structure within each decomposition level. Both of these arrays are used along with other inputs by the INIT_S_R and LEV_S routines. The I/O structure for these procedures are depicted in Figures F.10 and F.12 and the corresponding algorithmic structures are defined in Figures F.11 and F.13, Tables F.3 and F.4. As with I_θ, if the DFS or ADS marker segments are not defined for the current tile-component, I_R or I_S respectively is set to zero. When either I_R or I_S equal zero, the INIT_S_R routine will modify the respective arrays to allow full sub-level depths with joint horizontal and vertical decomposition splits for all sub-bands in the wavelet decomposition. The first purpose of these procedures is to determine the LUT $S(a_b)$ which defines how a sub-band a_b decomposes into other sub-bands. This LUT is defined so that $S(a_b)$ equals the set of sub-band indices for decomposed sub-bands from sub-band a_b. The length LUT $J(a_b)$ is also defined by these routines to be the number of sub-bands which decompose from sub-band a_b. Therefore, $S(a_b) = \{S_0(a_b), \ldots, S_{J(a_b)-1}(a_b)\}$, where $S_j(a_b)$ is the sub-band index of the j-th sub-band decomposed from a_b. Also, $S(a_b) = \{b\}$ when a_b is not further decomposed. This occurs when terminating zeros are indexed in $d_S(i)$ or the sub-level depth of sub-band a_b equals the maximum, θ(*lev*), allowed for its level. Finally, the notation $a_{S(a_b)}$ is used to denote the set $\{a_{S_0(a_b)}, \ldots, a_{S_{J(a_b)-1}(a_b)}\}$.

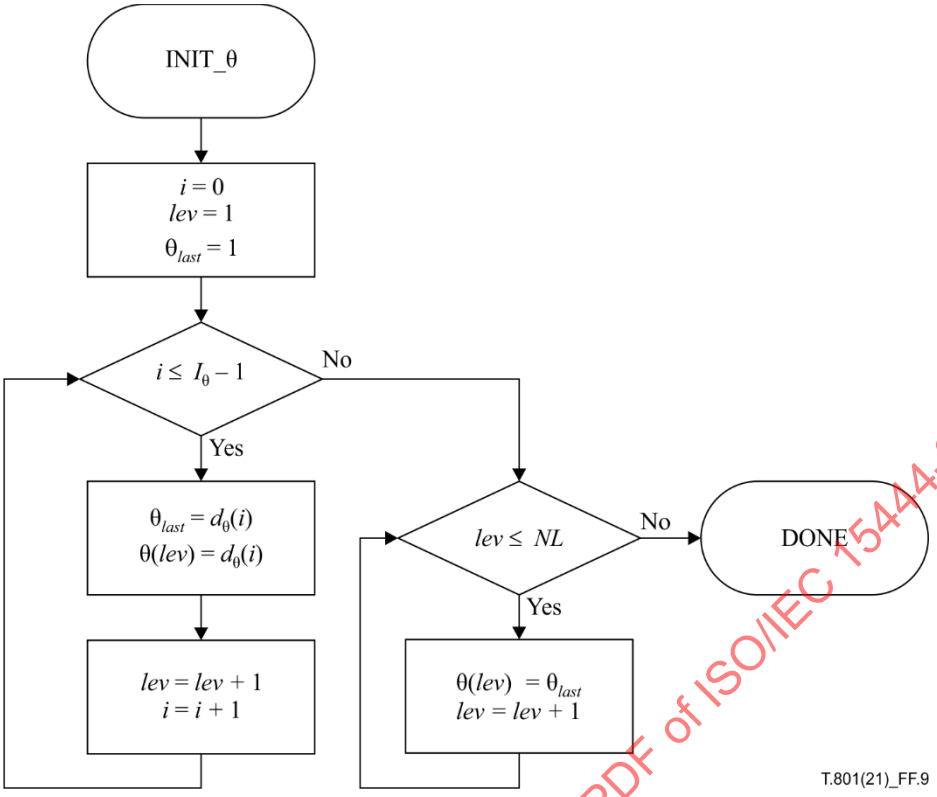

Figure F.9 – Procedure for setting maximum number of sub-levels, θ(*lev*)

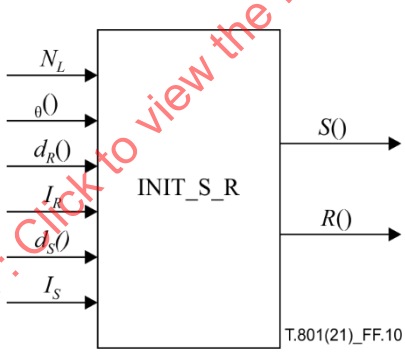

Figure F.10 – Parameters for the INIT_S_R procedure

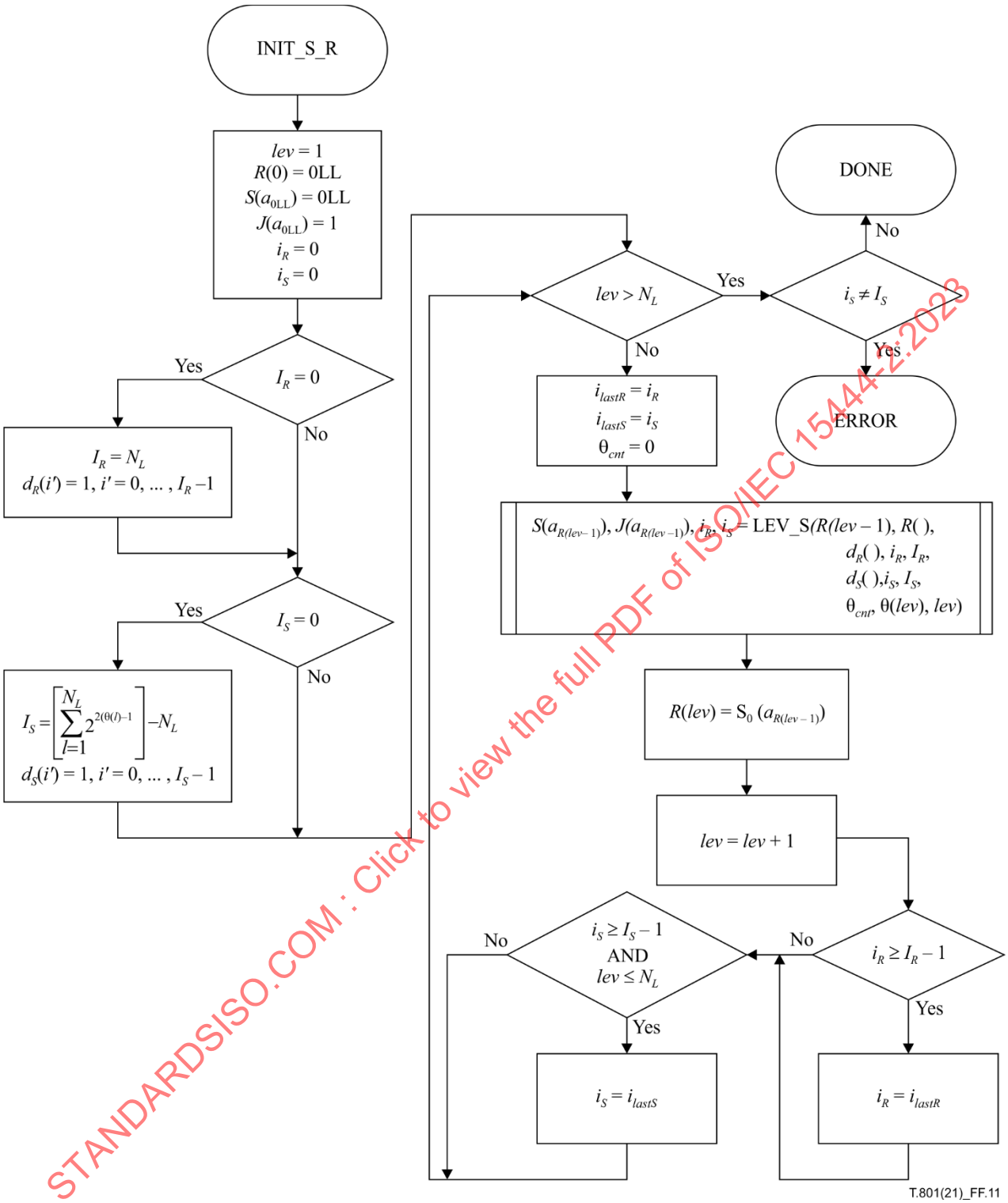

Figure F.11 – Upper level procedure for defining $S(a_b)$ and $R(lev)$

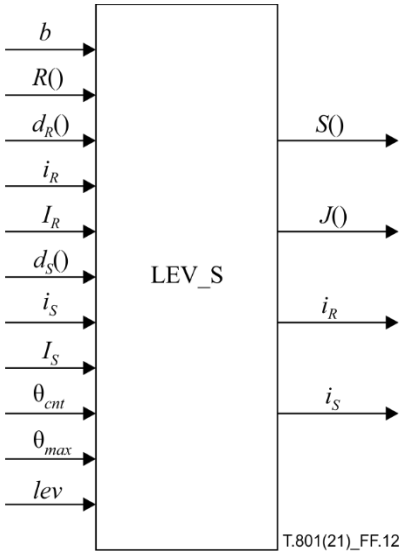

Figure F.12 – Parameters for the LEV_S procedure

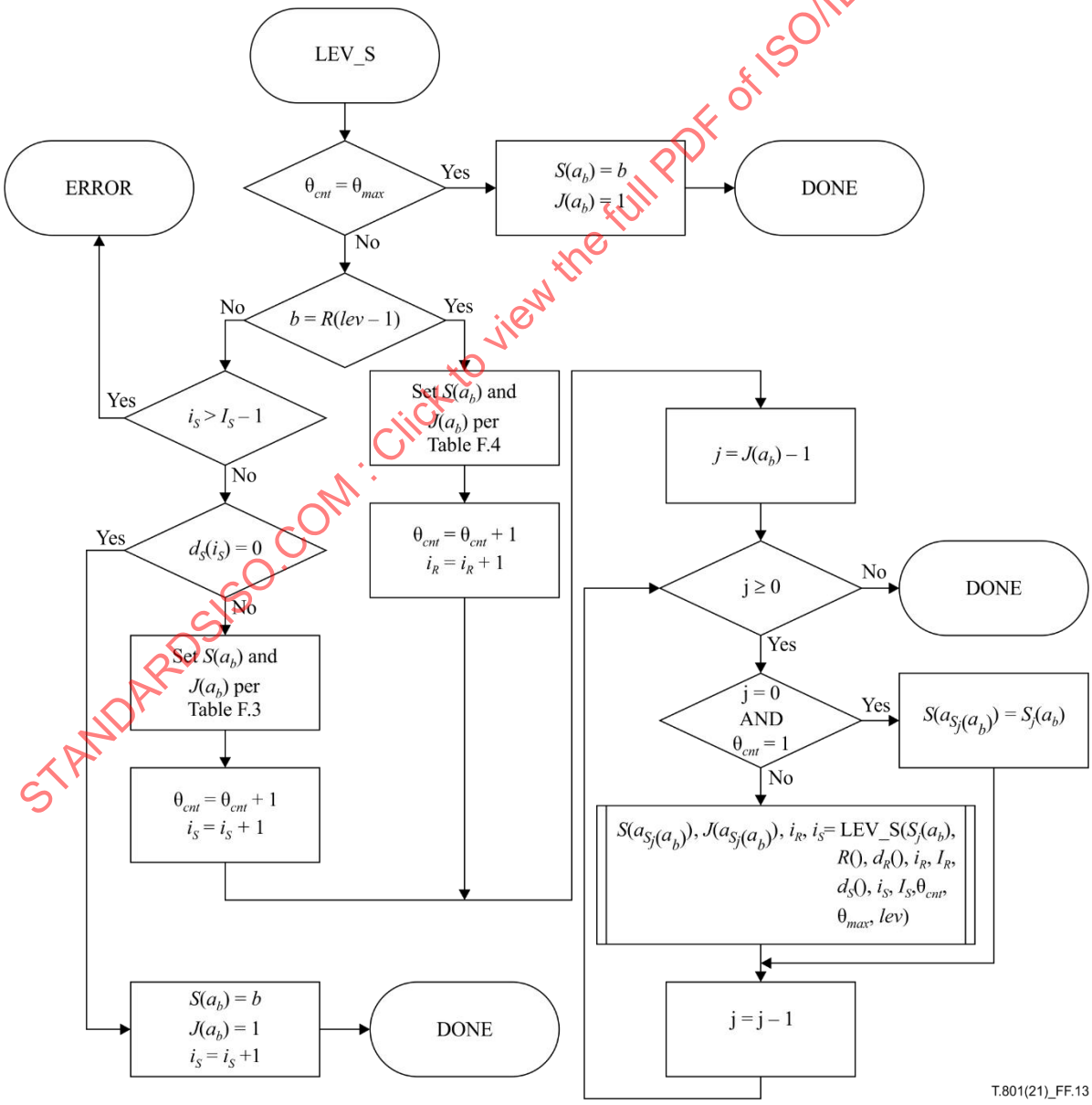

Figure F.13 – Procedure for defining $S(a_b)$

The final purpose of the INIT_S_R and LEV_S procedures is to define the $R(lev)$ LUT for each decomposition level. This LUT specifies the LL, LX, or XL sub-band orientation which results from the first sub-level of wavelet processing for a decomposition level. The corresponding sub-band is taken as the resolution at decomposition level *lev*. That is, resolution N_L-l*ev* is given by $a_{R(lev)}$.

Table F.3 – $S(a_b)$ and $J(a_b)$ as a function of $d_S(i)$

$d_S(i_S)$	$S(a_b)$ = Set of indices for decomposed sub-bands from a_b	$J(a_b)$ = Length of set $S(a_b)$
1	{*b*:LL, *b*:HL, *b*:LH, *b*:HH}	4
2	{*b*:LX, *b*:HX}	2
3	{*b*:XL, *b*:XH}	2

Table F.4 – $S(a_b)$ and $J(a_b)$ as a function of $d_R(i)$

$d_R(i_R)$	$S(a_b)$ = Set of indices for decomposed sub-bands from a_b	$J(a_b)$ = Length of set $S(a_b)$
1	{*lev*LL, *lev*HL, *lev*LH, *lev*HH}	4
2	{*lev*LX, *lev*HX}	2
3	{*lev*XL, *lev*XH}	2

NOTE – The INIT_S_R routine uses $d_S(i)$ and $d_R(i)$ array elements in order to define $S()$, $J()$, and $R()$ for all decomposition levels.

Figure F.14 illustrates a sample wavelet decomposition. In this decomposition, $N_L = 3$, $d_0() = 31$, $I_0 = 2$, $d_R() = 123$, $I_R = 3$, $d_S() = 320300203$, and $I_S = 9$. Table F.5 shows the various characteristics for this decomposition, including the $R()$ notation for each level, indicating that sub-band a_{0LL} represents resolution 3 (the original image), and that resolutions 2, 1 and 0 are represented by sub-bands a_{1LL}, a_{2LX} and a_{3XL} respectively. As in Rec. ITU-T T.800 | ISO/IEC 15444-1, precincts are specified with respect to these resolutions.

F.3 Inverse discrete wavelet transformation for general decompositions

The inverse transformation process is much like that described in Clause F.3 of Rec. ITU-T T.800 | ISO/IEC 15444-1. The only modifications necessary to provide arbitrary decomposition functionality are to the IDWT, 2D_SR, and 2D_INTERLEAVE procedures defined in that Annex.

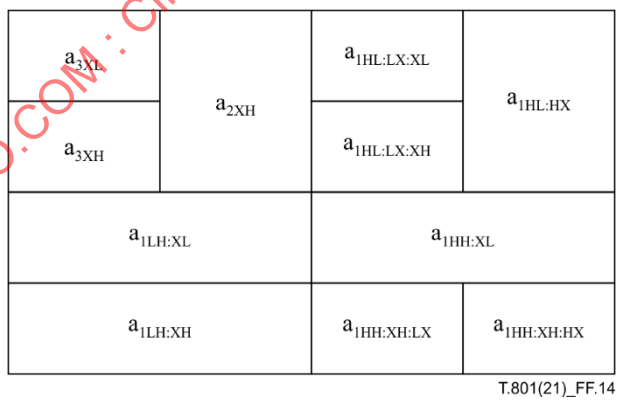

Figure F.14 – Sample wavelet decomposition with labelled sub-bands

Table F.5 – Characteristics for sample wavelet decomposition in Figure F.14

lev	θ(*lev*)	*S*()	Final Sub-bands (a_b)	HorOrient(a_b) / VerOrient(a_b)	PrimeOrient (a_b)	*R*(*lev*)
0	NA	NA	a_{0LL}	L/L	LL	0LL
		$S(a_{0LL})$ = {1LL, 1HL, 1LH, 1HH}				
		$S(a_{1LL})$ = {1LL}				
		$S(a_{1HL})$ = {1HL:LX, 1HL:HX}				
		$S(a_{1HL:LX})$ = {1HL:LX:XL, 1HL:LX:XH}				
		$S(a_{1HL:LX:XL})$ = {1HL:LX:XL}	$a_{1HL:LX:XL}$	X/L	HL	
		$S(a_{1HL:LX:XH})$ = {1HL:LX:XH}	$a_{1HL:LX:XH}$	X/H	HL	
		$S(a_{1HL:HX})$ = {1HL:HX}	$a_{1HL:HX}$	H/X	HL	
1	3	$S(a_{1LH})$ = {1LH:XL, 1LH:XH}				1LL
		$S(a_{1LH:XL})$ = {1LH:XL}	$a_{1LH:XL}$	X/L	LH	
		$S(a_{1LH:XH})$ = {1LH:XH}	$a_{1LH:XH}$	X/H	LH	
		$S(a_{1HH})$ = {1HH:XL, 1HH:XH}				
		$S(a_{1HH:XL})$ = {1HH:XL}	$a_{1HH:XL}$	X/L	HH	
		$S(a_{1HH:XH})$ = {1HH:XH:LX, 1HH:XH:HX}				
		$S(a_{1HH:XH:LX})$ = {1HH:XH:LX}	$a_{1HH:XH:LX}$	L/X	HH	
		$S(a1HH:XH:HX)$ = {1HH:XH:HX}	$a_{1HH:XH:HX}$	H/X	HH	
		$S(a_{1LL})$ = {2LX, 2HX}				
2	1	$S(a_{2LX})$ = {2LX}				2LX
		$S(a_{2HX})$ = {2HX}	a_{2HX}	H/X	HX	
		$S(a_{2LX})$ = {3XL, 3XH}				
3	1	$S(a_{3XL})$ = {3XL}	a_{3XL}	X/L	XL	3XL
		$S(a_{3XH})$ = {3XH}	a_{3XH}	X/H	XH	

F.3.1 Modified IDWT procedure

The IDWT procedure is almost the same as that in Rec. ITU-T T.800 | ISO/IEC 15444-1. Differences deal with changes in the call to the modification of the 2D_SR procedure (MOD_2D_SR) as well as the extra calls to INIT_θ and INIT_S_R. Nevertheless, the modified usage is depicted in Figure F.15 and the actual procedure is shown in Figure F.16. Sub-bands stored in the codestream are provided to MOD_2D_SR in the same order as provided by the order LUT *O()* defined in clause F.2.4.

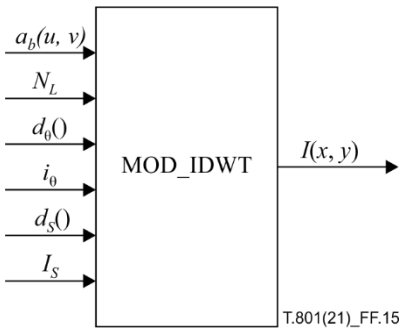

Figure F.15 – Parameters for the MOD_IDWT procedure

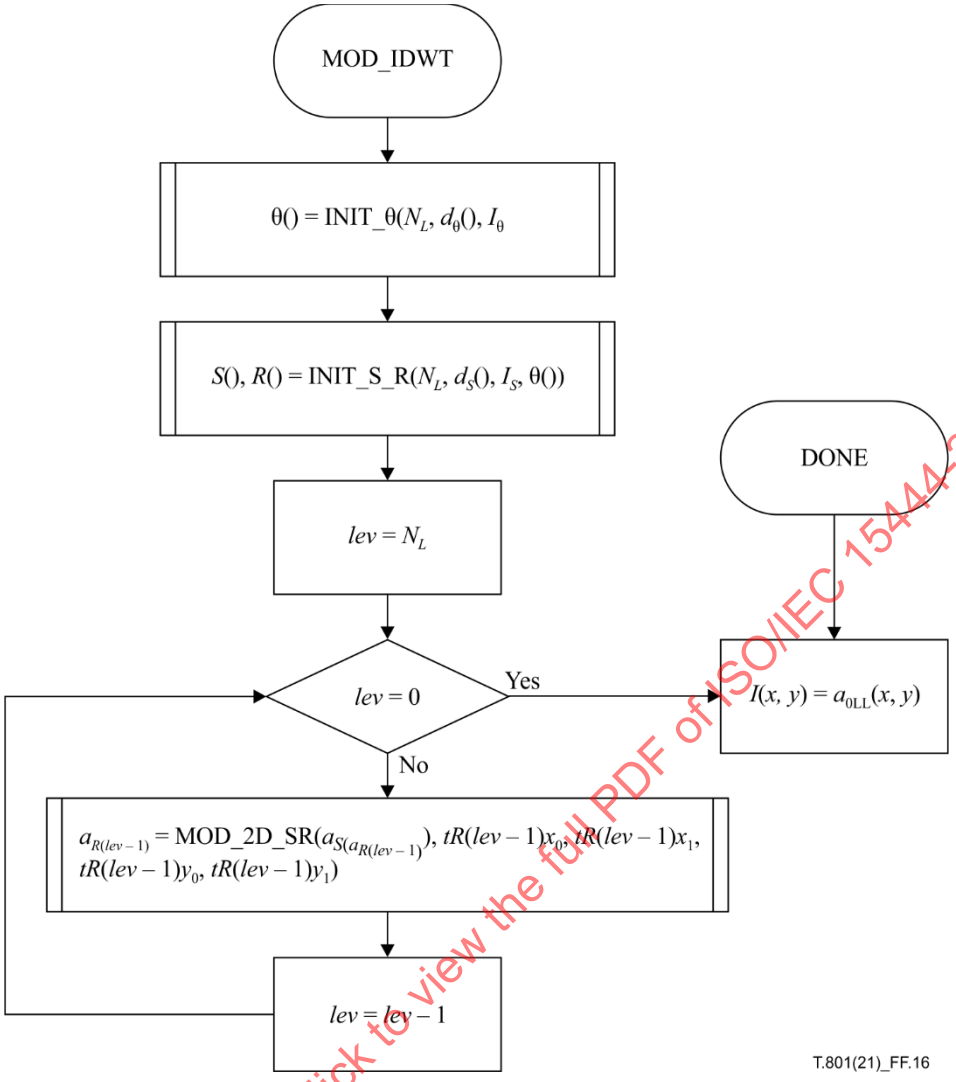

Figure F.16 – The MOD_IDWT procedure

F.3.2 Modified 2D_SR procedure

Major changes are required for 2D_SR from that in Rec. ITU-T T.800 | ISO/IEC 15444-1. This procedure is composed of operations which combine 2 or 4 sub-bands into a resulting sub-band. This procedure shall also handle such processing throughout all sub-levels inside a decomposition level. To accommodate such processing, a recursive structure is used for this procedure. The parameters necessary for this procedure are shown in Figure F.17 and the new procedure itself is diagrammed in Figure F.18.

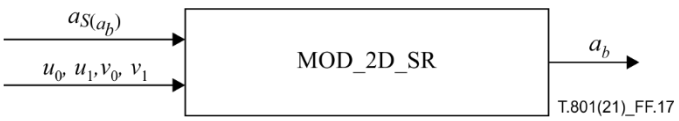

Figure F.17 – Parameters for the MOD_2D_SR procedure

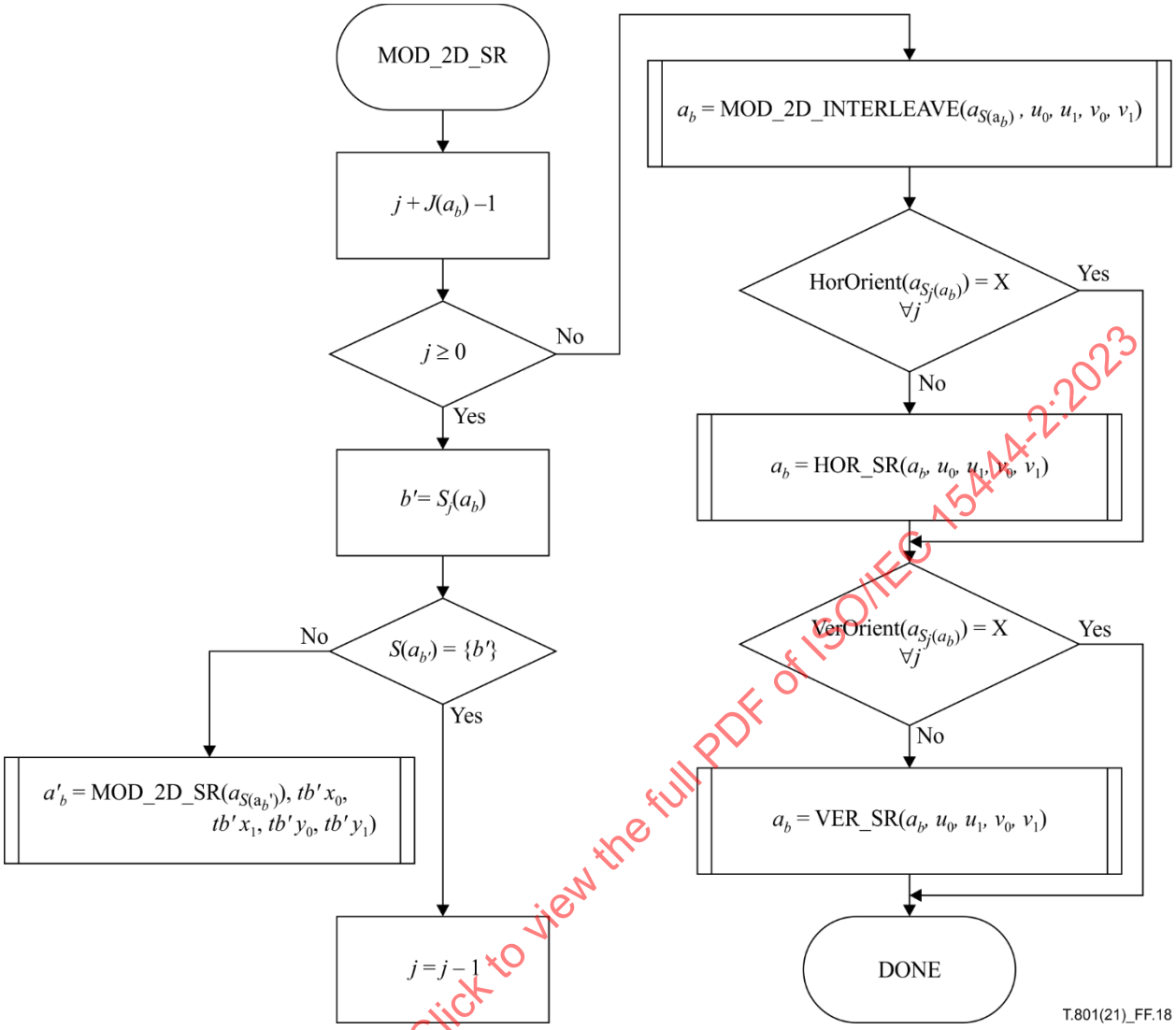

Figure F.18 – The MOD_2D_SR procedure

F.3.3 Modified 2D_INTERLEAVE procedure

Significant changes are also necessary for the 2D_INTERLEAVE procedure. These changes are due to both sub-level processing and disjoint horizontal/vertical sub-band splits. This procedure is shown in both Figures F.19 and F.20. As shown in the latter of these two figures, this routine now decides which of three lower level procedures shall be used to interleave wavelet samples. The values of u_0, u_1, v_0 and v_1 in each of these three lower level procedures are those of tbx_0, tbx_1, tby_0 and tby_1 as redefined in clause F.2.4, where a_b is the sub-band to be interleaved and eventually reconstructed.

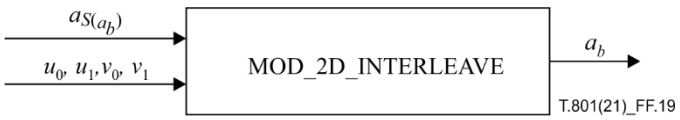

Figure F.19 – Parameters for the MOD_2D_INTERLEAVE procedure

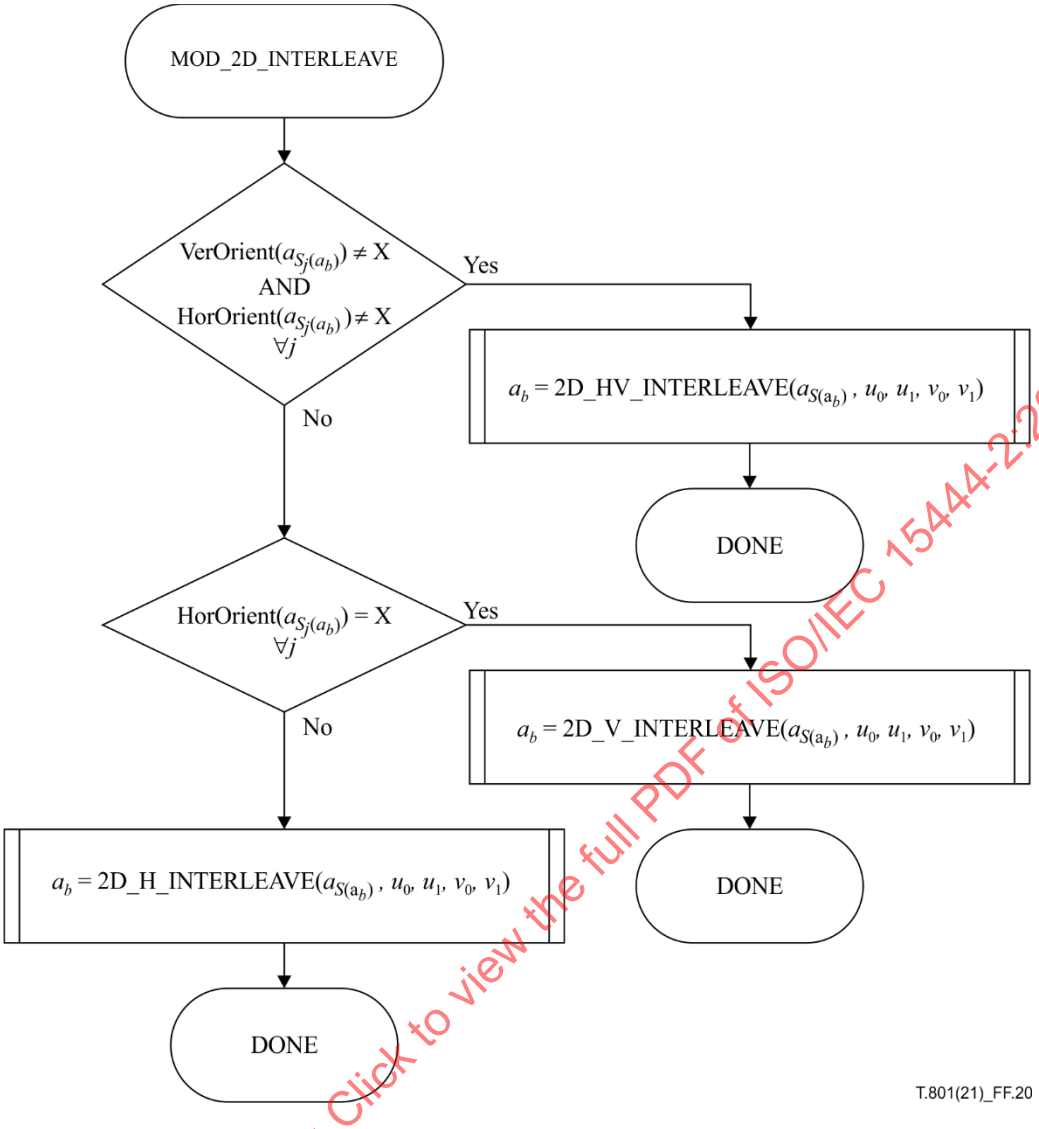

Figure F.20 – The MOD_2D_INTERLEAVE procedure

F.3.3.1 The 2D_HV_INTERLEAVE procedure

The 2D_HV_INTERLEAVE procedure is similar to the 2D_INTERLEAVE procedure from Rec. ITU-T T.800 | ISO/IEC 15444-1. Usage for this procedure is shown in Figure F.21 and the actual procedure is shown in Figure F.22.

Figure F.21 – Parameters for the 2D_HV_INTERLEAVE procedure

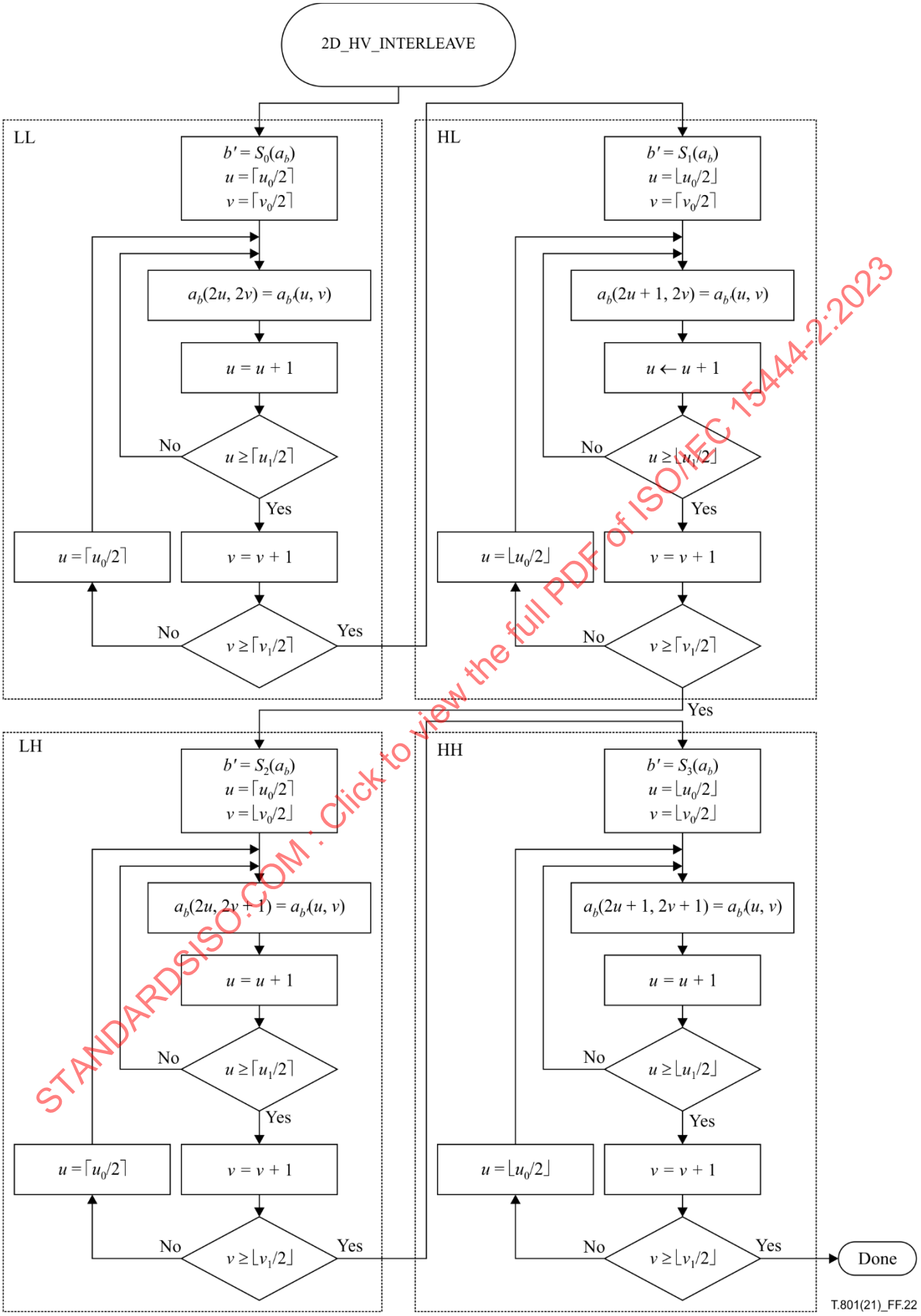

Figure F.22 – The 2D_HV_INTERLEAVE procedure

Figure F.23 – Parameters for the 2D_H_INTERLEAVE procedure

F.3.3.2 The 2D_H_INTERLEAVE procedure

The 2D_H_INTERLEAVE procedure defined in Figures F.23 and F.24 is used to accommodate disjoint processing along just the horizontal direction. As such, this procedure requires roughly half the 2D_HV_INTERLEAVE procedure logic to interleave samples in just the horizontal direction.

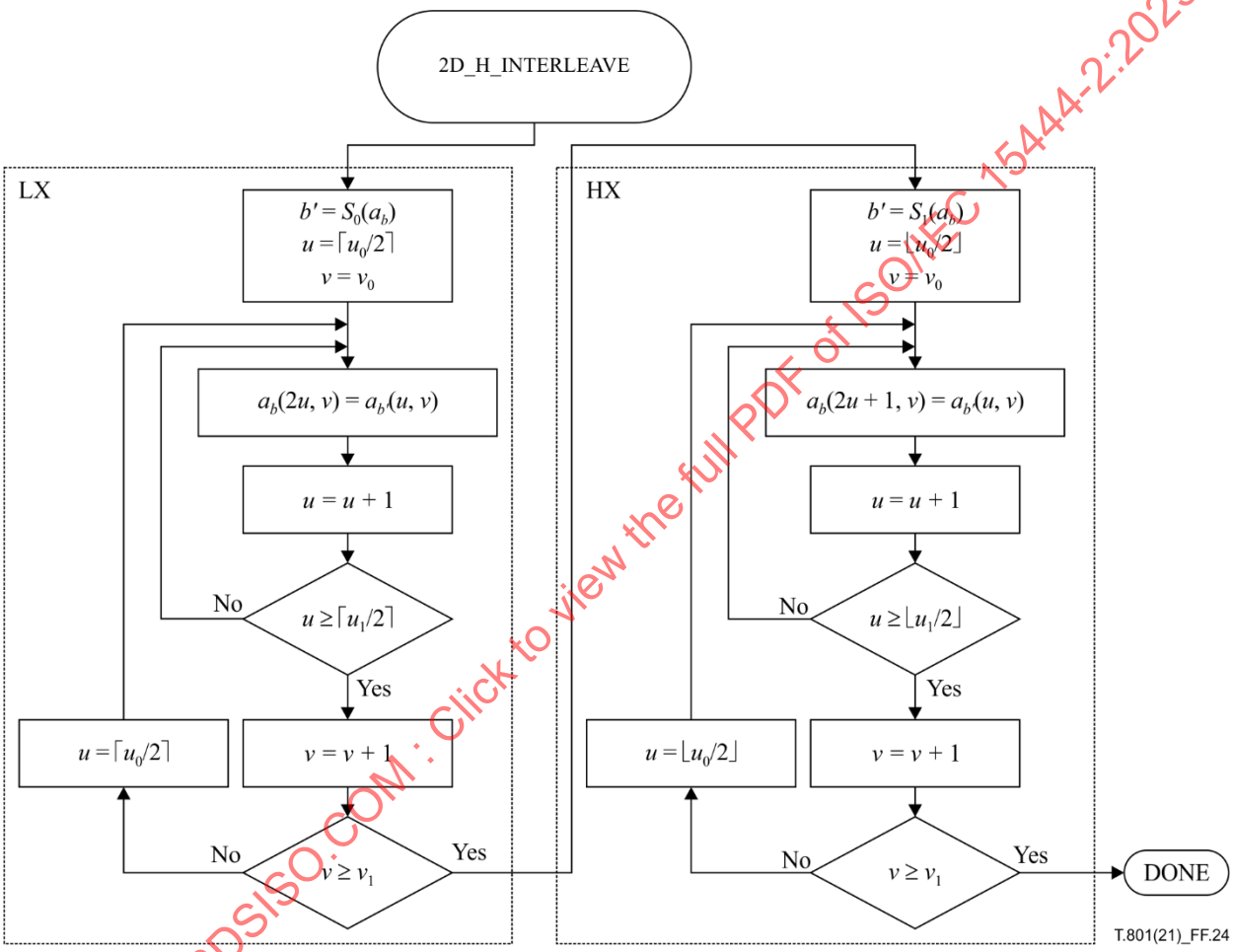

Figure F.24 – The 2D_H_INTERLEAVE procedure

F.3.3.3 The 2D_V_INTERLEAVE procedure

The procedure for interleaving samples due to disjoint wavelet processing in just the vertical direction is quite like that for the procedure defined above in clause F.3.3.2. The procedure for this case is depicted in Figures F.25 and F.26.

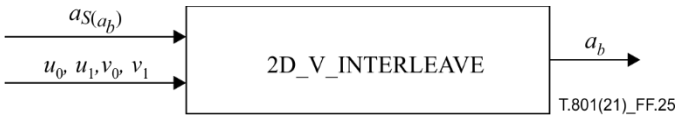

Figure F.25 – Parameters for the 2D_V_INTERLEAVE procedure

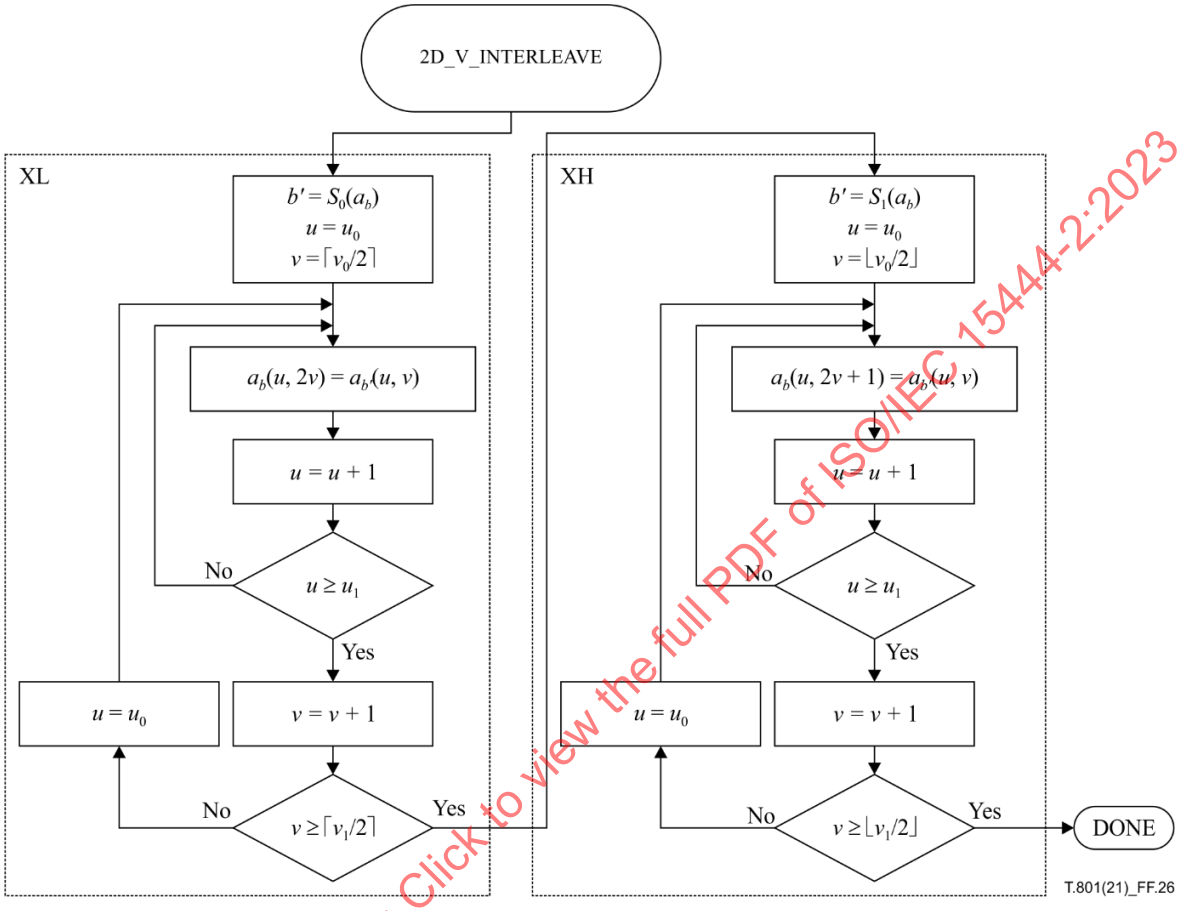

Figure F.26 – The 2D_V_INTERLEAVE procedure

F.4 Forward discrete wavelet transformation for general decompositions (informative)

Similar to the inverse transformation process, forward wavelet transformation requires changes to only the FDWT, 2D_SD, and 2D_DEINTERLEAVE Rec. ITU-T T.800 | ISO/IEC 15444-1 procedures.

F.4.1 Modified FDWT procedure

Like the MOD_IDWT procedure, the FDWT remains much like that in Rec. ITU-T T.800 | ISO/IEC 15444-1. The parameters for this procedure are shown in Figure F.27 and the structure of the procedure is shown in Figure F.28.

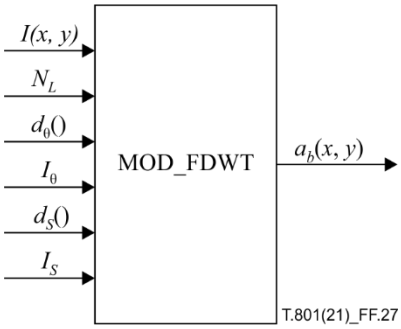

Figure F.27 – Parameters for the MOD_FDWT procedure

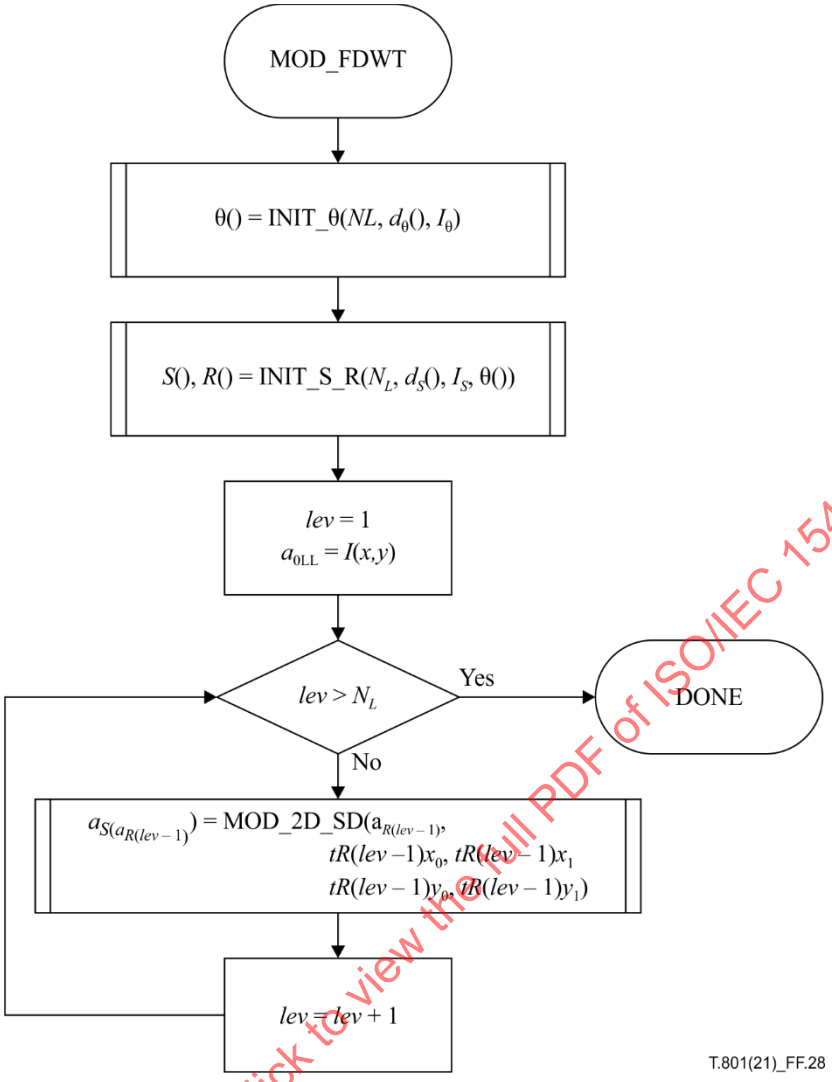

Figure F.28 – The MOD_FDWT procedure

F.4.2 Modified 2D_SD procedure

Major changes are required for 2D_SD from that in Rec. ITU-T T.800 | ISO/IEC 15444-1. This procedure is composed of operations which split one sub-band into 2 or 4 resulting sub-bands. This procedure shall also handle such processing throughout all sub-levels inside a decomposition level. To accommodate such processing, a recursive structure is used for this procedure. The parameters necessary for this procedure are shown in Figure F.29 and the new procedure itself is diagrammed in Figure F.30.

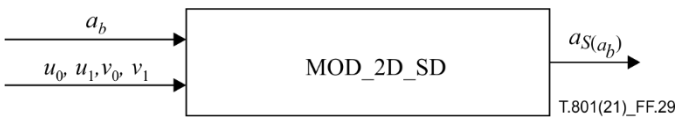

Figure F.29 – Parameters for the MOD_2D_SD procedure

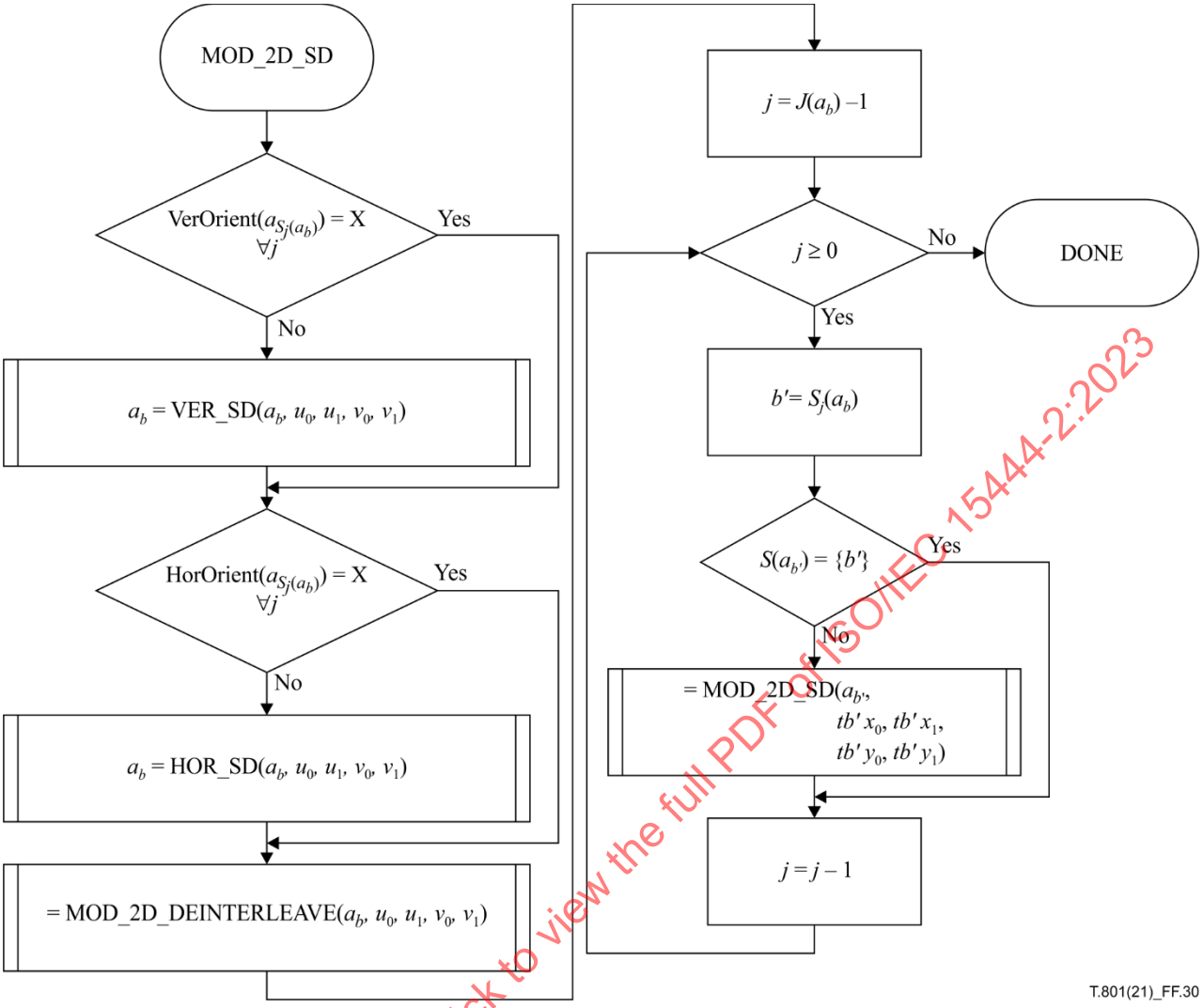

Figure F.30 – The MOD_2D_SD procedure

F.4.3 Modified 2D_DEINTERLEAVE procedure

Significant changes were also necessary for the 2D_DEINTERLEAVE procedure. These changes are due to both sub-level processing and disjoint horizontal/vertical sub-band splits. This procedure is shown in both Figures F.31 and F.32. As shown in the latter of these two figures, this routine now decides which of three lower level procedures shall be used to deinterleave wavelet samples. As with the MOD_2D_INTERLEAVE procedure, the values of u_0, u_1, v_0 and v_1 in each of these three lower level procedures are those of tbx_0, tbx_1, tby_0 and tby_1 as defined in clause F.2.4 for the sub-band a_b which is being decomposed/deinterleaved.

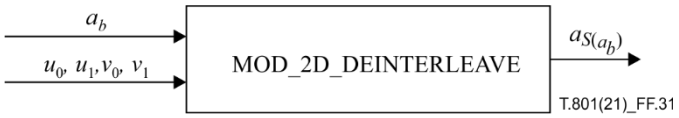

Figure F.31 – Parameters for the MOD_2D_DEINTERLEAVE procedure

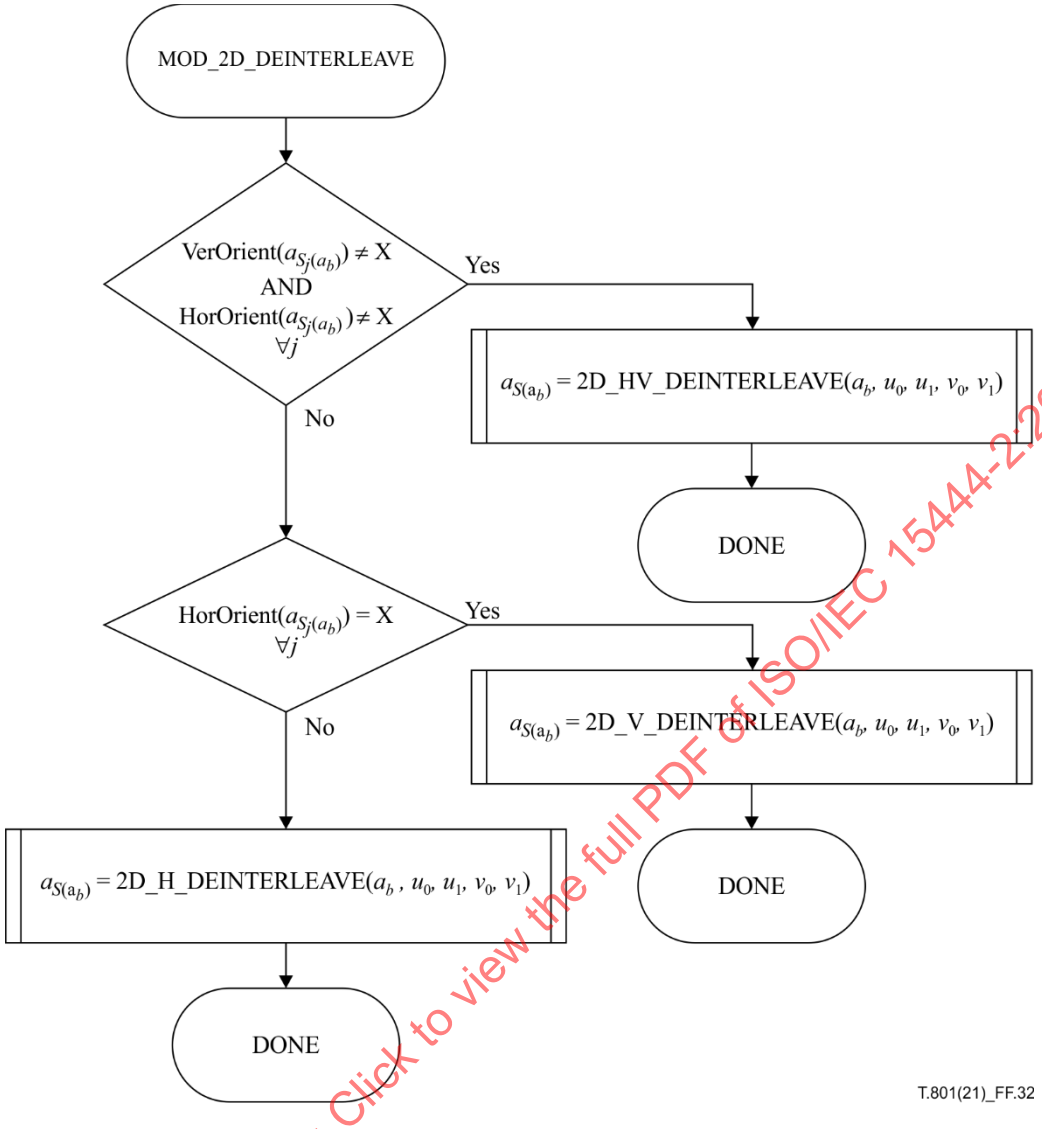

Figure F.32 – The MOD_2D_DEINTERLEAVE procedure

F.4.3.1 The 2D_HV_DEINTERLEAVE procedure

The 2D_HV_DEINTERLEAVE procedure is similar to the 2D_DEINTERLEAVE procedure from Rec. ITU-T T.800 | ISO/IEC 15444-1. Usage for this procedure is shown in Figure F.33 and the actual procedure is shown in Figure F.34.

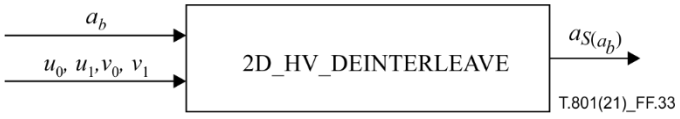

Figure F.33 – Parameters for the 2D_HV_DEINTERLEAVE procedure

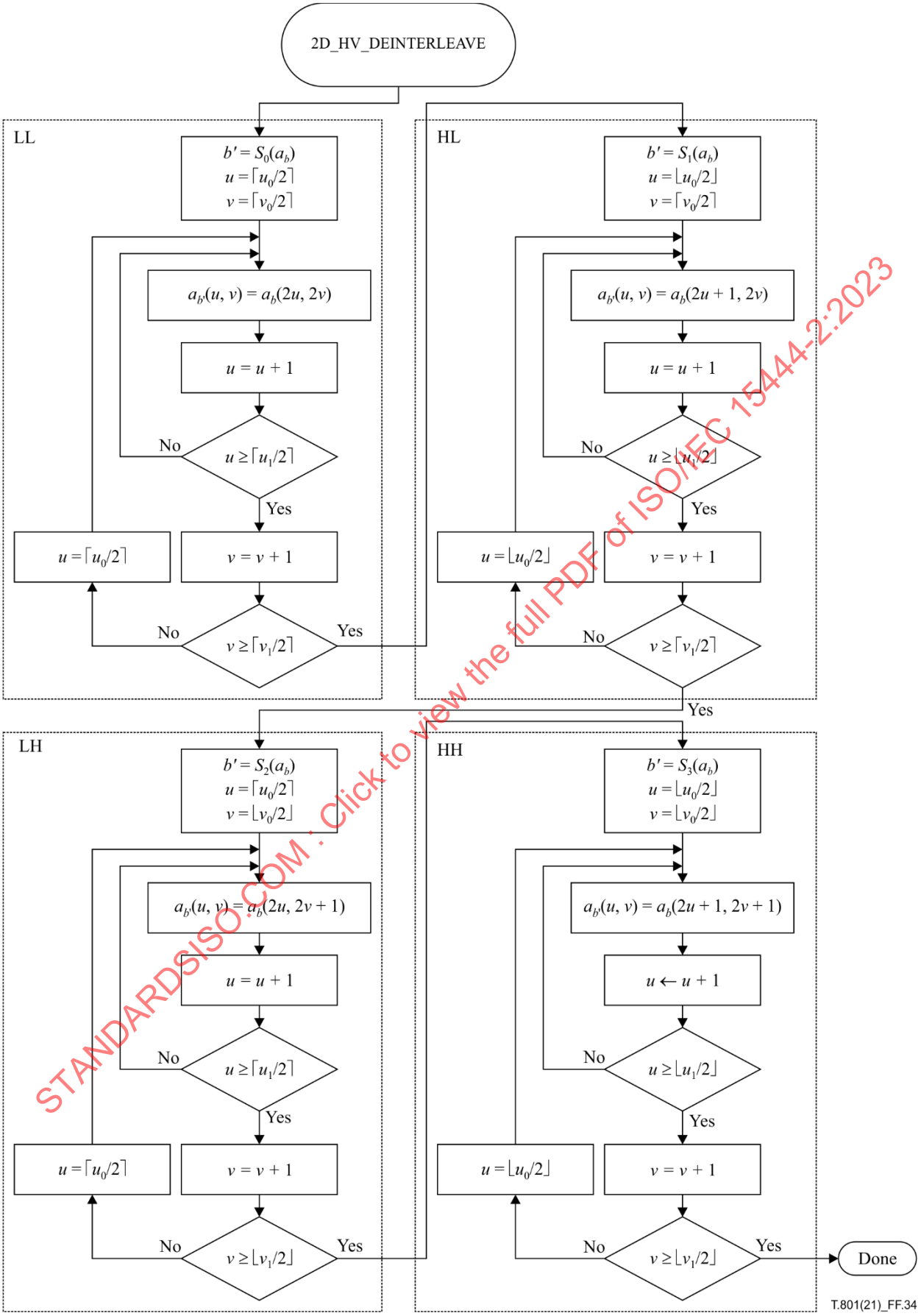

Figure F.34 – The 2D_HV_DEINTERLEAVE procedure

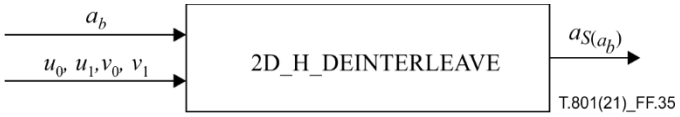

Figure F.35 – Parameters for the 2D_H_DEINTERLEAVE procedure

F.4.3.2 The 2D_H_DEINTERLEAVE procedure

The 2D_H_DEINTERLEAVE procedure is used to accommodate disjoint processing along just the horizontal direction. As such, this procedure requires roughly half the logic in the 2D_HV_DEINTERLEAVE procedure above. The diagram of this procedure is given in Figures F.35 and F.36.

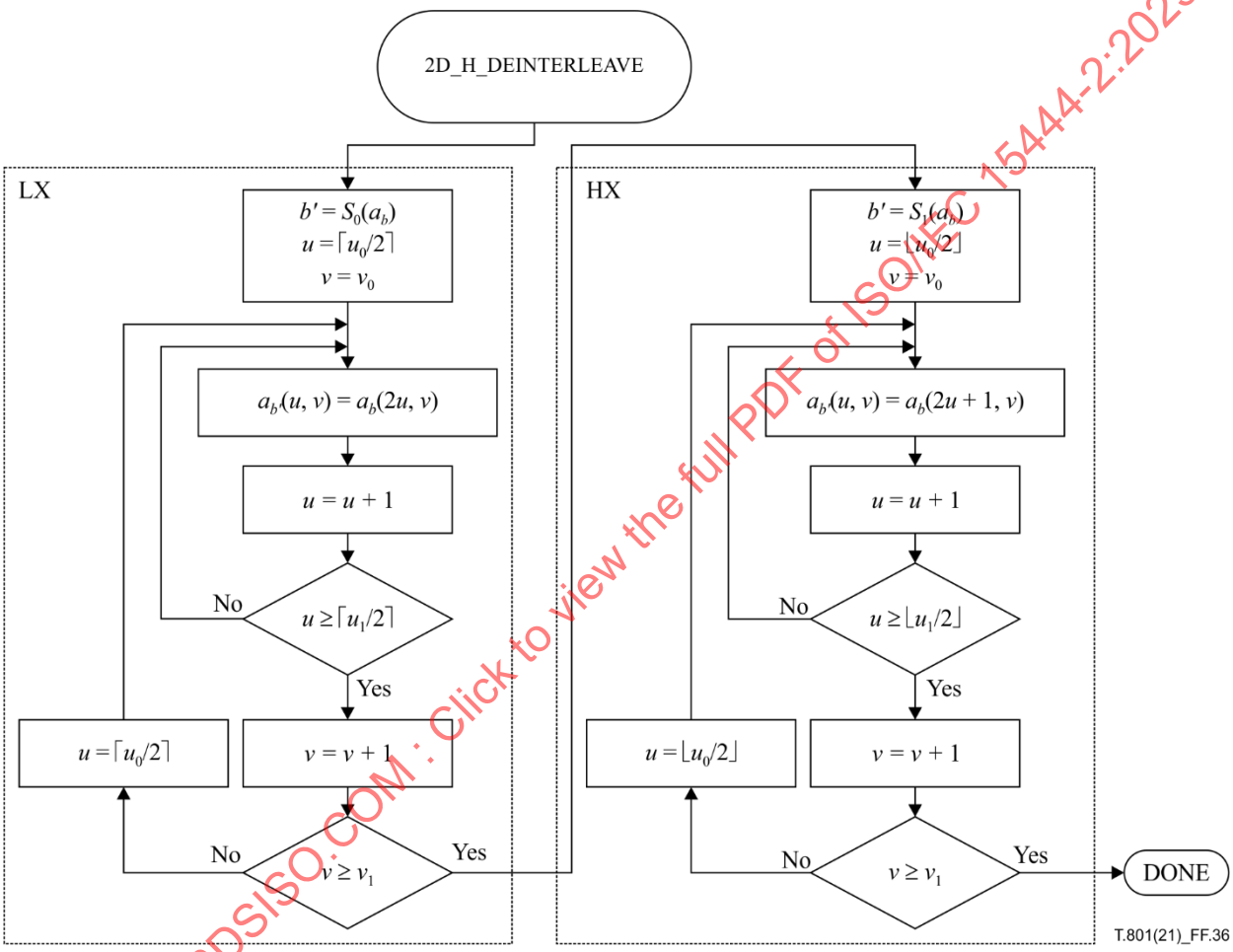

Figure F.36 – The 2D_H_DEINTERLEAVE procedure

F.4.3.3 The 2D_V_DEINTERLEAVE procedure

The procedure for deinterleaving samples due to disjoint wavelet processing in just the vertical direction is quite like that for the procedure defined in clause F.4.3.2. The procedure for this case is depicted in Figures F.37 and F.38.

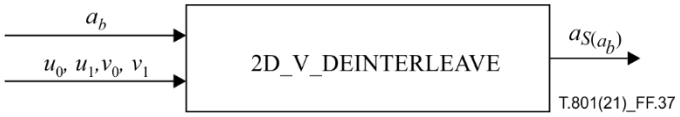

Figure F.37 – Parameters for the 2D_V_DEINTERLEAVE procedure

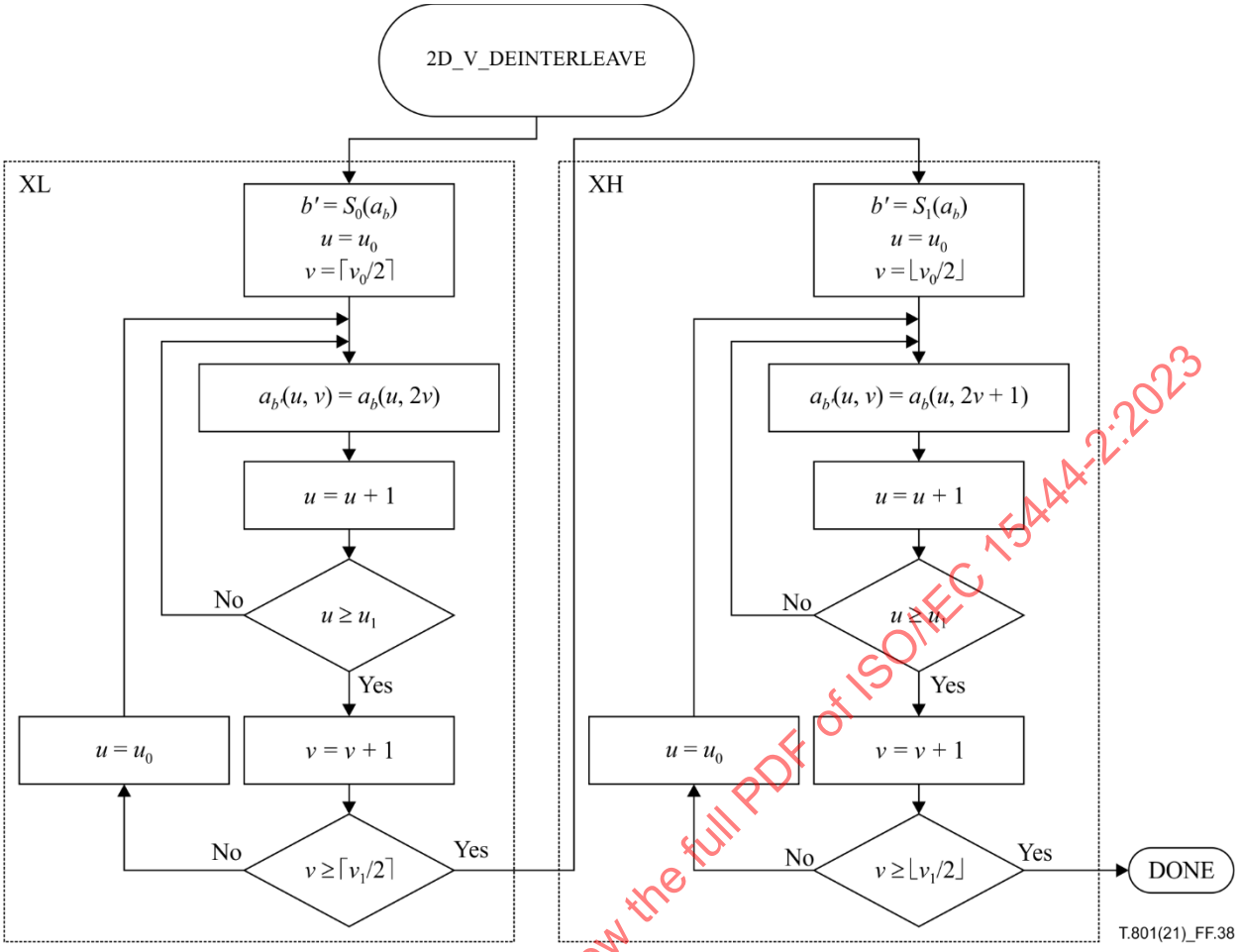

Figure F.38 – The 2D_V_DEINTERLEAVE procedure

Annex G

Whole-sample symmetric transformation of images, extensions

(This annex forms an integral part of this Recommendation | International Standard.)

This Recommendation | International Standard uses a transformation of tile components.

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

This annex specifies two extensions of the one-dimensional sub-band reconstruction procedure 1D_SR (see Rec. ITU-T T.800 | ISO/IEC 15444-1): the 1D_SR_WS procedure and the 1D_SD_WS procedure for whole-sample symmetric (WS) wavelet transformations.

G.1 Wavelet transformation parameters, definitions and normalizations

Table G.1 lists the parameters used in the wavelet transformations that are signalled in the codestream. Signalling for these parameters is defined in the ATK marker segment (see clause A.3.5).

Table G.1 – Parameters for wavelet transformations

Parameter tag	Meaning	Value(s)
Coeff_Typ	Numerical type of lifting step coefficients	8-bit signed integer 16-bit signed integer 32-bit float 64-bit float 128-bit float
Filt_Cat	Wavelet transformation category	arbitrary (*ARB*) whole-sample symmetric (*WS*)
WT_Typ	Wavelet transformation type	irreversible (*IRR*) reversible (*REV*)
m_{init}	Update characteristic of first reconstruction lifting step	even-subsequence update (0) odd-subsequence update (1)
N_{LS}	Number of lifting steps	8-bit unsigned integer
ε_s	Base-2 scaling exponent for lifting step *s* (reversible transformations only)	8-bit unsigned integer: defined for $0 \le s < N_{LS}$
β_s	Additive residue for lifting step *s* (reversible transformations only)	*Coeff_Typ*: defined for $0 \le s < N_{LS}$
K	Scaling factor (irreversible transformations only)	Coeff_Typ
L_s	Number of lifting coefficients for lifting step *s*	8-bit unsigned integer: defined for $0 \le s < N_{LS}$
$\alpha_{s,k}$	*kth* lifting coefficient for lifting step *s*	*Coeff_Typ*: defined for $0 \le s < N_{LS}$ $0 \le k < L_S$

G.2 Whole-sample symmetric (WS) wavelet transformations reconstruction

The procedures specified in this clause apply only in the case of ATK marker segments for which *Filt_Cat = WS*.

G.2.1 Normalization of WS wavelet transformations

This clause specifies conditions on parameters that a compliant codestream shall satisfy.

Define D_s to be the sum of the lifting coefficients $\alpha_{s,k}$ for lifting step *s*, $0 \le s < N_{LS}$ (normalized in the case of reversible transforms):

$$D_s = 2 \cdot \sum_{k=0}^{L_s-1} \alpha_{s,k} \quad \text{if} \quad WT\_Typ = IRR$$

$$D_s = \frac{2}{2^{\varepsilon_s}} \sum_{k=0}^{L_s-1} \alpha_{s,k} \quad \text{if} \quad WT\_Typ = REV$$

Define parameters B_s recursively:

$$B_s = D_s B_{s-1} + B_{s-2}, \quad \text{for } s = 0 \text{ to } N_{LS^{-1}} \tag{G-1}$$

with the initial conditions being: $B_{-1} = 1$ and $B_{-2} = 1$.

G.2.1.1 Normalization of reversible wavelet transformation

For reversible wavelet transformations (*WT_Typ=REV*), the parameters B_s shall satisfy one of the following conditions:

$$B_{N_{LS^{-2}}} = 1 \quad \text{if} \quad m_{init} = 1, \quad \text{or}$$

$$B_{N_{LS^{-1}}} = 1 \quad \text{if} \quad m_{init} = 0$$

G.2.1.2 Normalization of irreversible wavelet transformation

For irreversible wavelet transformations (*WT_Typ = IRR*), the parameters B_s and the scaling parameter, K, shall satisfy one of the following conditions:

$$B_{N_{LS^{-2}}} = K \quad \text{if} \quad m_{init} = 1, \quad \text{or}$$

$$B_{N_{LS^{-1}}} = K \quad \text{if} \quad m_{init} = 0$$

G.2.2 One-dimensional sub-band reconstruction procedure for WS wavelet transformations

The one-dimensional sub-band reconstruction (1D_SR_WS) procedure is implemented as a sequence of primitive lifting steps, which alternately modify odd-indexed samples with a weighted sum of even-indexed samples and even-indexed samples with a weighted sum of odd-indexed samples.

G.2.2.1 The 1D_SR_WS procedure

As illustrated in Figure G.1, the 1D_SR_WS procedure takes as input a one-dimensional array, *Y*, of interleaved lowpass and highpass coefficients, the index i_0 of the first sample in array *Y*, the index i_1 of the sample following the last sample in array *Y*. It produces as output an array, *X*, with the same indices (i_0, i_1).

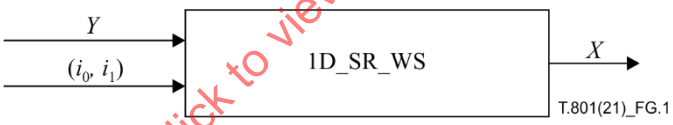

Figure G.1 – Parameters of the 1D_SR_WS procedures

For signals of length one (i.e., $i_0 = i_1 - 1$), the 1D_SR_WS procedure sets the value of $X(i_0)$ to $X(i_0) = Y(i_0)$ if i_0 is an even integer, and to $X(i_0) = Y(i_0)/2$ if i_0 is an odd integer.

For signals of length greater than or equal to two (i.e., $i_0 < i_1 - 1$), as illustrated in Figure G.2, the 1D_SR_WS procedure applies the 1D_FILTR_WS procedure to *Y* to produce the reconstructed signal, *X*.

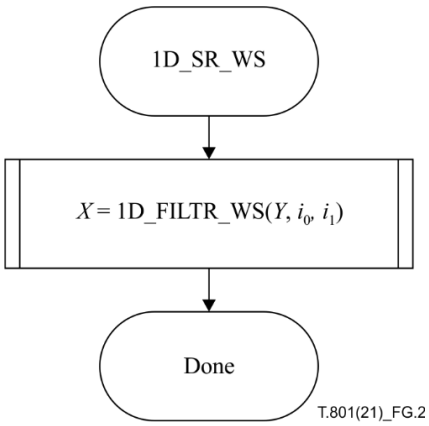

Figure G.2 – The 1D_SR_WS procedure

NOTE – Unlike in Rec. ITU-T T.800 | ISO/IEC 15444-1, the 1D_SR_WS procedure does not extend the signal prior to applying the 1D_FILTR_WS procedure. Instead, a procedure equivalent to the extension procedure is included in the 1D_FILTR_WS procedure.

G.2.2.2 The 1D_FILTR_WS procedures

Two versions of the reconstruction procedure (1D_FILTR_WS) are specified, depending on whether the transformation is reversible or not (*WT_Typ=REV* or *IRR*). In the following two subclauses, the function $PSE_O(i)$ equals the function $PSE_O(i, i_0, i_1)$ specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation F-4.

G.2.2.2.1 The reversible one-dimensional reconstruction (1D_FILTR_WS) procedure

As shown in Figure G.3, the input parameters to procedure 1D_FILTR_WS are V, i_0, i_1, N_{LS}, and $\alpha_{s,k}$, β_s, ε_s, L_s for $s = 0, 1, 2, \ldots, N_{LS} - 1$ and $k = 0, 1, 2, \ldots, L_s - 1$.

Figure G.3 – Parameters of the 1D_FILTR_WS procedure

The 1D_FILTR_WS procedure starts with the following N_{LS} lifting steps, where the variable s decreases from $N_{LS} - 1$ to zero (for $s = N_{LS} - 1, N_{LS} - 2, \ldots, 1, 0$):

$$V(2n + m_s) = V(2n + m_s) - \left\lfloor \frac{\left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot (V(PSE_O(2n+m_s-(2k+1))) + V(PSE_O(2n+m_s+(2k+1))))\right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor \quad \text{(G-2)}$$

where $m_{N_{LS-1}} = m_{init}$ and $m_s = 1 - m_{s+1}$ indicates whether the sth lifting step updates even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), where L_s is the number of lifting coefficients for lifting step s, and where the range of n is defined by $i_0 \le 2n + m_s < i_1$.

The values of $V(k)$ such that $i_0 \le k < i_1$ form the output $W(k)$ of the 1D_FILTR procedure:

$$W(k) = V(k) \quad \text{(G-3)}$$

G.2.2.2.2 The irreversible one-dimensional reconstruction (1D_FILTR_WS) procedure

As shown in Figure G.4, the input parameters to procedure 1D_FILTR_WS are V, i_0, i_1, K, N_{LS}, and $\alpha_{s,k}$, L_s for $s = 0, 1, 2, \ldots, N_{LS} - 1$ and $k = 0, 1, 2, \ldots, L_s - 1$.

Figure G.4 – Parameters of the 1D_FILTR_WS procedure

The 1D_FILTR_WS procedure starts with two scaling steps:

$$V(2n) = K \cdot V(2n) \text{for} i_0 \le 2n < i_i \quad \text{(G-4)}$$

$$\text{and} V(2n+1) = (1/K) \cdot V(2n+1) \text{for} i_0 \le 2n+1 < i_i \quad \text{(G-5)}$$

The 1D_FILTR_WS procedure then performs the following N_{LS} lifting steps (for $s = N_{LS} - 1, N_S - 2, \ldots, 1, 0$):

$$V(2n + m_s) = V(2n + m_s) - \left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot \left(V\left(PSE_O(2n + m_s - (2k+1))\right) + V\left(PSE_O(2n + m_s + (2k+1))\right)\right)\right) \quad \text{(G-6)}$$

where $m_{N_{LS-1}} = m_{init}$ and $m_s = 1 - m_{s+1}$ indicates whether the s^{th} lifting step updates even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), where L_s is the number of lifting coefficients for lifting step s, and where the range of n is defined by $i_0 \le 2n + m_s < i_1$.

The values of $V(k)$ such that $i_0 \le k < i_1$ form the output $W(k)$ of the 1D_FILTR_WS procedure:

$$W(k) = v(k) \tag{G-7}$$

G.3 Whole-sample symmetric (WS) wavelet transformation decomposition (informative)

The one-dimensional sub-band decomposition procedure 1D_SD_WS is implemented as a sequence of primitive lifting steps, which alternately modify odd-indexed samples with a weighted sum of even-indexed samples and even-indexed samples with a weighted sum of odd-indexed samples.

G.3.1 The 1D_SD_WS procedure (informative)

As illustrated in Figure G.5, the 1D_SD_WS procedure takes as input a one-dimensional array, X, the index i_0 of the first sample in array X, and the index i_1 of the sample following the last sample in array X. They produce as output an array, Y, of interleaved lowpass and highpass coefficients, with the same indices (i_0, i_1).

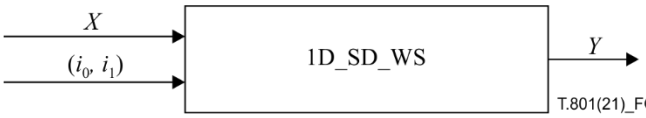

Figure G.5 – Parameters of the 1D_SD_WS procedure

For signals of length one (i.e., $i_0 = i_1 - 1$), the 1D_SD_WS procedure sets the value of $Y(i_0)$ to $Y(i_0) = X(i_0)$ if i_0 is an even integer, and to $Y(i_0) = 2X(i_0)$ if i_0 is an odd integer.

For signals of length greater than or equal to two (i.e., $i_0 < i_1 - 1$), as illustrated in Figure G.6, the 1D_SD_WS procedure applies the 1D_FILTD_WS procedure to X to produce the decomposed signal, Y.

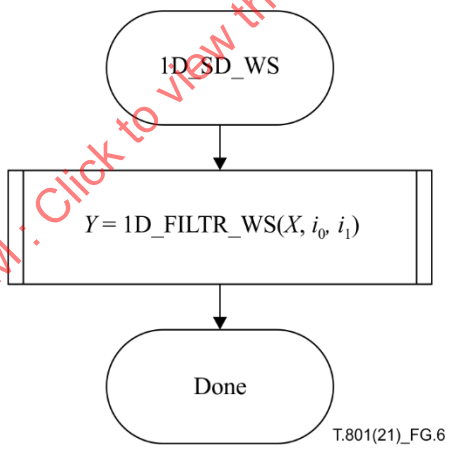

Figure G.6 – The 1D_SD_WS procedure

NOTE – Unlike in Rec. ITU-T T.800 | ISO/IEC 15444-1, the 1D_SD_WS procedure does not extend the signal prior to applying the 1D_FILTD_WS procedure. Instead, a procedure equivalent to the extension procedure is included in the 1D_FILTD_WS procedure.

G.3.2 The 1D_FILTD_WS one-dimensional decomposition procedure (informative)

Two versions of the decomposition procedures are specified, depending on whether the transformation is reversible or not (*WT_Typ* = *REV* or *IRR*). In the following two subclauses, the function $PSE_O(i)$ equals the function $PSE_O(i, i_0, i_1)$ specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation F-4.

G.3.2.1 The 1D_FILTD_WS reversible one-dimensional decomposition procedure (informative)

As shown in Figure G.7, the input parameters to the reversible version of procedure 1D_FILTD_WS are V, i_0, i_1, N_{LS}, and $\alpha_{s,k}$, β_s, ε_s, L_s for for $s = 0, 1, 2, \ldots, N_{LS} - 1$ and $k = 0, 1, 2, \ldots, L_s - 1$.

Figure G.7 – Parameters of the 1D_FILTD_WS procedure

The reversible 1D_FILTD_WS procedure consists in the following N_{LS} lifting steps (for $s = 0, 1, 2, \ldots, N_{LS} - 1$, starting with $s = 0$):

$$V(2n + m_s) = V(2n + m_s) + \left\lfloor \frac{\left(\sum_{k=0}^{l_s-1} \alpha_{s,k} \cdot (V(PSE_O(2n+m_s-(2k+1))) + V(PSE_O(2n+m_s+(2k+1))))\right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor \quad \text{(G-8)}$$

where $m_{N_{LS-1}} = m_{init}$ and $m_s = 1 - m_{s-1}$ indicates whether the sth lifting step updates even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), where L_s is the number of lifting coefficients for lifting step s, and where the range of n is defined by $i_0 \le 2n + m_s < i_1$.

The values $W(k) = V(k)$ form the output $W(k)$ of the 1D_FILTD_WS procedure. The output values are the values in the range $i_0 \le k < i_1$.

G.3.2.2 The irreversible one-dimensional decomposition procedure (1D_FILTD_WS) (informative)

As shown in Figure G.8, the input parameters to the irreversible version of procedure 1D_FILTD_WS are V, i_0, i_1, K, N_{LS}, and $\alpha_{s,k}$, L_s for for $s = 0, 1, 2, \ldots, N_{LS} - 1$ and $k = 0, 1, 2, \ldots, L_s - 1$.

Figure G.8 – Parameters of the 1D_FILTD_WS procedure

The 1D_FILTD_WS procedure performs the following N_{LS} lifting steps (for $s = 0, 1, \ldots, N_{LS} - 1$, starting with $s = 0$):

$$V(2n + m_s) = V(2n + m_s) + \left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot (V(PSE_O(2n + m_s - (2k + 1))) + V(PSE_O(2n + m_s + (2k + 1))))\right) \quad \text{(G-9)}$$

where $m_{N_{LS-1}} = m_{init}$ and $m_s = 1 - m_{s-1}$ indicates whether the s^{th} lifting step updates even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), where L_s is the number of lifting coefficients for lifting step s, and where the range of n is defined by $i_0 \le 2n + m_s < i_1$.

The 1D_FILTD_WS procedure ends with two scaling steps:

$$V(2n) = (1/K) \cdot V(2n) \text{for} i_0 \le 2n < i_i \quad \text{(G-10)}$$

$$\text{and} V(2n + 1) = K \cdot V(2n + 1) \text{for} i_0 \le 2n + 1 < i_1 \quad \text{(G-11)}$$

The values $W(k) = V(k)$ form the output $W(k)$ of the 1D_FILTD_WS procedure. The output values are the values in the range $i_0 \le k < i_1$.

G.4 Examples of WS wavelet transformations (informative)

Examples of wavelet transformations are specified in terms of their signalled values, as listed in Table G.1. Parameters that occur in sequences (e.g., L_s, $s = 0, \ldots, N_{LS} - 1$) are enumerated in order of increasing index.

Both examples are mathematically equivalent to transformations already defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 and are provided solely for didactic purposes.

G.4.1 Reversible WS wavelet transformations (*WT_Typ = REV*) (informative)

Typical values for the β_s parameter are $2^{\varepsilon_s - 1}$.

G.4.1.1 Reversible 5-3 wavelet transformation (informative)

The transformation specified by the parameter values found in Table G.2 is mathematically equivalent to the default reversible transformation specified in Rec. ITU-T T.800 | ISO/IEC 15444-1.

Table G.2 – Parameters of the 5-3 reversible wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	2
L_s	1, 1
$\alpha_{s,k}$	$\alpha_{0,0} = -1$ $\alpha_{1,0} = 1$
ε_s	1, 2
β_s	1, 2

The first decomposition step *(s = 0)* is specified with negative lifting coefficients and an additive residue of 1. This apparent difference with the Rec. ITU-T T.800 | ISO/IEC 15444-1 definition is necessitated by the fact that decomposition updates are systematically *added* to the input vector by the procedures of this annex whereas the first reversible decomposition lifting step in the definition of Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.4.8.1, *subtracts* its update. The corresponding reconstruction lifting steps also differ in the sign of their updates. The transformation specified by the above parameter values is, however, mathematically equivalent to the Rec. ITU-T T.800 | ISO/IEC 15444-1 definition of the reversible 5-3 wavelet transformation.

G.4.1.2 Reversible 13-7 wavelet transformation (informative)

Table G.3 shows the parameters of the 13-7 reversible wavelet transformation.

Table G.3 – Parameters of the 13-7 reversible wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	2
L_s	2, 2
$\alpha_{s,k}$	$\alpha_{0,k} = -9, 1$ $\alpha_{1,k} = 5, -1$
ε_s	4, 4
β_s	8, 8

G.4.2 Irreversible WS wavelet transformations (*WT_Typ = IRR*) (informative)

G.4.2.1 Irreversible 5-3 wavelet transformation (informative)

Table G.4 shows the parameters for the irreversible version of the Rec. ITU-T T.800 | ISO/IEC 15444-1 reversible 5-3 wavelet transformation specified in clause G.4.1.1:

Table G.4 – Parameters of the 5-3 irreversible wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	2
L_s	1, 1
$\alpha_{s,k}$	$\alpha_{0,0} = -\frac{1}{2}$ $\alpha_{1,0} = \frac{1}{4}$
K	1

G.4.2.2 Irreversible 7-5 wavelet transformation (informative)

Table G.5 gives the parameters of the 7-5 irreversible wavelet transformation.

Table G.5 – Parameters of the irreversible 7-5 wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	3
L_s	1, 1, 1
$\alpha_{s,k}$	$\alpha_{0,0} = \frac{2}{25}$ $\alpha_{1,0} = -\frac{175}{406}$ $\alpha_{2,0} = \frac{609}{2500}$
K	$\frac{116}{100}$

G.4.2.3 Irreversible 9-7 wavelet transformation (informative)

This is the default irreversible wavelet transformation specified in Rec. ITU-T T.800 | ISO/IEC 15444-1; the transformation specified by the following parameter values is mathematically equivalent to the Rec. ITU-T T.800 | ISO/IEC 15444-1 transformation. Exact expressions for the values given by decimal approximations in Table G.6 can be found in Annex F of Rec. ITU-T T.800 | ISO/IEC 15444-1.

Table G.6 – Parameters of the irreversible 9-7 wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	4
L_s	1, 1, 1, 1
$\alpha_{s,k}$	$\alpha_{0,0}$ = –1.586 134 342 059 924 $\alpha_{1,0}$ = –0.052 980 118 572 961 $\alpha_{2,0}$ = 0.882 911 075 530 934 $\alpha_{3,0}$ = 0.443 506 852 043 971
K	1.230 174 104 914 001

Annex H

Transformation of images using arbitrary wavelet transformations

(This annex forms an integral part of this Recommendation | International Standard.)

This Recommendation | International Standard defines a transformation of tile components. In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

This annex specifies an extension of the one-dimensional sub-band reconstruction procedure 1D_SR specified in Rec. ITU-T T.800 | ISO/IEC 15444-1. The internal structure of this extension differs from the internal structure of procedure 1D_SR in Rec. ITU-T T.800 | ISO/IEC 15444-1 because of the greater generality of this annex. The procedure in this annex, 1D_SR_ARB, is specified as a sequence of lifting steps, each of which involves a signal extension operation and an update operation. A scaling operation is performed in the case of irreversible wavelet transformations. Syntax is specified whereby an implementation may use wavelet transformations signalled in the codestream by the encoder.

H.1 Wavelet transformation parameters and normalizations

Table G.1 lists filter parameters for wavelet transformations that are signalled in the codestream. Signalling for these parameters is included in the ATK marker segment (see clause A.3.5). Table H.1 lists additional parameters contained in the ATK marker segment that are used exclusively in this annex.

Table H.1 – Additional parameters for arbitrary wavelet transformations

Parameter tag	Meaning	Value(s)
Exten	Boundary extension method used in lifting steps	constant (*CON*) whole-sample symmetric (*WS*)
off_s	Offset for lifting step *s*	8-bit signed integer: defined for $0 \le s < N_{LS}$

H.1.1 Normalization of ARB wavelet transformations

The procedures specified in this clause apply only in the case of ATK marker segment for which *Filt_Cat* = *ARB*. The parameter D_s defined in clause G.2.1 represents the sum of the lifting coefficients for step *s*, $0 \le s < N_{LS}$ (normalized in the case of reversible transformations). This parameter is redefined in this annex as follows:

$$D_s = \sum_{k=0}^{L_s-1} \alpha_{s,k} \text{ if } WT\_Typ = IRR$$

$$D_s = \frac{1}{2^{\varepsilon_s}} \sum_{k=0}^{L_s-1} \text{ if } WT\_Typ = REV$$

The normalization requirements in clause G.2.1 remain as specified**,** using the above redefinition of D_s.

H.1.2 Compatibility of ARB and WS wavelet transformations

The transformations defined in this annex constitute an extension of the transformations defined in Annex G**.** The output produced by the procedures in Annex G using an ATK marker segment specifying *Filt_Cat* = *WS* is equal to the output produced by the procedures in this annex if the ATK marker segment is interpreted by setting the extension option to *Exten* = *WS* and modifying ("unfolding") the filter parameters for each lifting step as follows, in the order specified:

1) Duplicate the sequence of lifting coefficients: define $\alpha_{s,k+L_s} = \alpha_{s,k}$ for $k = 0, \ldots, L_s - 1$.
2) Reverse the first half of the extended sequence of lifting coefficients: set $\alpha_{s,k} = \alpha_{s,2L_s-1-k}$ for $k = 0, \ldots, L_s - 1$.
3) Define the value of the offset parameter: $off_s = m_s - L_s$.
4) Redefine the value of the parameter indicating the number of lifting coefficients: $L_s = 2L_s$.

H.2 Arbitrary (ARB) wavelet transformation reconstruction procedures

The procedures specified in this clause apply only in the case of ATK marker segment for which *Filt_Cat* = *ARB*. The extended one-dimensional sub-band reconstruction filtering (1D_SR_ARB) procedure defined in this annex is specified

as a sequence of lifting steps, which alternately update the odd subsequence with a weighted sum of even-indexed samples and update the even subsequence with a weighted sum of odd-indexed samples. The option of reversible or irreversible transformation is signalled by the *WT_Typ* parameter.

H.2.1 The extended 1D_SR_ARB procedure

As illustrated in Figure H.1, the extended 1D_SR_ARB procedure takes as input a one-dimensional array, *Y*, of interleaved lowpass and highpass coefficients, the index i_0 of the first sample in array *Y*, and the index i_1 of the sample following the last sample in array *Y*. It produces as output a reconstructed array, *X*, with the same indices (i_0, i_1).

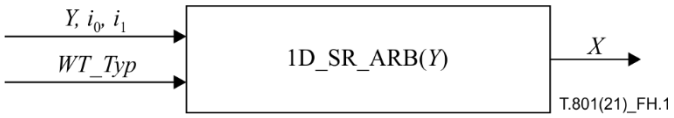

Figure H.1 – Parameters of the extended 1D_SR_ARB procedure

For signals of length one (i.e., $i_0 = i_1 - 1$), the 1D_SR_ARB procedure sets the value of $X(i_0)$ to $X(i_0) = Y(i_0)$ if i_0 is an even integer, and to $X(i_0) = Y(i_0)/2$ if i_0 is an odd integer.

For signals of length greater than or equal to two (i.e., $i_0 < i_1 - 1$), as illustrated in Figure H.2, the 1D_SR_ARB procedure applies a scaling step in the case of irreversible transformations (*WT_Typ* = *IRR*) and then applies a sequence of lifting steps, defined by the parameters from Tables G.1 and H.1, to produce the reconstructed signal, *X*. The variable *s*, which indexes the lifting steps, decreases from $N_{LS} - 1$ to zero in the reconstruction process.

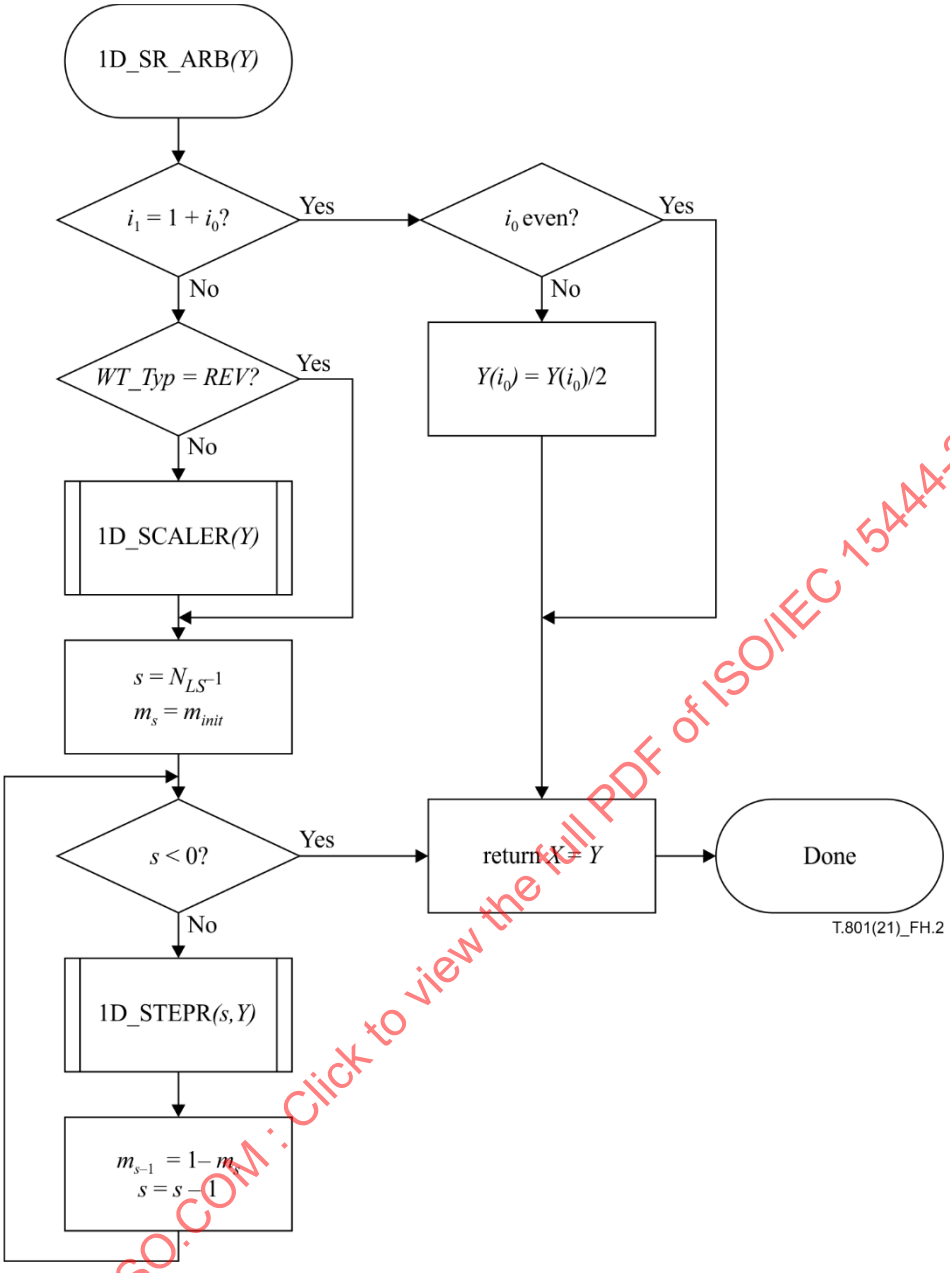

Figure H.2 – Extended procedure 1D_SR_ARB

H.2.2 The 1D_SCALER procedure

As shown in Figure H.3, procedure 1D_SCALER applies a scaling procedure to interleaved input vector *V* using signalled parameter *K* from Table G.1 and produces an updated version of vector *V* with the same indices (i_0, i_1). This procedure is used only in irreversible transformations.

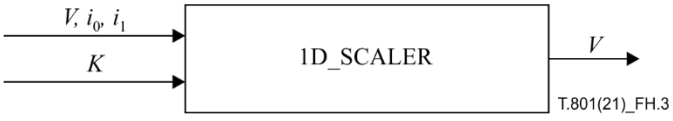

Figure H.3 – Parameters of the 1D_SCALER procedure

The 1D_SCALER procedure performs the following scaling operations:

$$V(2n) = K \cdot V(2n) \text{for} i_0 \leq 2n < i_1 \quad \text{(H-1)}$$

$$\text{and} V(2n+1) = (1/K) \cdot V(2n+1) \text{for} i_0 \leq 2n+1 < i_1 \quad \text{(H-2)}$$

H.2.3 The 1D_STEPR procedure

As shown in Figure H.4, procedure 1D_STEPR applies one reconstruction lifting step to interleaved input vector *V* and produces an updated version of vector *V* with the same indices (i_0, i_1).

Figure H.4 – Parameters of the 1D_STEPR procedure

Procedure 1D_STEPR applies an extension procedure, determined by the *Exten* parameter, to the input, *V*. This is followed by a reconstruction update filtering procedure, determined by the *WT_Typ* parameter, as seen in Figure H.5. Only one of the two subsequences of *V* (the even- or the odd-indexed subsequence) is updated on each pass through 1D_STEPR.

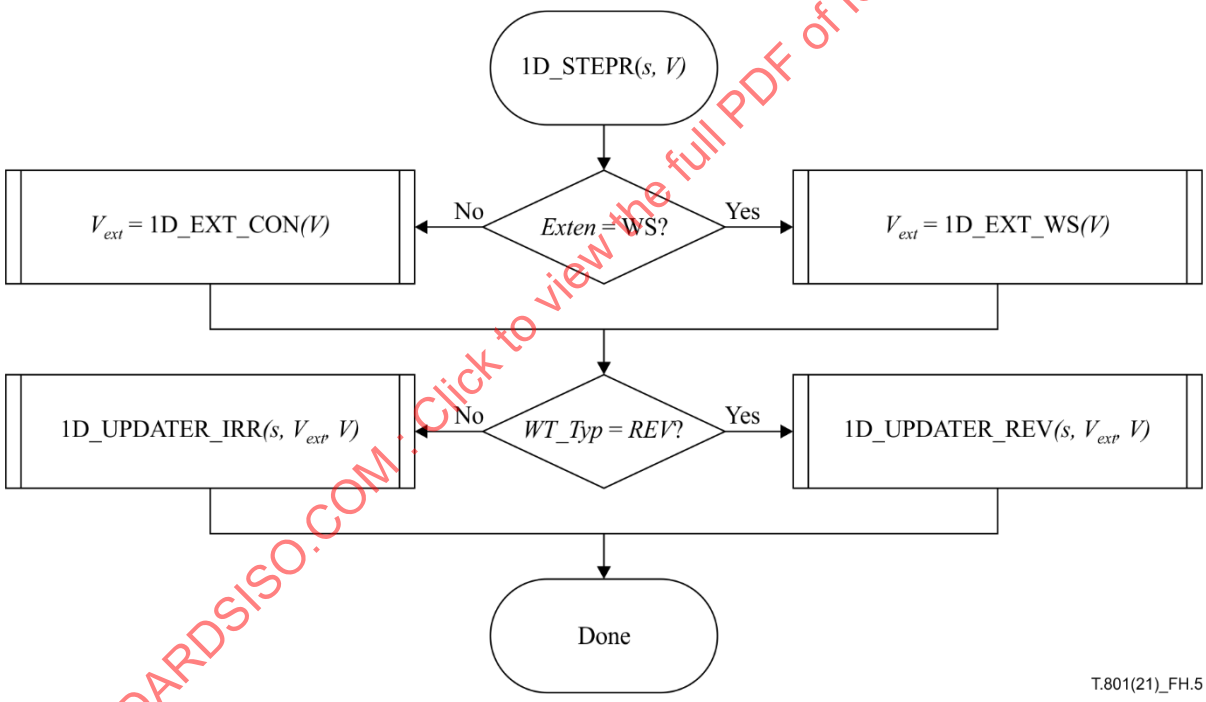

Figure H.5 – Procedure 1D_STEPR

H.2.4 Extension procedures

The exact manner in which extended samples are accessed in a realization of this Recommendation | International Standard (e.g., by copying extended arrays, by buffering, by indirect addressing, or by some other strategy) is implementation-dependent. This clause is normative only insofar as it defines mathematical extensions of the input vector of sufficient length to enable the 1D_UPDATER_REV and 1D_UPDATER_IRR procedures to perform their update filtering operations as specified.

H.2.4.1 Minimum extension lengths

Although procedures 1D_EXT_WS and 1D_EXT_CON in principle define arbitrarily long extensions of the input vector, *V*, the minimum number of extended samples required to perform a given lifting step, *s*, can be calculated in terms of the wavelet transformation parameters for that step. The minimum extension lengths, i_{left} and i_{right}, for lifting step *s* are defined to be the smallest non-negative integers such that the interval $[i_0 - i_{left}, i_1 - 1 + i_{right}]$ contains all indexes addressed by the update filtering procedures 1D_UPDATER_REV*(s)* and 1D_UPDATER_IRR*(s)*. Minimum extension lengths for lifting step *s* are given in Tables H.2 and H.3 as functions of the parity of i_0 and i_1, the update characteristic, m_s, the number of lifting coefficients, L_s, and the offset, off_s. A minimum extension length is zero whenever an expression in either of these two tables evaluates to a negative number for some particular set of parameter values.

Table H.2 – Minimum left extension length

i_{left}:	$m_s = 0$	$m_s = 1$
i_0 even	$-1 - 2off_s$	$-2off_s$
i_0 odd	$-2 - 2off_s$	$1 - 2off_s$

Table H.3 – Minimum right extension length

i_{right}:	$m_s = 0$	$m_s = 1$
i_1 even	$2(L_s - 1 + off_s)$	$-1 + 2(L_s - 1 + off_s)$
i_1 odd	$1 + 2(L_s - 1 + off_s)$	$-2 + 2(L_s - 1 + off_s)$

H.2.4.2 1D_EXT_WS procedure

As shown in Figure H.6, the 1D_EXT_WS procedure accepts as input a vector *V* supported on an interval (i_0, i_1) and outputs a vector V_{ext} supported on a larger interval containing values of *i* beyond the range $i_0 \le i < i_1$. Except for the minimum extension lengths, i_{left} and i_{right} (specified above in clause H.2.4.1), this procedure is identical to the 1D_EXTR procedure defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.3.7. This procedure does not modify its input vector, *V*.

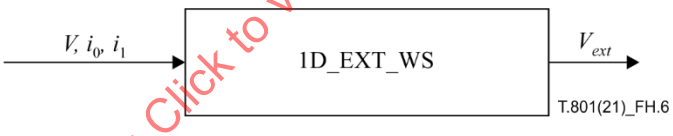

Figure H.6 – Parameters of the 1D_EXT_WS procedure

H.2.4.3 1D_EXT_CON procedure

As shown in Figure H.7, the 1D_EXT_CON procedure accepts as input a vector *V* supported on an interval (i_0, i_1) and outputs a vector V_{ext} supported on a larger interval containing values of *i* beyond the range $i_0 \le i < i_1$. The minimum extension lengths required for an extension created by procedure 1D_EXT_CON are specified above in clause H.2.4.1. This procedure does not modify its input vector, *V*.

Figure H.7 – Parameters of the 1D_EXT_CON procedure

Procedure 1D_EXT_CON defines constant extensions of the even- and odd-indexed subsequences in *V* according to the following rules.

For $i_0 \le i < i_1$:

$$V_{ext}(i) = V(i)$$

For $k \geq 1$:

$$V_{ext}(i_0 - 2k) = V(i_0)$$

$$V_{ext}(i_0 + 1 - 2k) = V(i_0 + 1)$$

$$V_{ext}(i_1 - 1 + 2k) = V(i_1 - 1)$$

$$V_{ext}(i_1 - 2 + 2k) = V(i_1 - 2)$$

H.2.5 One-dimensional reconstruction update filtering procedures

Two reconstruction update filtering procedures are defined, one for reversible transformations (1D_UPDATER_REV) and one for irreversible transformations (1D_UPDATER_IRR). Reconstruction steps are defined recursively as even-subsequence updates ($m_s = 0$) or odd-subsequence updates ($m_s = 1$), beginning with step number $N_{LS} - 1$, whose update characteristic is signalled in Table G.1 by the m_{init} parameter, and recursing downward: $m_{s-1} = 1 - m_s$. Both procedures take as input an interleaved input vector V and produce as output an updated version of vector V with the same indices (i_0, i_1).

H.2.5.1 Reversible one-dimensional reconstruction update (1D_UPDATER_REV) procedure

Procedure 1D_UPDATER_REV modifies either the even- or the odd-indexed subsequence in vector V by a weighted sum of samples from the extended sequence, V_{ext}, after applying a rounding operation to the weighted sum. Figure H.8 shows the input parameters to procedure 1D_UPDATER_REV.

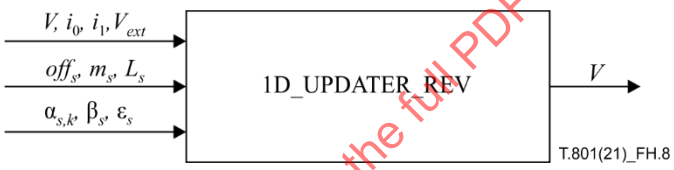

Figure H.8 – Parameters of the 1D_UPDATER_REV procedure

The 1D_UPDATER_REV procedure performs the following update filtering operation:

$$V(2n + m_s) = V(2n + m_s) - \left\lfloor \frac{\left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot V_{ext}(2n+1-m_s+2(k+off_s))\right)+\beta_s}{2^{\varepsilon_s}} \right\rfloor \quad \text{(H-3)}$$

for all n such that $i_0 \leq 2n + m_s < i_1$.

H.2.5.2 Irreversible one-dimensional reconstruction update (1D_UPDATER_IRR) procedure

Procedure 1D_UPDATER_IRR modifies either the even- or the odd-indexed subsequence in vector V by a weighted sum of samples from the extended sequence, V_{ext}. Figure H.9 shows the input parameters to procedure 1D_UPDATER_IRR.

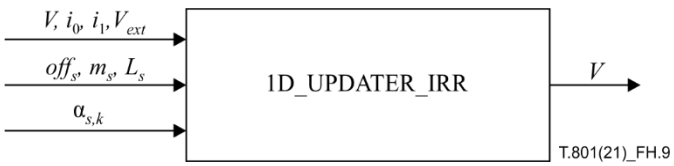

Figure H.9 – Parameters of the 1D_UPDATER_IRR procedure

The 1D_UPDATER_IRR procedure performs the following update filtering operation:

$$V(2n + m_s) = V(2n + m_s) - \sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot V_{ext}(2n + 1 - m_s + 2(k + off_s)) \quad \text{(H-4)}$$

for all n such that $i_0 \leq 2n + m_s < i_1$.

H.3 Arbitrary (ARB) wavelet transformation decomposition procedures (informative)

The extended one-dimensional sub-band decomposition filtering (1D_SD_ARB) procedure is implemented as a sequence of lifting steps, which alternately update the odd subsequence with a weighted sum of even-indexed samples and update the even subsequence with a weighted sum of odd-indexed samples.

H.3.1 Extended 1D_SD_ARB procedure (informative)

As illustrated in Figure H.10, the extended 1D_SD_ARB procedure takes as input a one-dimensional array, X, of data, the index i_0 of the first sample in array X, and the index i_1 of the sample following the last sample in array X. It produces as output an array, Y, of interleaved sub-band samples, with the same indices (i_0, i_1).

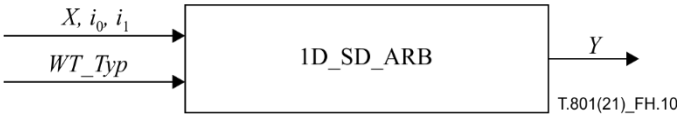

Figure H.10 – Parameters of the extended 1D_SD_ARB procedure

For signals of length one (i.e., $i_0 = i_1 - 1$), the 1D_SD_ARB procedure sets the value of $Y(i_0)$ to $Y(i_0) = X(i_0)$ if i_0 is an even integer, and to $Y(i_0) = 2X(i_0)$ if i_0 is an odd integer.

For signals of length greater than or equal to two (i.e., $i_0 < i_1 - 1$), as illustrated in Figure H.11, the 1D_SD_ARB procedure applies a sequence of lifting steps, defined by the parameters from Table H.1, and then applies a scaling step in the case of irreversible transformations *(WT_Typ = IRR)* to produce the decomposed signal, Y. The variable s, which indexes the lifting steps, increases from zero to $N_{LS} - 1$ in the decomposition process.

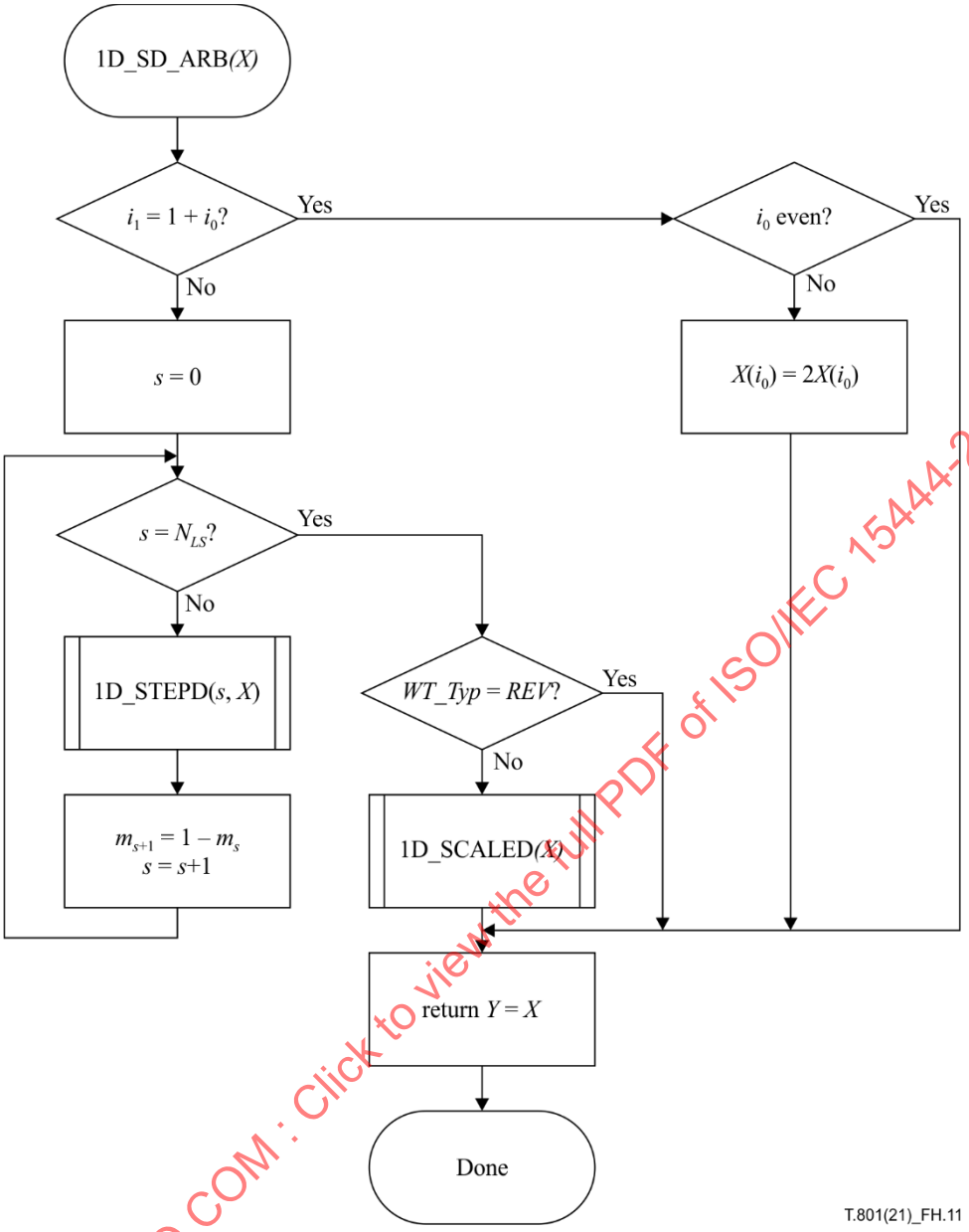

Figure H.11 – Extended procedure 1D_SD_ARB

H.3.2 The 1D_STEPD procedure (informative)

As shown in Figure H.12, procedure 1D_STEPD applies one decomposition lifting step to input vector *V* and produces an updated version of vector *V* with the same indices (i_0, i_1).

Figure H.12 – Parameters of the 1D_STEPD procedure

Procedure 1D_STEPD applies an extension procedure, determined by the *Exten* parameter, to the input, *V*. This is followed by a decomposition update filtering procedure, determined by the *WT_Typ* parameter, as seen in Figure H.13. Only one of the two subsequences of *V* (the even- or the odd-indexed subsequence) is updated on each pass through 1D_STEPD.

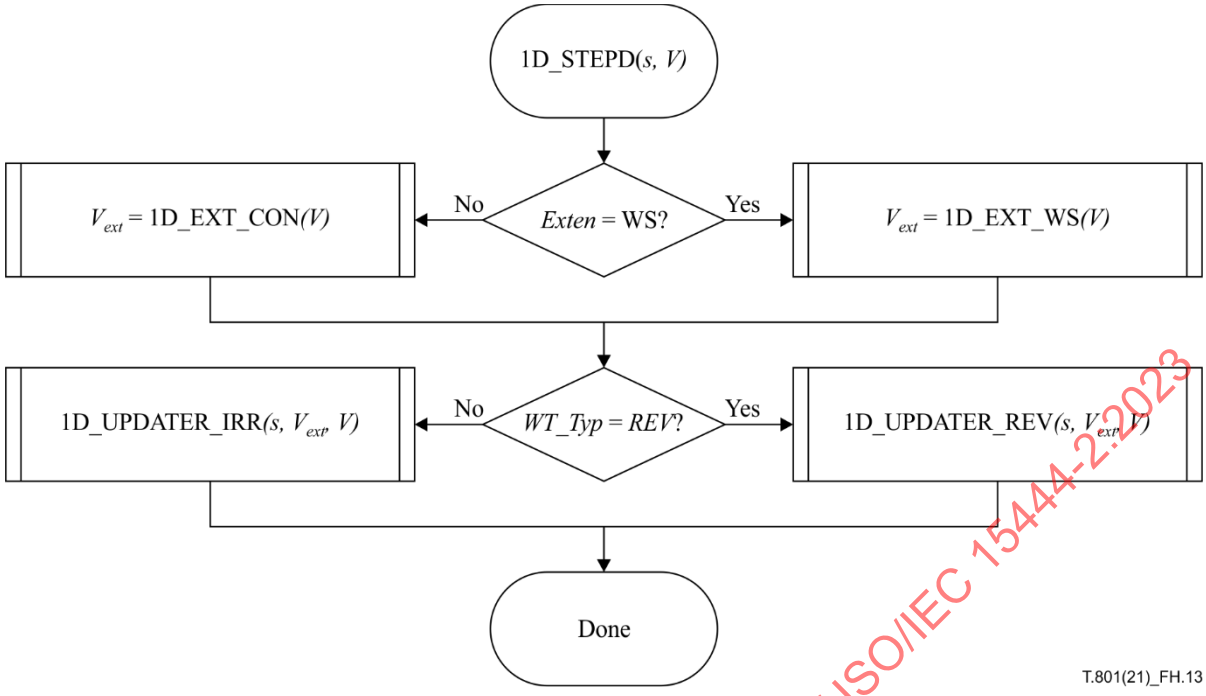

Figure H.13 – Procedure 1D_STEPD

H.3.3 Extension procedures (informative)

The extension procedures 1D_EXT_CON and 1D_EXT_WS shown in Figure H.13 are identical to the procedures specified in clause H.2.4, including the specifications of minimum extension lengths.

H.3.4 One-dimensional decomposition update procedures (informative)

Two decomposition update filtering procedures are defined, one for reversible transformations (1D_UPDATED_REV) and one for irreversible transformations (1D_UPDATED_IRR). Decomposition steps are defined recursively as even-subsequence updates ($m_s = 0$) or odd-subsequence updates ($m_s = 1$). Both procedures take as input a vector V and produce as output an updated version of vector V with the same indices (i_0, i_1).

H.3.4.1 Reversible one-dimensional decomposition update (1D_UPDATED_REV) procedure (informative)

Procedure 1D_UPDATED_REV modifies either the even- or the odd-indexed subsequence in vector V by a weighted sum of samples from the extended sequence, V_{ext}, after applying a rounding operation to the weighted sum. Figure H.14 shows the input parameters to procedure 1D_UPDATED_REV.

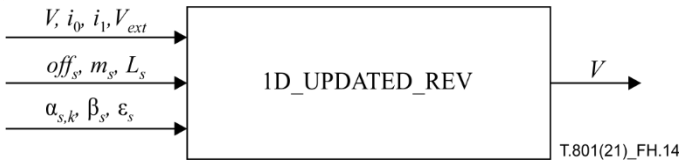

Figure H.14 – Parameters of the 1D_UPDATED_REV procedure

The 1D_UPDATED_REV procedure performs the following update filtering operation:

$$V(2n+m_s) = V(2n+m_s) + \left\lfloor \frac{\left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot V_{ext}(2n+1-m_s+2(k+off_s))\right)+\beta_s}{2^{\varepsilon_s}} \right\rfloor \quad \text{(H-5)}$$

for all n such that $i_0 \le 2n + m_s < i_1$.

H.3.4.2 Irreversible one-dimensional decomposition update (1D_UPDATED_IRR) procedure (informative)

Procedure 1D_UPDATED_IRR modifies either the even- or the odd-indexed subsequence in vector *V* by a weighted sum of samples from the extended sequence, V_{ext}. Figure H.15 shows the input parameters to procedure 1D_UPDATED_IRR.

Figure H.15 – Parameters of the 1D_UPDATED_IRR procedure

The 1D_UPDATED_IRR procedure performs the following update filtering operation:

$$V(2n + m_s) = V(2n + m_s) + \sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot V_{ext}(2n + 1 - m_s + 2(k + off_s)) \quad \text{(H-6)}$$

for all *n* such that $i_0 \le 2n + m_s < i_1$.

H.3.5 1D_SCALED procedure (informative)

As shown in Figure H.16, procedure 1D_SCALED applies a scaling procedure to interleaved input vector *V* using signalled parameter *K* from Table H.1 and produces an updated version of vector *V* with the same indices (i_0, i_1). This procedure is used only in irreversible transformations.

Figure H.16 – Parameters of the 1D_SCALED procedure

The 1D_SCALED procedure performs the following scaling operations:

$$V(2n) = \left(\frac{1}{K}\right) \cdot V(2n) \text{ for } i_0 \le 2n < i_1 \quad \text{(H-7)}$$

$$\text{and } V(2n + 1) = K \cdot V(2n + 1) \text{ for } i_0 \le 2n + 1 < i_1 \quad \text{(H-8)}$$

H.4 Examples of ARB wavelet transformations (informative)

Examples of optional wavelet transformations are specified in terms of their signalled parameters, as listed in Tables G.1 and H.1. Parameters that occur in sequences (e.g., L_s, $s = 0, \ldots, N_{LS} - 1$) are enumerated in order of increasing index.

H.4.1 Examples of arbitrary wavelet transformations (Filt_Cat = ARB) (informative)

All of the example wavelet transformations presented in this clause are based on linear phase wavelet transformations of the type known as "half-sample symmetric". The equivalent convolutional filters have even-length impulse responses, with symmetric lowpass impulse responses and antisymmetric highpass impulse responses.

H.4.1.1 Reversible ARB wavelet transformations (WT_Typ = REV) (informative)

H.4.1.1.1 The reversible Haar 2-2 wavelet transformation (informative)

Tables H.4 to H.6 provide information on reversible ARB wavelet transformations Haar 2-2, 2-6 and 2-10.

Table H.4 – Parameters of the reversible Haar 2-2 wavelet transformation

Parameter	Value(s)
m_{init}	0
N_{LS}	2
L_s	1, 1
$\alpha_{s,k}$	$\alpha_{0,0} = -1$ $\alpha_{1,0} = 1$
ε_s	0, 1
β_s	0, 1
Exten	CON
off_s	0, 0

H.4.1.1.2 Reversible 2-6 wavelet transformation (informative)

Table H.5 – Parameters of the reversible 2-6 wavelet transformation

Parameter	Value(s)
m_{init}	1
N_{LS}	3
L_s	1, 1, 3
$\alpha_{s,k}$	$\alpha_{0,0} = -1$ $\alpha_{1,0} = 1$ $\alpha_{2,k} = 1, 0, -1$
ε_s	0, 1, 2
β_s	0, 1, 2
Exten	CON
off_s	0, 0, –1

H.4.1.1.3 Reversible 2-10 wavelet transformation (informative)

Table H.6 – Parameters of the reversible 2-10 wavelet transformation

Parameter	Value(s)
m_{init}	1
N_{LS}	3
L_s	1, 1, 5
$\alpha_{s,k}$	$\alpha_{0,0} = -1$ $\alpha_{1,0} = 1$ $\alpha_{2,k} = -3, 22, 0, -22, 3$
ε_s	0, 1, 6
β_s	0, 1, 32
Exten	CON
off_s	0, 0, –2

H.4.1.2 Irreversible ARB wavelet transformations (WT_Typ=IRR) (informative)

H.4.1.2.1 Irreversible 6-10 wavelet transformation (informative)

Tables H.7 and H.8 provide information on irreversible ARB wavelet transformations 6-10 and 10-18.

Table H.7 – Parameters of the irreversible 6-10 wavelet transformation

Parameter	Value(s)
m_{init}	1
N_{LS}	7
L_s	1, 1, 2, 1, 2, 1, 3
$\alpha_{s,k}$	$\alpha_{0,0}$ = –1 $\alpha_{1,0}$ = 1,586 134 342 06 $\alpha_{2,k}$ = –0,460 348 209 828, 0,460 348 209 828 $\alpha_{3,0}$ = 0,25 $\alpha_{4,k}$ = 0,374 213 867 768, –0,374 213 867 768 $\alpha_{5,k}$ = –1,336 134 342 06 $\alpha_{6,k}$ = 0,293 067 171 03, 0, –0,293 067 171 03
K	1
Exten	WS
off_s	0, 0, 0, –1, 0, 0, –1

H.4.1.2.2 Irreversible 10-18 wavelet transformation (informative)

Table H.8 – Parameters of the irreversible 10-18 wavelet transformation

Parameter	Value(s)
m_{init}	1
N_{LS}	11
L_s	1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 5
$\alpha_{s,k}$	$\alpha_{0,0}$ = –1 $\alpha_{1,0}$ = 0,997 150 691 05 $\alpha_{2,k}$ = –1,005 731 278 27, 1,005 731 278 27 $\alpha_{3,0}$ = –0,270 403 576 31 $\alpha_{4,k}$ = 2,205 099 723 43, –2,205 099 723 43 $\alpha_{5,0}$ = 0,080 599 957 36 $\alpha_{6,k}$ = –1,626 825 323 50, 1,626 825 323 50 $\alpha_{7,0}$ = 0,520 403 576 31 $\alpha_{8,k}$ = 0,604 046 642 50, –0,604 046 642 50 $\alpha_{9,0}$ = –0,827 750 648 41 $\alpha_{10,k}$ = –0,066 158 129 64, 0,294 021 377 20 0, –0,294 021 377 20 , 0,066 158 129 64
K	1
Exten	WS
off_s	0, 0, 0, –1, 0, 0, 0, –1, 0, 0, –2

H.4.2 Example of a structure for lifting implementation of half-sample symmetric wavelet transformations (informative)

The lifting steps given in Figure H.17 guarantee that the filters implemented are half-sample symmetric. The parameters $\alpha_{s,k}$ used in Figure H.17 are different from those used in all other sections. The function $R(x)$ is either the identity $R(x) = x$ (for irreversible transformations) or $R(x) = \left\lfloor \frac{x+\beta_s}{2^{\varepsilon_s}} \right\rfloor$ for reversible transformations.

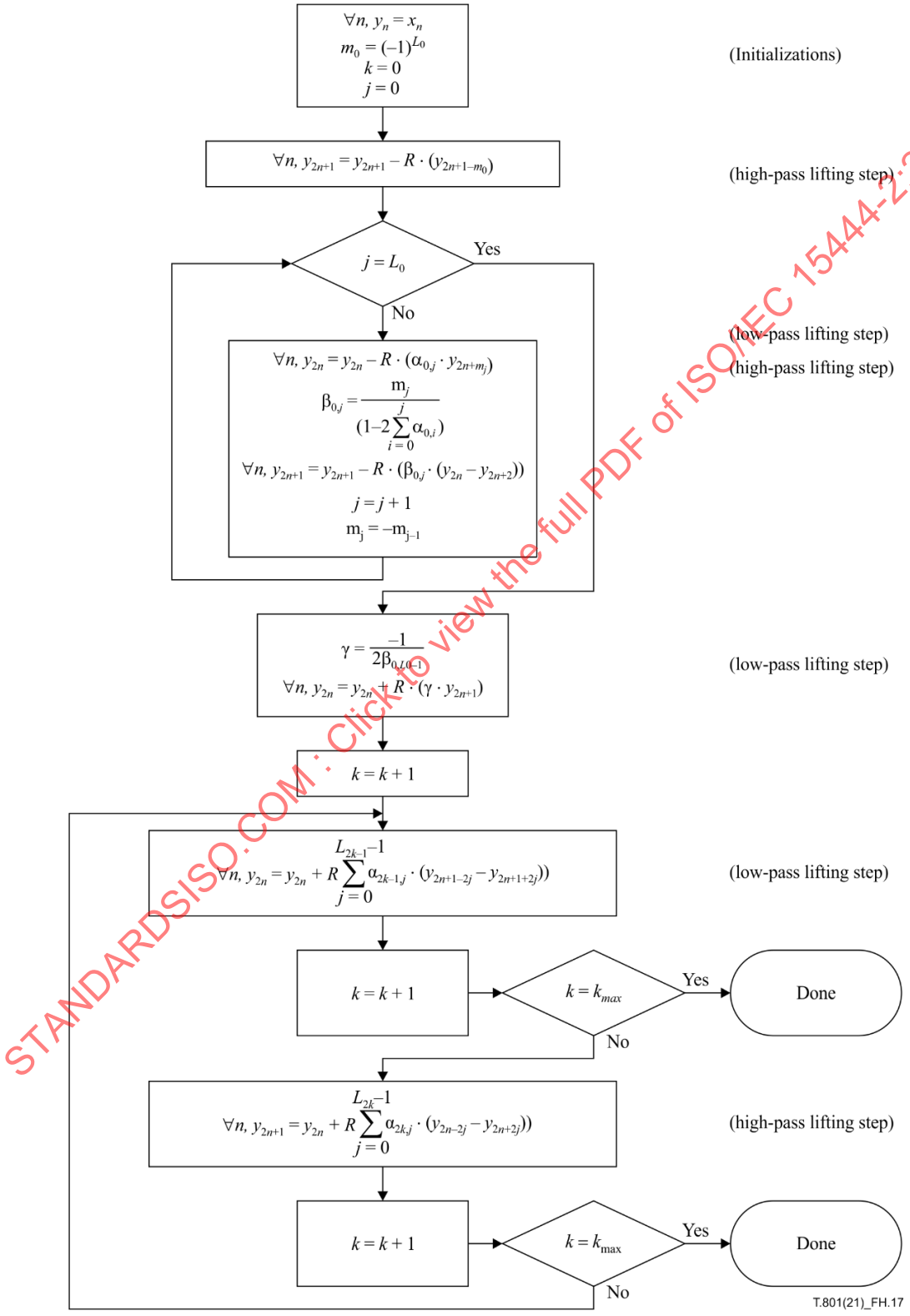

Figure H.17 – Lifting implementation for forward half-sample symmetric wavelet transformations

Annex I

Single sample overlap discrete wavelet transform, code-block anchor point and progression order extensions

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 as well as an extension to Annex G that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1) and by the CAP marker segment (see clause A.3.13).

I.1 Introduction to single sample overlapping

This annex specifies four possible extensions.

The code-block anchor point (CBAP) extensions concern the partitioning of sub-bands into code-blocks, and enable memory-efficient implementations of the DWT, as well as memory-efficient geometric manipulations of compressed image data (90°, 180° and 270° rotations, and mirror operations).

The position-resolution level-component-layer progression is an extended progression order that is specified in I.2. This progression order can be used with or without CBAP extensions.

The single sample overlap (SSO) extension (see I.3.1 and I.3.2) concerns the independent application of the discrete wavelet transformation to blocks of samples which overlap by one row and one column, which enables a low-memory block-based implementation of the discrete wavelet transformations, both forward and inverse.

The tile single sample overlap (TSSO) extension (see I.3.1 and I.3.2) concerns the partitioning of images into image tiles which overlap by one row and one column of samples.

I.2 The code-block anchor points (CBAP) extension

The parameters z_x and z_y are signalled in the Scod marker parameter (see clause A.2.3). If they are both equal to zero, then no modification need be made to Rec. ITU-T T.800 | ISO/IEC 15444-1. If either of z_x and z_y is equal to 1, then the following modifications to annexes of Rec. ITU-T T.800 | ISO/IEC 15444-1 need to be made.

I.2.1 Division of resolution levels in precincts

This clause replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause B.6.

Consider a particular tile-component and resolution level whose bounding sample coordinates in the reduced resolution image domain are (trx_0, try_0) and (trx_1–1, try_1–1), as already described. Figure I.1 shows the partitioning of this tile-component resolution level into precincts. The precinct is anchored at location (z_x, z_y), so that the upper left hand corner of any given precinct in the partition is located at ($z_x + m \cdot 2^{PPx}$, $z_y + n \cdot 2^{PPy}$) where m and n are integers, and PPx and PPy are signalled in the COD or COC marker segments (see Rec. ITU-T T.800 | ISO/IEC 15444-1, clauses A.6.1 and A.6.2). PPx and PPy may be different for each tile-component and resolution level. PPx and PPy shall be at least 1 for all resolution levels except $r = 0$ where they are allowed to be zero.

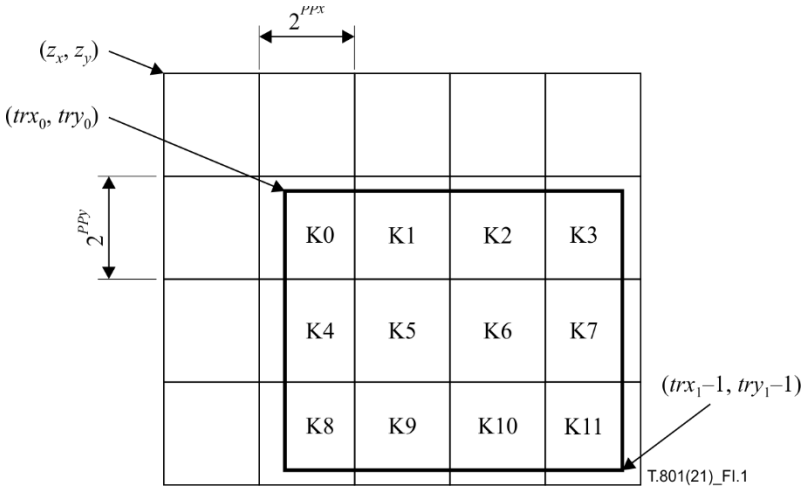

Figure I.1 – Precincts of one reduced resolution (modified Figure B.8 of Rec. ITU-T T.800 | ISO/IEC 15444-1)

The number of precincts which span the tile-component at resolution level, *r*, is given by:

$$numprecinctswide = \begin{cases} \left\lceil \frac{trx_1 - z_x}{2^{PPx}} \right\rceil - \left\lfloor \frac{trx_0 - z_x}{2^{PPx}} \right\rfloor, & trx_1 > trx_0 \\ 0, \text{otherwise} \end{cases} \quad \text{(I-1)}$$

$$numprecinctshigh = \begin{cases} \left\lceil \frac{try_1 - z_y}{2^{PPy}} \right\rceil - \left\lfloor \frac{try_0 - z_y}{2^{PPy}} \right\rfloor, & try_1 > try_0 \\ 0, \text{otherwise} \end{cases} \quad \text{(I-2)}$$

Even if Equations I-1 and I-2 indicate that both *numprecinctswide* and *numprecinctshigh* are nonzero, some, or all, precincts may still be empty as explained below. The precinct index runs from 0 to *numprecincts* – 1 where *numprecincts* = *numprecinctswide* · *numprecinctshigh* in raster order (see Figure I.1). This index is used in determining the order of appearance, in the codestream, of packets corresponding to each precinct, as explained in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause B.12.

The partition element associated with integers *m* and *n* has a corresponding precinct within the codestream, having upper-left and lower-right corners (px_0, py_0) and $(px_1 - 1, py_1 - 1)$given by the following equations, whenever $px_1 > px_0$ and $py_1 > py_0$:

$$px_0 = \max\{trx_0, z_x + m \cdot 2^{PPx}\}$$

$$py_0 = \max\{try_0, z_y + n \cdot 2^{PPy}\}$$

$$px_1 = \min\{trx_1, z_x + (m+1) \cdot 2^{PPx}\}$$

$$py_1 = \min\{try_1, z_y + (n+1) \cdot 2^{PPy}\}$$

where trx_0, try_0, trx_1 and try_1 are defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation B-14.

This precinct maps into the three sub-bands $(N_L - r + 1)HL$, $(N_L - r + 1)LH$ and $(N_L - r + 1)HH$ with upper-left and lower-right corners (pbx_0, pby_0) and $(pbx_1 - 1, pby_1 - 1)$ where:

$$pbx_0 = \left\lceil \frac{px_0 - xo_b}{2} \right\rceil + (1 - xo_b) z_x \quad \text{(I-3)}$$

$$pby_0 = \left\lceil \frac{py_0 - yo_b}{2} \right\rceil + (1 - yo_b) z_y \quad \text{(I-4)}$$

$$pbx_1 = \left\lceil \frac{px_1 - xo_b}{2} \right\rceil + (1 - xo_b) z_x \quad \text{(I-5)}$$

$$pby_1 = \left\lceil \frac{py1 - yo_b}{2} \right\rceil + (1 - yo_b) z_y \quad \text{(I-6)}$$

where xo_b and yo_b are given in Rec. ITU-T T.800 | ISO/IEC 15444-1, Table B.1.

It can happen that a precinct is empty, meaning that no sub-band coefficients from the relevant resolution level actually contribute to the precinct. This can occur, for example, at the lower right of a tile-component due to sampling with respect to the reference grid. When this happens, every packet corresponding to that precinct shall still appear in the codestream (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause B.9).

I.2.2 Division of the sub-bands into codeblocks

This clause modifies Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause B.7.

The following two sentences concerning the code-block partition should replace the ones present in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause B.7, following Equation B-19.

Like the precinct, the code-block partition is anchored at (z_x, z_y), as illustrated in Rec. ITU-T T.800 | ISO/IEC 15444-1, Figure I.2. Thus, all first rows of code-blocks in the code-block partition are located at $y = z_y + m \cdot 2^{ycb'}$ and all first columns of code-blocks are located at $x = z_x + n \cdot 2^{xcb'}$, where m and n are integers.

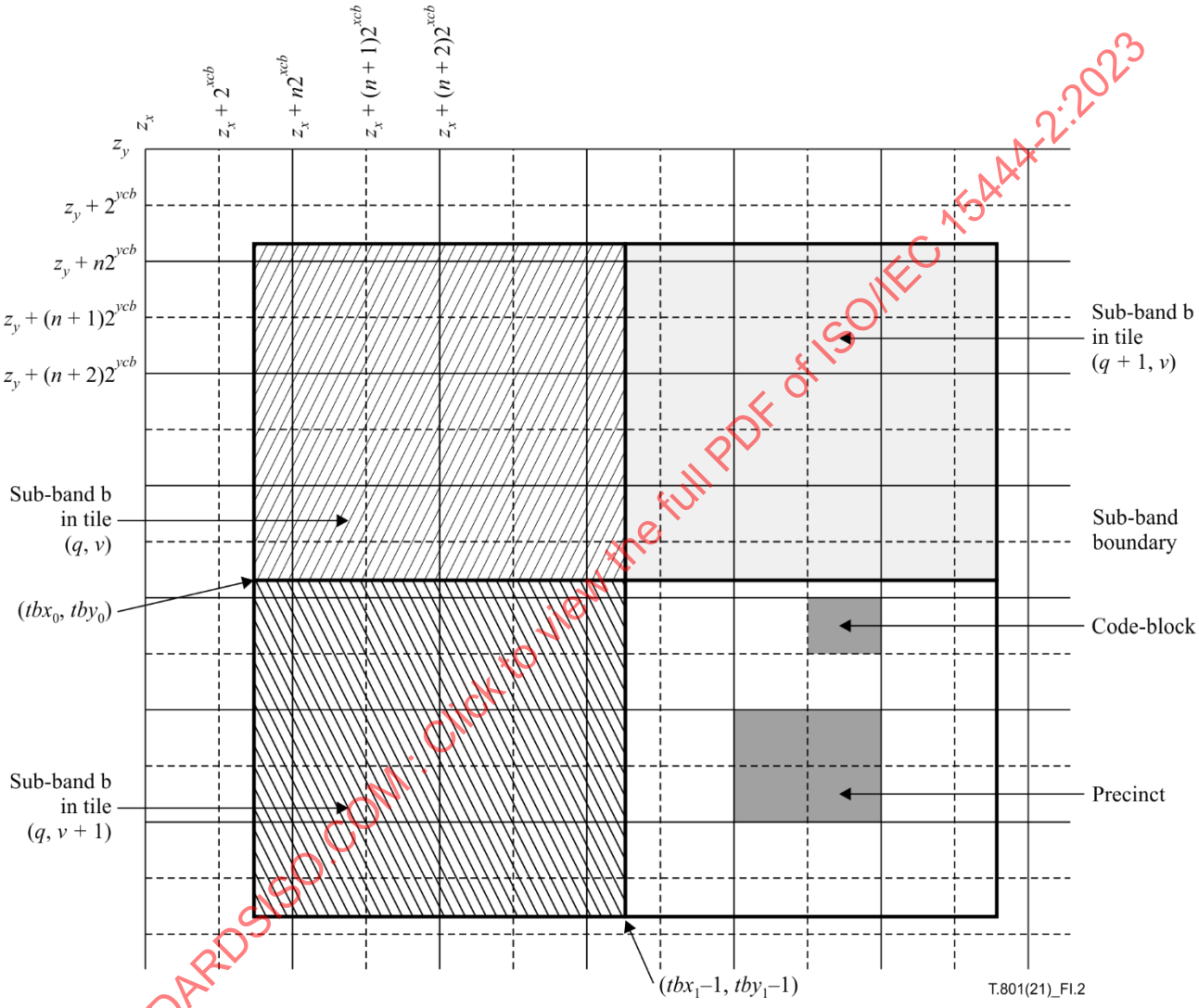

Figure I.2 – Codeblocks and precincts in sub-band *b* from four different tiles

I.2.3 Resolution level-position-component-layer progression

This clause replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, clause B.12.1.3.

Resolution level-position-component-layer progression is defined as the interleaving of the packets in the following order:

for each $r = 0, \ldots, N_{max}$

for each $y = ty_0, \ldots, ty_1-1$,

for each $x = tx_0, \ldots, tx_1-1$,

for each $i = 0, \ldots,$ Csiz–1

if $((y - z_y)$ divisible by $YRsiz(i) \cdot 2^{PPy(r,i)+N_L(i)-r})$ OR $((y = ty_0)$ AND $((try_0 - z_y) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPy(r,i)+N_L(i)-r}))$

if $((x - z_x)$ divisible by $XRsiz(i) \cdot 2^{PPx(r,i)+N_L(i)-r})$ OR $((x = tx_0)$ AND $((trx_0 - z_x) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPx(r,i)+N_L(i)-r}))$

for the next precinct, *k*, if one exists,

for each $l = 0, \ldots, L-1$

packet for component *i*, resolution level *r*, layer *l*, and precinct *k*.

In the above, *k* can be obtained from:

$$k = \left\lfloor \frac{\left\lfloor \frac{x - z_x}{XRsiz(i) \cdot 2^{N_L - 1}} \right\rfloor}{2^{PPx(r,i)}} \right\rfloor - \left\lfloor \frac{trx_0 - z_x}{2^{PPx(r,i)}} \right\rfloor + numprecinctswide(r,i) \cdot \left(\left\lfloor \frac{\left\lfloor \frac{y - z_y}{YRsiz(i) \cdot 2^{N_L - 1}} \right\rfloor}{2^{PPy(r,i)}} \right\rfloor - \left\lfloor \frac{try_0 - z_y}{2^{PPy(r,i)}} \right\rfloor \right) \quad \text{(I-7)}$$

I.2.4 Position-component-resolution level-layer progression

This clause replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, clause B.12.1.4.

Position-component-resolution level-layer progression is defined as the interleaving of the packets in the following order:

for each $y = ty_0, \ldots, ty_1-1$,

for each $x = tx_0, \ldots, tx_1-1$,

for each $i = 0, \ldots,$ Csiz–1

for each $r = 0, \ldots, N_L$ where N_L is the number of decomposition levels for component *i*,

if $((y - z_y)$ divisible by $YRsiz(i) \cdot 2^{PPy(r,i)+N_L(i)-r})$ OR $((y = ty_0)$ AND $((try_0 - z_y) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPy(r,i)+N_L(i)-r}))$

if $((x - z_x)$ divisible by $XRsiz(i) \cdot 2^{PPx(r,i)+N_L(i)-r})$ OR $((x = tx_0)$ AND $((trx_0 - z_x) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPx(r,i)+N_L(i)-r}))$

for the next precinct, *k*, if one exists, in the sequence shown in Figure I.1

for each $l = 0, \ldots, L-1$

packet for component *i*, resolution level *r*, layer *l*, and precinct *k*.

In the above, *k* can be obtained from Equation I-7. To use this progression, *XRsiz* and *YRsiz* values shall be powers of two for each component. A progression of this type might be useful in providing high sample accuracy for a particular spatial location in all components.

I.2.5 Component-position-resolution level-layer progression

This clause replaces Rec. ITU-T T.800 | ISO/IEC 15444-1, clause B.12.1.5.

Component-position-resolution level-layer progression is defined as the interleaving of the packets in the following order:

for each $i = 0, \ldots,$ Csiz–1

for each $y = ty_0, \ldots, ty_1-1$,

for each $x = tx_0, \ldots, tx_1-1$,

for each $r = 0, \ldots, N_L$ where N_L is the number of decomposition levels for component *i*,

if $((y - z_y)$ divisible by $YRsiz(i) \cdot 2^{PPy(r,i)+N_L(i)-r})$ OR $((y = ty_0)$ AND $((try_0 - z_y) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPy(r,i)+N_L(i)-r}))$

if $((x - z_x)$ divisible by $XRsiz(i) \cdot 2^{PPx(r,i)+N_L(i)-r})$ OR $((x = tx_0)$ AND $((trx_0 - z_x) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPx(r,i)+N_L(i)-r}))$

for the next precinct, *k*, if one exists, in the sequence shown in Figure I.1

for each $l = 0, \ldots, L-1$

packet for component *i*, resolution level *r*, layer *l*, and precinct *k*.

In the above, *k* can be obtained from Equation I-7. A progression of this type might be useful in providing high accuracy for a particular spatial location in a particular image component.

I.2.6 Position-resolution level-component-layer progression

This clause defines the position-resolution level-component-layer progression order that is enabled if

Position-resolution level-component-layer progression is defined as the interleaving of the packets in the following order:

for each $y = ty_0, \ldots, ty_1-1$,

for each $x = tx_0, \ldots, tx_1-1$,

for each $r = 0, \ldots, N_L$ where N_L is the number of decomposition levels for component *i*,

for each $i = 0, \ldots,$ Csiz–1

if $((y - z_y)$ divisible by $YRsiz(i) \cdot 2^{PPy(r,i)+N_L(i)-r})$ OR $((y = ty_0)$ AND $((try_0 - z_y) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPy(r,i)+N_L(i)-r}))$

if $((x - z_x)$ divisible by $XRsiz(i) \cdot 2^{PPx(r,i)+N_L(i)-r})$ OR $((x = tx_0)$ AND $((trx_0 - z_x) \cdot 2^{N_L(i)-r}$ NOT divisible by $2^{PPx(r,i)+N_L(i)-r}))$

for the next precinct, *k*, if one exists, in the sequence shown in Figure I.1

for each $l = 0, \ldots, L-1$

packet for component *i*, resolution level *r*, layer *l*, and precinct *k*.

In the above, *k* can be obtained from Equation I-7. To use this progression, *XRsiz* and *YRsiz* values shall be powers of two for each component.

I.3 The SSO extension

This clause applies only if the SSO extension is selected. The selection of the SSO extension is signalled in the extended COD and COC markers (see clause A.2.3) and is only applicable to WS wavelet transformations (i.e., *Filt_Cat = WS*). The parameters relevant to the SSO extension *XC, YC* are signalled in the COD and COC extended marker segment.

I.3.1 Single sample overlap inverse discrete wavelet transformation (SSO-IDWT)

The selection of the SSO extension requires a modification of the 1D_FILTR_WS filtering procedure described in clause G.2.2.2 (the 1D_FILTR_SSO procedure), as well as a modification of the IDWT, 2D_SR, HOR_SR, VER_SR and 1D_SR procedures described in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause F.3: the IDWT_SSO, 2D_SR_SSO, HOR_SR_SSO, VER_SR_SSO and 1D_SR_SSO procedures. These modifications are specified in this clause.

I.3.1.1 The IDWT_SSO procedure

The IDWT_SSO procedure (illustrated in Figure I.3) starts with the initialization of the variable *lev* (the current decomposition level) to N_L, of the variable XC_{N_L} to $XC/2^{N_L}$ and of the variable YC_{N_L} to $YC/2^{N_L}$, where *XC* and *YC* are given in the COD/COC marker, in the SSO offset portion. The 2D_SR_SSO procedure (described in clause I.3.1.2) is performed at every level *lev*, where the level *lev* decreases at each iteration, until iterations are performed. The 2D_SR_SSO procedure is iterated over the *lev*LL, *lev*LX or *lev*XL sub-band produced at each iteration. Finally, the sub-band $a_{0LL}(u_{0LL}, v_{0LL})$ is the output array $I(x, y)$.

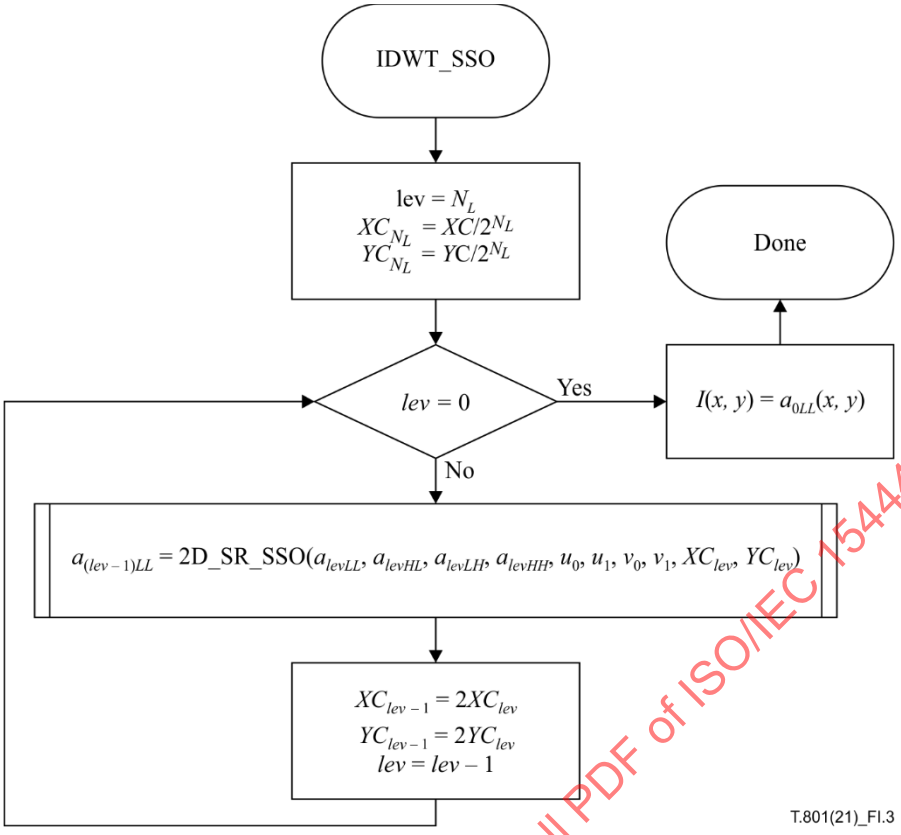

Figure I.3 – The IDWT_SSO Procedure

I.3.1.2 The 2D_SR_SSO Procedure

The 2D_SR_SSO procedure is identical to the 2D_SR procedure described in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.3.2, except for the addition of the parameters XC_{lev}, YC_{lev} (see Figure I.4), which are respectively used by the HOR_SR_SSO and VER_SR_SSO procedures (see I.3.1.3 and I.3.1.4). The 2D_SR_SSO procedure uses the 2D_INTERLEAVE procedure specified in Rec. ITU-T T.800 | ISO/IEC 15444-1.

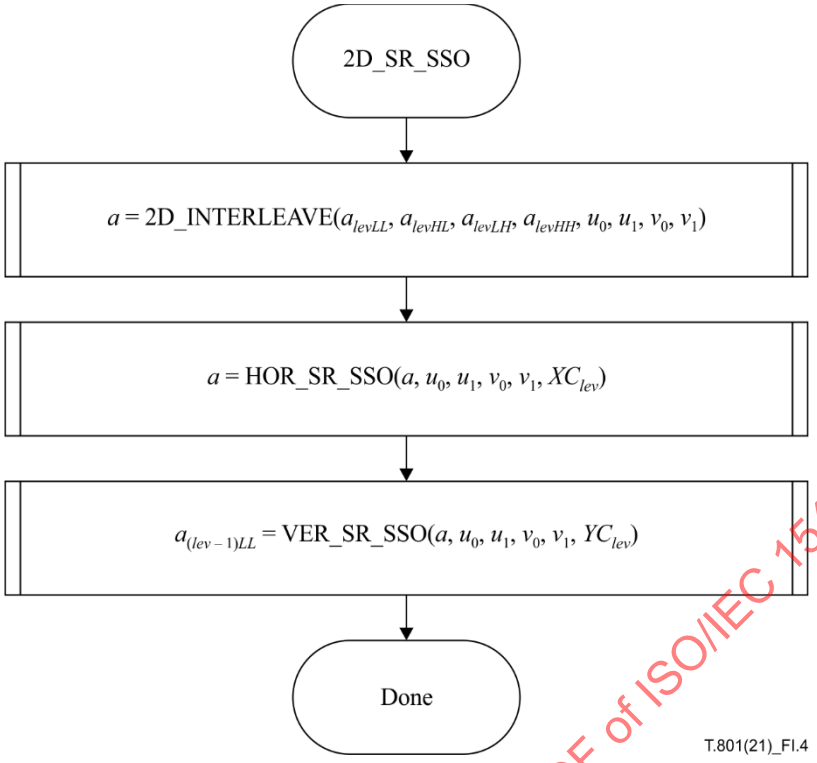

Figure I.4 – The 2D_SR_SSO procedure

I.3.1.3 The HOR_SR_SSO procedure

The HOR_SR_SSO procedure is identical to the HOR_SR procedure described in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.3.4, except for the addition of the parameter *xC*, which is used by the 1D_SR_SSO procedure (see I.3.1.5).

I.3.1.4 The VER_SR_SSO procedure

The VER_SR_SSO procedure is identical to the VER_SR procedure described in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.3.5, except for the addition of the parameter *yC*, which is used by the 1D_SR_SSO procedure (see I.3.1.5).

I.3.1.5 The 1D_SR_SSO procedure

The 1D_SR_SSO procedure is identical to the 1D_SR procedure described in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.3.6, except for the addition of the parameter *dC* (which is an input to the 1D_FILTR_SSO procedure) and the replacement of the 1D_FILTR procedure by the 1D_FILTR_SSO procedure (see I.3.1.6). The parameter *dC* is either the parameter *xC* (if called by the HOR_SR_SSO procedure) or the parameter *yC* (if called by the VER_SR_SSO procedure).

I.3.1.6 The 1D_FILTR_SSO procedure

The 1D_FILTR_SSO procedure is a modification of the 1D_FILTR_WS procedure described in clause G.2.2.2. The input and output parameters of the 1D_FILTR_SSO procedure are given in Figure I.5.

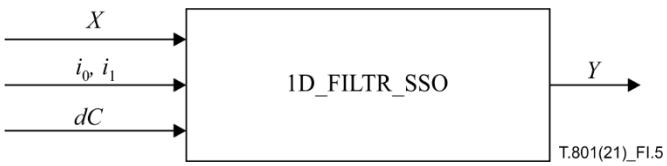

Figure I.5 – Parameters of the 1D_FILTR_SSO procedure

Let k_0 be defined by:

$$k_0 = \left\lfloor \frac{i_0}{dC} \right\rfloor \tag{I-8}$$

and N_I is defined by:

$$N_I = \left\lceil \frac{i_1 - 1}{dC} \right\rceil - \left\lfloor \frac{i_0}{dC} \right\rfloor \tag{I-9}$$

Subdivide the interval $[i_0, i_1 - 1]$ into the N_I intervals $I_p = [n_p, n_{p+1}]$ ($p = 0, 1, \ldots, N_I - 1$), where n_p is defined by:

$$n_0 = i_0, n_{N_I} = i_1 - 1 \text{ and } n_p = (k_0 + p)dC \text{ for } p = 1, \ldots, N_I - 1 \tag{I-10}$$

For an index $i \in I_p$, define the function $PSE_{O,p}(i)$ as:

$$PSE_{O,p}(i) = PSE_O(i, n_p, n_{p+1} + 1) \tag{I-11}$$

where the function $PSE_O(i, i_0, i_1)$ is defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation F-4.

I.3.1.6.1 Reversible transformations

This clause specifies for reversible transformations the modifications of each lifting step *s* as specified in Equation G-2. The modification of Equation G-2 ensures that each coefficient $V(2n + m_s)$ is calculated exclusively from coefficients the indices of which belong to the same interval I_p as $2n + m_s$. At each lifting step, all values $V(n_p)$ for $\{n_p | \text{mod}(n_p, dC) = 0; p = 0, 1, \ldots, N_I\}$ if any, remain unmodified, while all other values $V(2n + m_s)$ (i.e., for which $2n + m_s$ belongs to a single interval I_p) are modified according to Equation I-12:

$$V(2n + m_s) = V(2n + m_s) - \left\lfloor \frac{\left(\sum_{k=0}^{L_s - 1} \alpha_{s,k} \cdot (V(PSE_{O,p}(2n + m_s - (2k+1))) + V(PSE_{O,p}(2n + m_s + (2k+1))))\right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor \tag{I-12}$$

I.3.1.6.2 Irreversible transformations

This clause specifies for irreversible transformations the modifications of each lifting step *s* as specified in Equation G-6. The scaling steps specified in Equations G-4 and G-5 are not modified. The modification of Equation G-6 ensures that each coefficient $V(2n + m_s)$ is calculated exclusively from coefficients the indices of which belong to the same interval I_p as $2n + m_s$. At each lifting step, all values $V(n_p)$ for $\{n_p | \text{mod}(n_p, dC) = 0; p = 0, 1, \ldots, N_I\}$, if any, are modified according to Equation I-13:

$$V(n_p) = (1/B_s)V(n_p) \tag{I-13}$$

where B_s is defined in Equation G-1, while all other values $V(2n + m_s)$ (i.e., for which $2n + m_s$ belongs to a single interval I_p) are modified according to Equation I-14:

$$V(2n + m_s) = V(2n + m_s) - \left(\sum_{k=0}^{L_s - 1} \alpha_{s,k} \cdot (V(PSE_{O,p}(2n + m_s - (2k+1))) + V(PSE_{O,p}(2n + m_s + (2k+1))))\right) \tag{I-14}$$

I.3.2 Single sample overlap forward discrete wavelet transformation (informative)

The selection of the SSO extension requires a modification of the 1D_FILTD_WS filtering procedure described in clause G.3.2 (the 1D_FILTD_SSO procedure), as well as a modification of the FDWT, 2D_SD, HOR_SD, VER_SD and 1D_SD procedures specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause F.4 (the FDWT_SSO, 2D_SD_SSO, HOR_SD_SSO, VER_SD_SSO and 1D_SD_SSO). These modified procedures are specified in this clause.

I.3.2.1 The FDWT_SSO procedure

The FDWT_SSO procedure (illustrated in Figure I.6) starts with the initialization of the variable *lev* (the current decomposition level) to 1, of the variable XC_1 to *XC* and of the variable YC_1 to *YC*, where *XC* and *YC* are given in the COD/COC marker (see Table A.9). The 2D_SD_SSO procedure (described in clause I.3.1) is performed at every level *lev*, where the level *lev* increases at each iteration, until N_L iterations are performed.

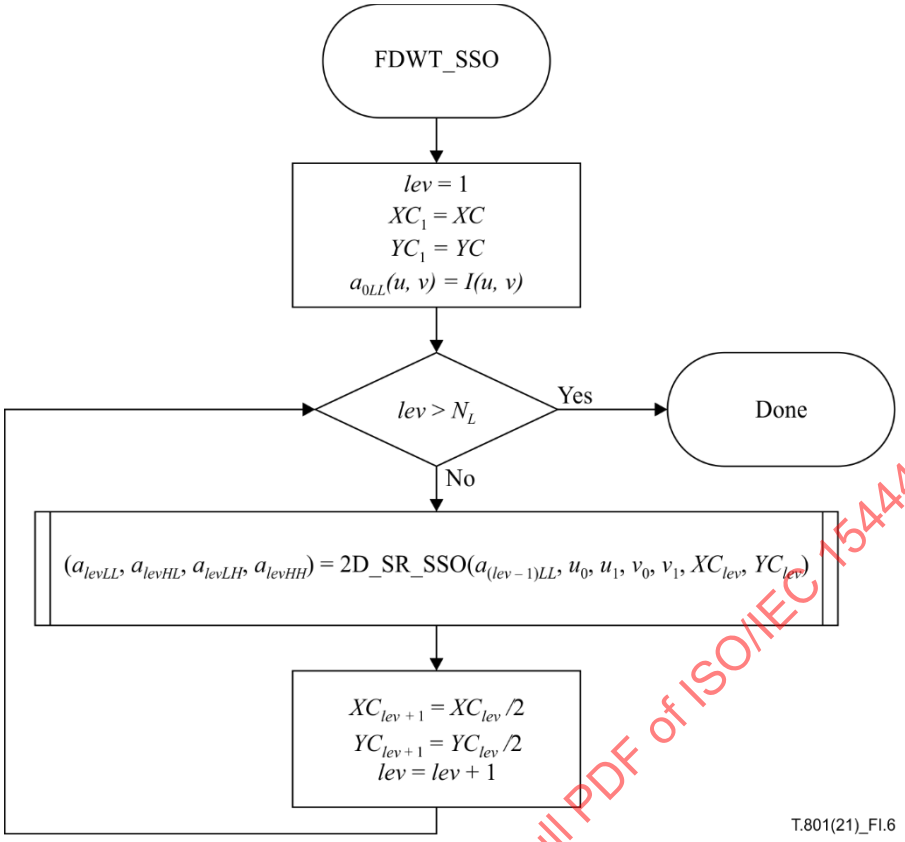

Figure I.6 – The FDWT_SSO procedure

I.3.2.2 The 2D_SD_SSO procedure

The 2D_SD_SSO procedure is identical to 2D_SD procedure specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.4.2, except for the addition of the parameters XC_{lev}, YC_{lev} (see Figure I.7), which are respectively used by the HOR_SD_SSO and VER_SD_SSO procedures (see I.3.2.3 and I.3.2.4).

I.3.2.3 The HOR_SD_SSO procedure

The HOR_SD_SSO procedure is identical to the HOR_SD procedure specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.4.4, except for the addition of the parameter *xC*, which is used by the 1D_SD_SSO procedure (see I.3.1).

I.3.2.4 The VER_SD_SSO procedure

The VER_SD_SSO procedure is identical to the VER_SD procedure specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.4.3, except for the addition of the parameter *yC*, which is used by the 1D_SD_SSO procedure (see I.3.1).

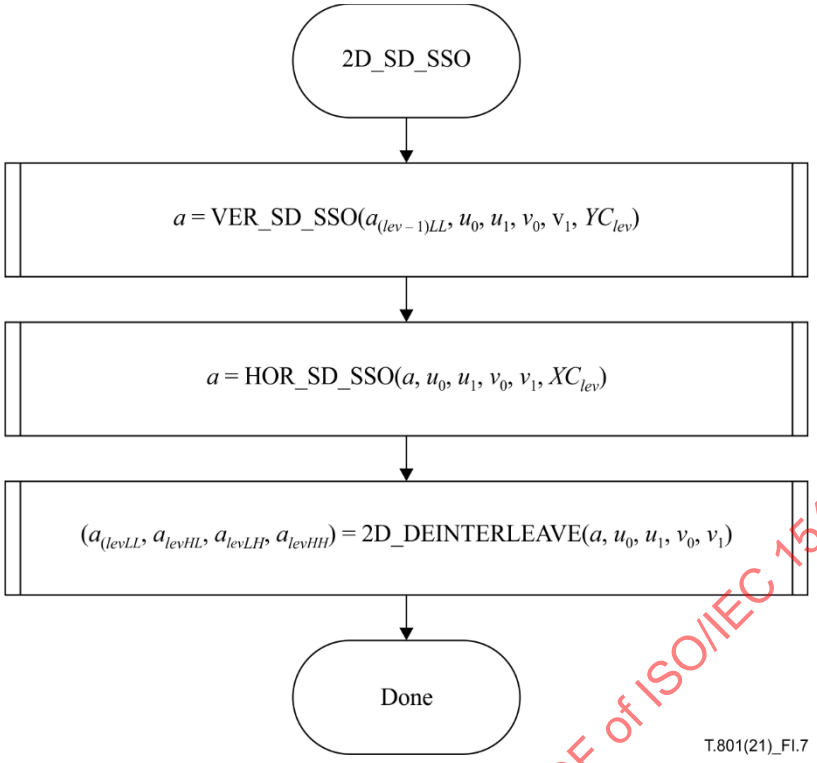

Figure I.7 – The 2D_SD_SSO procedure

I.3.2.5 The 1D_SD_SSO procedure

The 1D_SD_SSO procedure is identical to the 1D_SD procedure specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause F.4.6, except for the addition of the parameter *dC* (which is an input to the 1D_FILTD_SSO procedure) and the replacement of the 1D_FILTD_WS procedure by the 1D_FILTD_SSO procedure (see I.3.2.6). The parameter *dC* is either the parameter *xC* (if called by the HOR_SD_SSO procedure) or the parameter *yC* (if called by the VER_SD_SSO procedure).

I.3.2.6 The 1D_FILTD_SSO procedure

The 1D_FILTD_SSO procedure is a modification of the 1D_FILTD_WS procedure described in clause G.2.2.2. The input and output parameters of the 1D_FILTD_SSO procedure are given in Figure I.8.

I.3.2.6.1 Reversible transformations

This clause specifies for reversible transformations the modifications of each lifting step *s* as specified in Equation G-8. The modification of Equation G-8 ensures that each coefficient $V(2n + m_s)$ is calculated exclusively from coefficients the indices of which belong to the same interval I_p as $2n + m_s$. As a consequence, at each lifting step, all values $V(n_p)$ for $\{n_p | \text{mod}(n_p, dC) = 0; p = 0, 1, \ldots, N_I\}$, if any, remain unmodified, while all other values $V(2n + m_s)$ (i.e., for which $2n + m_s$ belongs to a unique interval I_p) are modified according to Equation I-15:

$$V(2n + m_s) = V(2n + m_s) + \left\lfloor \frac{\left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot (V(PSE_{O,p}(2n+m_s-(2k+1))) + V(PSE_{O,p}(2n+m_s+(2k+1))))\right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor \quad \text{(I-15)}$$

I.3.2.6.2 Irreversible transformations

This clause specifies for irreversible transformations the modifications of each lifting step *s* as specified in Equation G-9. The modification of Equation G-9 ensures that each coefficient $V(2n + m_s)$ is calculated exclusively from coefficients the indices of which belong to the same interval I_p as $2n + m_s$. At each lifting step, all values $V(n_p)$ for $\{n_p | \text{mod}(n_p, dC) = 0; p = 0, 1, \ldots, N_I\}$, if any, are modified according to Equation I-16:

$$V(n_p) = B_s V(n_p) \quad \text{(I-16)}$$

where B_s is defined in Equation G-1, while all other values $V(2n + m_s)$ (i.e., for which $2n + m_s$ belongs to a single interval I_p) are modified according to Equation I-17:

$$V(2n+m_s) = V(2n+m_s) + \left(\sum_{k=0}^{L_s-1} \alpha_{s,k} \cdot (V(PSE_{O,p}(2n+m_s-(2k+1))) + V(PSE_{O,p}(2n+m_s+(2k+1))))\right) \quad \text{(I-17)}$$

The scaling steps specified in Equations G-10 and G-11 are not modified.

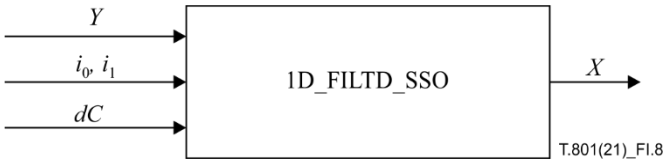

Figure I.8 – Parameters of the 1D_FILTD_SSO procedure

I.3.3 Selection of single sample overlap parameters (informative)

The selection of the SSO extension enables a low-memory block-based implementation of the wavelet transformations, both forward and inverse: for example, the forward transformation may be applied independently to SSO blocks of samples extracted from the image tile component. The parameters relevant to the selection of the SSO extension are: *XC*, *YC*, z_x and z_y (see I.2.2).

I.3.3.1 Division of image tile components into overlapping SSO blocks (informative)

SSO blocks are of width *XC* + 1 and height *YC* + 1 in the image tile component domain. The first and last row of a SSO block are always located at multiples of YC, while the first and last column of a SSO block are always located at multiples of XC (see Figure H.7). Two neighbouring SSO blocks overlap by either one row of samples (vertical neighbours), one column of samples (horizontal neighbours), or just one sample (diagonal neighbours).

I.3.3.2 Selection of tile parameters (informative)

To maximize coding efficiency, the following selection of tile parameters is recommended: mod(*XTsiz*, *XC*) = 0 and mod(*YTsiz*, *YC*) = 0.

To maximize memory efficiency, the following selection of encoding parameters is recommended: *XTOsiz* = z_x, *YTOsiz* = z_y.

I.3.4 SSO examples (informative)

I.3.4.1 Illustration in the case of the 5-3 forward reversible transformation (informative)

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) - \left\lfloor \frac{V_{ext}(2n)+V_{ext}(2n+2)}{2} \right\rfloor \text{ for } i_0 < 2n+1 < i_1 - 1 \quad \text{(I-18)}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \quad \text{(I-19)}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \quad \text{(I-20)}$$

The second lifting step is:

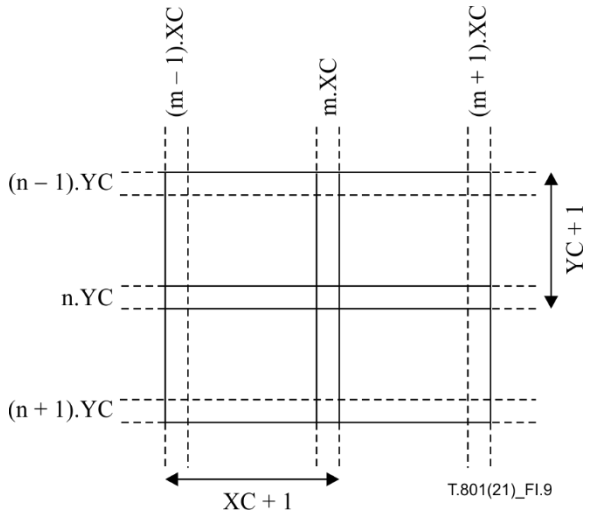

Figure I.9 – Position of SSO blocks

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n-1)+V_{ext}(2n+1)+2}{4} \right\rfloor \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-21)}$$

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n+1)+1}{2} \right\rfloor \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-22)}$$

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n-1)+1}{2} \right\rfloor \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-23)}$$

$$\text{and } V_{ext}(2n) = V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \quad \text{(I-24)}$$

I.3.4.2 Illustration in the case of the 5-3 forward irreversible transformation (informative)

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) - \left(\frac{V_{ext}(2n)+V_{ext}(2n+2)}{2} \right) \text{ for } i_0 < 2n+1 < i_1 - 1 \quad \text{(I-25)}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \quad \text{(I-26)}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \quad \text{(I-27)}$$

The second lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \left(\frac{V_{ext}(2n-1)+V_{ext}(2n+1)}{4} \right) \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-28)}$$

$$V_{ext}(2n) = V_{ext}(2n) + \frac{V_{ext}(2n+1)}{2} \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-29)}$$

$$V_{ext}(2n) = V_{ext}(2n) + \frac{V_{ext}(2n-1)}{2} \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-30)}$$

$$\text{and } \rightleftarrows V_{ext}(2n) = V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \quad \text{(I-31)}$$

I.3.4.3 Illustration in the case of the 9-7 forward irreversible transformation (informative)

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) + \alpha\left(V_{ext}(2n) + V_{ext}(2n+2)\right) \text{ for } i_0 < 2n+1 < i_1 - 1 \quad \text{(I-32)}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) + 2\alpha V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \quad \text{(I-33)}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) + 2\alpha V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \quad \text{(I-34)}$$

The second lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \beta\left(V_{ext}(2n-1) + V_{ext}(2n+1)\right) \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-35)}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\beta V_{ext}(2n+1) \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-36)}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\beta V_{ext}(2n+1) \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \quad \text{(I-37)}$$

$$\text{and } V_{ext}(2n) = (1 + 2\beta)V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \quad \text{(I-38)}$$

The third lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) + \gamma\big(V_{ext}(2n) + V_{ext}(2n+2)\big) \text{ for } i_0 < 2n+1 < i_1 - 1 \quad \text{(I-39)}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) + 2\gamma V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \quad \text{(I-40)}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) + 2\gamma V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \quad \text{(I-41)}$$

The fourth lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \delta(V_{ext}(2n-1) + \big(V_{ext}(2n+1)\big) \text{ for } i_0 < 2n < i_1 - 1 \text{and mod}(2n, dC) \neq 0 \quad \text{(I-42)}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\delta V_{ext}(2n+1) \text{ for } 2n = i_0 \text{ and mod}(2n, dC) \neq 0 \quad \text{(I-43)}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\delta V_{ext}(2n+1) \text{ for } 2n = i_1 - 1 \text{ and mod}(2n, dC) \neq 0 \quad \text{(I-44)}$$

$$\text{and } V_{ext}(2n) = \Big(1 + 2\beta(1+2\alpha) + 2\delta\big(1 + 2\gamma\big(1 + 2\beta(1+2\alpha)\big)\big)\Big) V_{ext}(2n) \text{ for mod}(2n, dC) = 0 \quad \text{(I-45)}$$

The scaling steps are the same for all coefficients.

I.4 The TSSO extension

This clause applies only if the TSSO extension is selected. The selection of the TSSO extension enables the use of tiles without any visible artifact at the boundary of tiles. The tiles shall overlap, but by one row and one column only.

I.4.1 Signalling for the TSSO

The selection of the TSSO extension is signalled in the extended COD and COC markers (see clause A.2.3). If the SSODWT is used with TSSO, then only wavelet transformations (reversible or irreversible) which use WS wavelet filters (i.e., *Filt_cat* = *WS*) may be used.

The parameters relevant to the selection of the TSSO extension *XTsiz*, *YTsiz* are signalled in the SIZ extended marker segments (see Rec. ITU-T T.800 | ISO/IEC 15444-1, clauses A.5.1), while the parameters *Hovlp* and *Vovlp* are signalled in the SSO parameter of the COD maker (see Table A.11 of A.2.3).

I.4.2 Partitioning of the image into single-sample overlapping tiles

The tile partitioning specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 is identical, except for these differences.

Equations B-7, B-8, B-9 and B-10 from Rec. ITU-T T.800 | ISO/IEC 15444-1 shall be modified as follows:

$$tx_0(p,q) = max(XTOsiz + p \cdot XTsiz - (1 - Hovlp), XOsiz) \quad \text{(I-46)}$$

$$ty_0(p,q) = max(YTOsiz + q \cdot YTsiz - (1 - Vovlp), YOsiz) \quad \text{(I-47)}$$

$$tx_1(p,q) = min(XTOsiz + (p+1) \cdot XTsiz + Hovlp, Xsiz) \quad \text{(I-48)}$$

$$ty_1(p,q) = min(YTOsiz + (q+1) \cdot YTsiz + Vovlp, Ysiz) \quad \text{(I-49)}$$

Tiles are of width $tx_1(p,q) - tx_0(p,q) = XTsiz + 1$ and height $ty_1(p\,q) - ty_0(p,q) = YTsiz + 1$. They overlap by one column and one row as shown in Figure I.10 in the case of *Hovlp* = *Vovlp* = 0. The parameters *Hovlp* and *Vovlp* may have a value of zero or one.

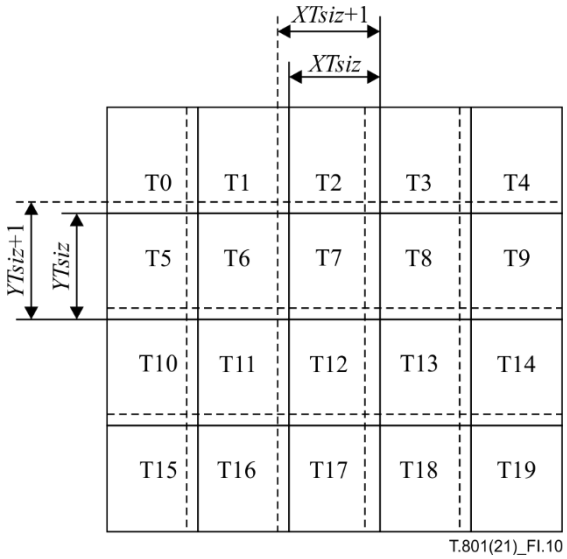

Figure I.10 – Tiling of the reference grid diagram

The tile parameters *XTsiz* and *YTsiz* shall satisfy the following equations:

$$\mathrm{mod}(XTsiz, R_x \cdot 2^{N_L}) = 0 \text{ and } \mathrm{mod}(YTsiz, R_y \cdot 2^{N_L}) = 0 \quad \text{(I-50)}$$

where R_x and R_y are the lowest common multiples of sub-sampling factor $XRsiz^i$ and $YRsiz^i$.

The tiling offsets *XTOsiz* and *YTOsiz* shall satisfy the following equation:

$$\mathrm{mod}(XTOsiz, R_x \cdot 2^{N_L}) = 1 - Hovlp \text{ and } \mathrm{mod}(YTOsiz, R_y \cdot 2^{N_L}) = 1 - Volvp. \quad \text{(I-51)}$$

Finally, the TSSO extension shall be used for all tile components.

I.4.3 Reconstruction of images samples from reconstructed tiles

Since the reconstructed tiles overlap by one row and one column with neighbouring tiles, some image samples will be reconstructed separately in two or four different tiles. For any such sample, one shall use the following rule.

If a sample is reconstructed from more than one tile, then the sample used for reconstruction is the sample value from the tile to the left if *Hovlp* = 0 and the sample value from the tile to the right if *Hovlp* = 1, the sample value from the tile to the top if *Vovlp* = 0 and the sample value from the tile to the bottom if *Hovlp* = 1.

I.5 Combining the SSO and TSSO extensions (informative)

It is possible to use the SSO extension in combination with the TSSO extension. When this occurs, the filtering procedures described in clause I.3 are applied to each overlapped tile separately. When *XTsiz* and *YTsiz* are multiples of *XC* and *YC* respectively, for example *XC* = *XTsiz* and *YC* = *YTsiz*, the values reconstructed at overlapping tile boundaries will not vary based upon the tile, so the rule described in clause I.4.3 is redundant. When *XTsiz* and *YTsiz* are not multiples of *XC* and *YC* respectively, then the rule for choosing reconstructed sample values in clause I.4.3 applies as described.

Furthermore, to improve memory efficiency, the following selection of encoding parameters is recommended: *XTOsiz* = z_x, *YTOsiz* = z_y.

Annex J

Multiple component transformations, extension

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard, except those found in Annex B. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1). The DC level shifting of tile-components described in Annex G of Rec. ITU-T T.800 | ISO/IEC 15444-1 is not performed when using any of the multiple component transformation procedures in this annex. Procedures exist in this annex that may be used in place of the DC level shifting described in Rec. ITU-T T.800 | ISO/IEC 15444-1 .

J.1 Introduction to multiple component transformation concepts

This annex specifies multiple component transformations. The most common multiple component transformation application is the compression of colour images. Standard colour images (RGB) are transformed into a colour space that is more conducive to spatial compression (i.e., YIQ). This technique can be extended for images that have more components; for example, LANDSAT images have seven components, six of which are highly correlated. It also can be used for the compression of CMYK images, multiple component medical images, and any other multiple component data.

There are two multiple component transformation techniques presented in this annex. The first is an array-based multiple component transformation which forms linear combinations of components to reduce the correlation of each component. This transformation structure permits component prediction transformations such as DPCM, and includes more complicated transformations such as the Karhunen-Loèvè Transformation (KLT). These array-based transformations can be implemented reversibly or irreversibly. The second multiple component transformation technique is a wavelet-based decorrelation transformation. The wavelet-based transformation may also be implemented reversibly or irreversibly. This annex provides a flexible mechanism to allow these techniques to be used in sequence if desired, e.g., an array-based transformation followed by a wavelet-based transformation. Furthermore, this annex provides mechanisms which allow components to be re-ordered and grouped into component collections.

Component collections may be formed to group together components with similar statistical properties to improve the compression efficiency of a multiple component transformation. Collections may also be used to reduce the computational complexity of component transformations by splitting a large component transformation involving many components into several transformations of smaller dimension. Frequently such splitting may be done with little loss in compression performance. Component collections also allow the application of array-based and wavelet-based transformations, on different collections, within the same compressed codestream.

It is possible using the techniques in this annex to generate more or fewer output (reconstructed image) components than the number of components encoded in the codestream. This allows encoders to transform original image components into a new domain and discard those transformed components containing little or no information prior to the creation of the compressed codestream. The encoder can nevertheless instruct a decoder on the proper way to regenerate an approximation to the original components given only the reduced set of codestream components. An encoder may also use the multiple component transformation processes to provide such functionality as generation of pseudo-colour or grayscale renditions of a multiple component image.

The techniques described in this annex are powerful and can serve many different uses. This annex does not prescribe how to apply these techniques to a multiple component image to increase compression efficiency; neither does it describe the many possible applications of these techniques. A detailed example is included in clause O.3 which attempts to illustrate some of their flexibility. Procedures are defined in this annex which strictly control the usage of the multiple component transformation techniques. These procedures ensure that any decoder conforming to this annex will successfully decode properly formed codestreams that use these techniques. As with any set of powerful tools, it is quite easy to make unintended errors, and great care should be exercised in adhering to the procedures of this annex in application of multiple component transformations.

J.2 Overview of inverse processing

A powerful characteristic of this multiple component transformation annex lies in its ability to accommodate multiple decorrelation techniques within the same framework and allow reconstruction using a generalized decoder. Reconstruction in this case includes inverse decorrelation transformation (e.g., KLT, etc.), inverse dependency

transformation (e.g., linear prediction, etc.) and inverse one-dimensional wavelet transformation. Figures J.1, J.3 and J.5 illustrate the inverse multiple component transformation processing steps needed to reconstruct image components from the codestream. The inverse multiple component transformation consists of a series of transformation stages. Within each stage, the set of available input components, called intermediate components, may be broken into component collections, each of which may be transformed with a different transformation method. The remainder of this clause provides details regarding the order of decoder actions and the locations of required information within the codestream. This clause does not address the equations governing application of a given transformation. Instead, individual multiple component transformations are referred to generically in this clause and are detailed in clause J.3.

Various marker segments convey the multiple component transformation information. For two of these marker segments, the MCT and the MCC (see clauses A.3.7 and A.3.8), it is possible that the amount of data required will be larger than the maximum amount of data allowed in a single marker segment. If multiple marker segments are needed to convey the data within an MCT or MCC marker segment, the data is split into a series of two or more marker segments. It is possible that a codestream may contain multiple series of marker segments in a main or first tile-part header. The marker segment index (Imct or Imcc) is repeated within each marker segment in the series. The data in a series of marker segments with the same marker segment index (Imct or Imcc) are grouped together. The entire series of marker segments shall be found in the same header, either the main header or the first tile-part header.

A field in each of the markers (Ymct or Ymcc) indicates the total number of marker segments that are used to convey the transformation information associated with that particular marker segment index (Imct or Imcc). A second field (Zmct or Zmcc) indicates the placement of a particular marker segment relative to all others in the same header sharing the marker index. When transformation information is distributed across more than one marker segment, the parameter lists from the marker segments are concatenated byte-wise in order of increasing Zmcc or Zmct. When such concatenation is completed, the resulting stream of parameters is then treated as if it had been transmitted in a single marker segment. The text that follows assumes that any such required concatenation of marker segment contents has been performed.

J.2.1 Inverse multiple component transformation (MCO_TRANSFORM)

As shown in Figure J.1, the inverse multiple component transformation is a transform that takes as its inputs the set of spatially reconstructed components from the codestream created by two-dimensional inverse wavelet transform and produces a set of reconstructed image components. Each multiple component sample is reconstructed by applying the processing steps as indicated in the codestream. The inverse transformation process is carried out in a series of steps known as transformation stages. The MCO marker segment (see Annex A.3.9) applicable to the given tile contains information regarding each of these stages. Specifically, the Nmco field of the MCO marker segment gives the number of transformation stages that will be applied during inverse processing.

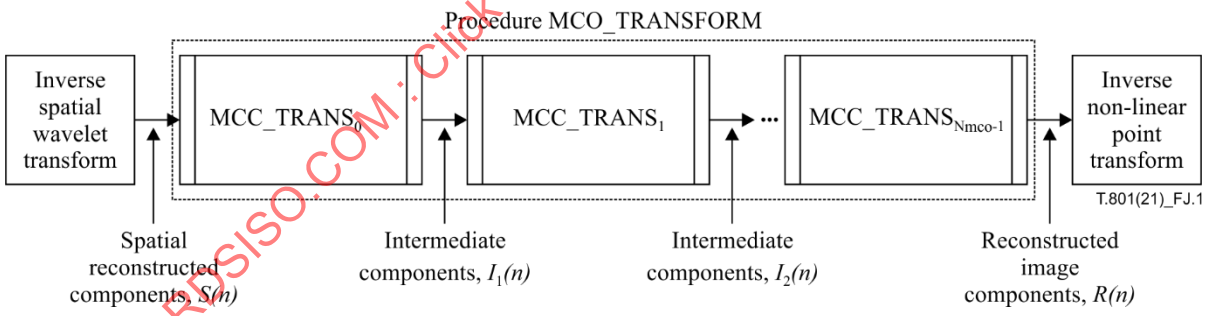

Figure J.1 – Inverse multiple component transformation processing

If Nmco = 0 for a tile, then no inverse multiple component transformation is performed on this tile and the *j*th reconstructed image component is given by the *j*th spatially reconstructed component. In this case, the tile is treated as if there were no multiple component transformations in use. The DC level shifting of tile-components described in Annex G of Rec. ITU-T T.800 | ISO/IEC 15444-1 is performed. The CBD marker segment, which is required with multiple component transformations, still applies to tiles where the multiple component transformation has been turned off. In fact the CBD marker segment applies to all tiles in an image that utilizes multiple component transformations. Since there may be different transformations in use in different tiles, the CBD marker segment shall be constructed to accommodate the largest bit depths found in all tiles for a given component.

When a transformation is performed, the *k*th Imco field of the MCO marker segment contains the index of the MCC marker segment (see clause A.3.8) that applies for the *k*th stage of the inverse transformation. It is recommended, but not required, that decoders complete all processing within a given stage of the inverse transformation before beginning the next stage. (It is possible that an intelligent decoder might be able to determine only those processing steps required to

produce a given set of reconstructed image components. However, completing all processing within a stage before proceeding to the next guarantees correct decoding of the codestream.) A flowchart corresponding to the operations that will result in the successful application of the inverse multiple component transformation is shown in Figure J.2. The processing consists of applying the MCC_TRANS procedure for each of the transformation stages indicated in the MCO marker segment.

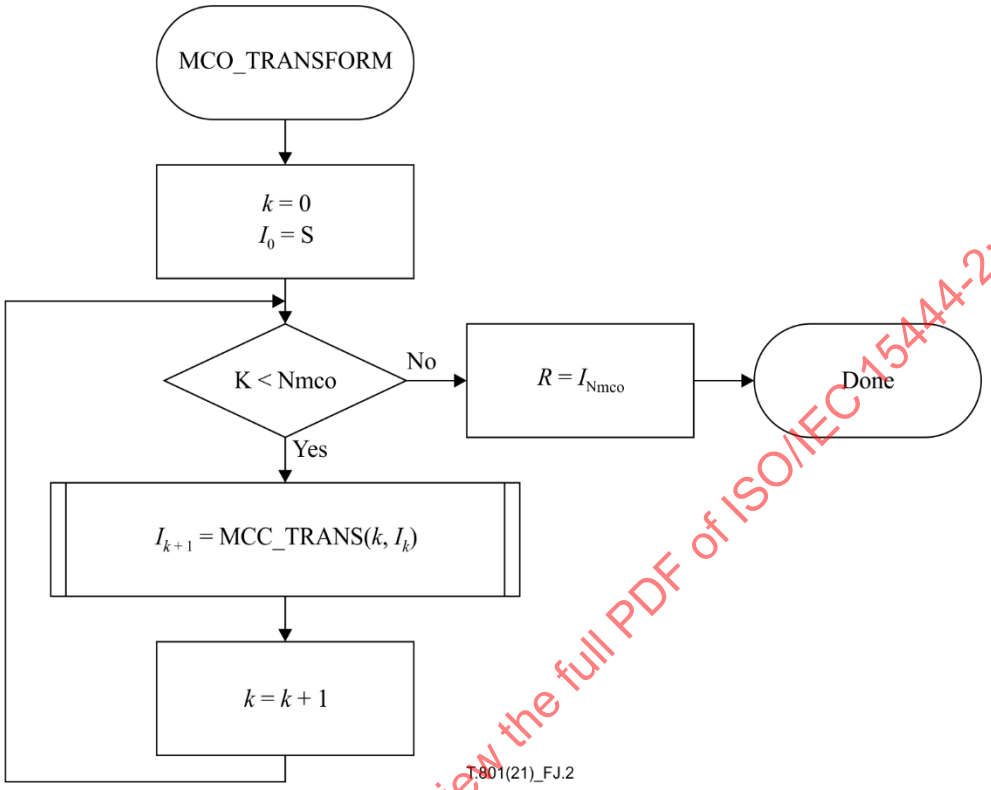

Figure J.2 – Procedure MCO_TRANSFORM

The multiple component transformation mechanism does not place restrictions on the bit depths of the reconstructed image components. Furthermore, it is possible for the number of spatially reconstructed components to differ from the number of reconstructed image components. Therefore, when using the multiple component transformation mechanism, a CBD marker segment (see clause A.3.6) shall be used. This marker segment indicates the total number of output image components and their respective bit depths after the inverse multiple component transformation is applied. Only one CBD marker segment may appear in the main header of the codestream. Thus all tiles in an image shall contain the same number of components, and the component bit depths in the CBD marker segment shall be of sufficient magnitude to cover the maximum bit depth of a component across all tiles.

When multiple component transformation processing is used, the SIZ marker segment (see clause A.2.1) shall indicate the number and bit depths of the components in the codestream after the inverse two-dimensional wavelet transform. In other words, the SIZ marker segment indicates the bit depths of codestream components after the forward multiple component transformation. This interpretation of the SIZ marker segment is subtly different from its interpretation under other decoding processes in this Recommendation | International Standard and Rec. ITU-T T.800 | ISO/IEC 15444-1, where it is used to indicate the number of output image components and their bit depths.

J.2.2 Multiple component transformation stage (MCC_TRANS)

Figure J.3 illustrates the processing involved in a single stage of the inverse multiple component transform. Within a given stage, a set of one or more CC_TRANS operations is performed. The order in which these operations are performed is unimportant; the syntax rules of the MCC marker segment guarantee that the CC_TRANS operations within a stage can be performed in parallel.

The set of input components available to the kth transformation stage, where $k \in [0, 1, \ldots, \text{Nmco}-1]$, is the set of intermediate components I_k. The set of components output from the kth transformation stage is the set of intermediate components I_{k+1}. The first set of intermediate components, I_0, is defined to be the set of spatially reconstructed

components produced by the two-dimensional inverse wavelet transform. Intermediate component set I_0 contains Csiz components, where Csiz is indicated in the SIZ marker segment. If $S(n)$ is the n^{th} spatially reconstructed component, then $I_0(n) = S(n)$, $n = 0, 1, \ldots, \text{Csiz}-1$. Similarly, the set of reconstructed image components is defined to be the final set of intermediate components. Intermediate component set I_{Nmco} contains Ncbd components, where Ncbd is indicated in the CBD marker segment. If $R(n)$ is the nth reconstructed image component, then $R(n) = I_{\text{Nmco}}(n)$, $n = 0, 1, \ldots, \text{Ncbd}-1$. The number of intermediate components, NI_k, in intermediate component set, I_k, for $0 \le k < \text{Nmco}$, is given by:

$$\text{NI}_k = 1 + \max_{i,j}[\text{Cmcc}^{ij}(k)] \tag{J-1}$$

In this expression, the $\text{Cmcc}^{ij}(\text{k})$ are taken from the MCC marker segment corresponding to the kth transformation stage. The max function simply finds the largest Cmcc^{ij} value from the kth transformation stage across all component collections in that stage. The variable $\text{NI}_0 = \text{Csiz}$ and $\text{NI}_{\text{Nmco}} = \text{Ncbd}$.

All of the information regarding the CC_TRANS operations is conveyed in the MCC marker segment. The index of the relevant MCC marker segment for the kth stage of the inverse multiple component transformation is obtained from the Imco^k field of the MCO marker segment. A flow diagram that results in correct decoding of the codestream is given in Figure J.4. (This flow diagram applies the CC_TRANS operations in the order that collections appear within the active MCC marker segment.)

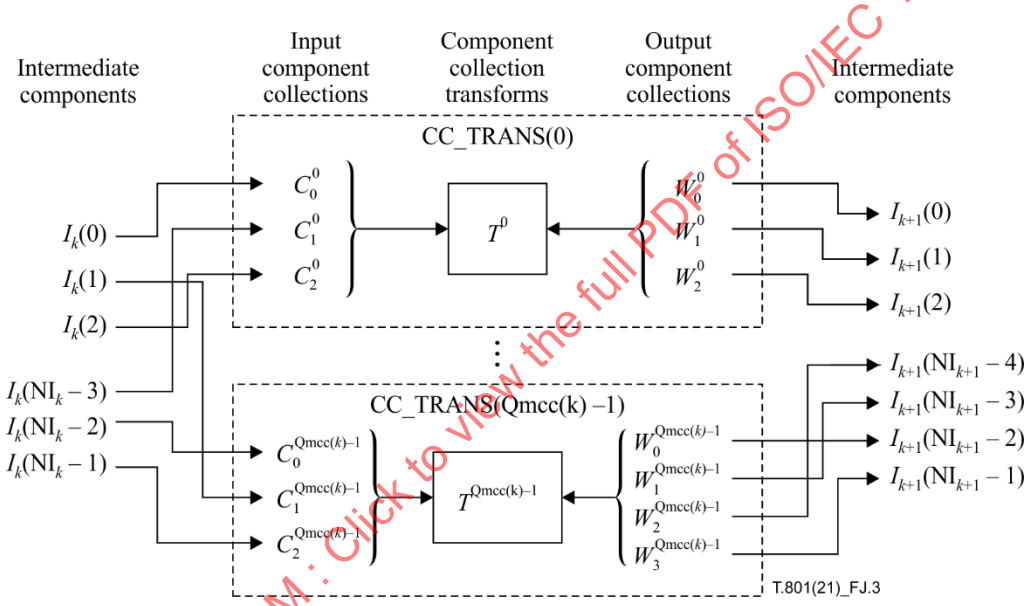

Figure J.3 – A single multiple component collection transformation (MCC_TRANS) stage

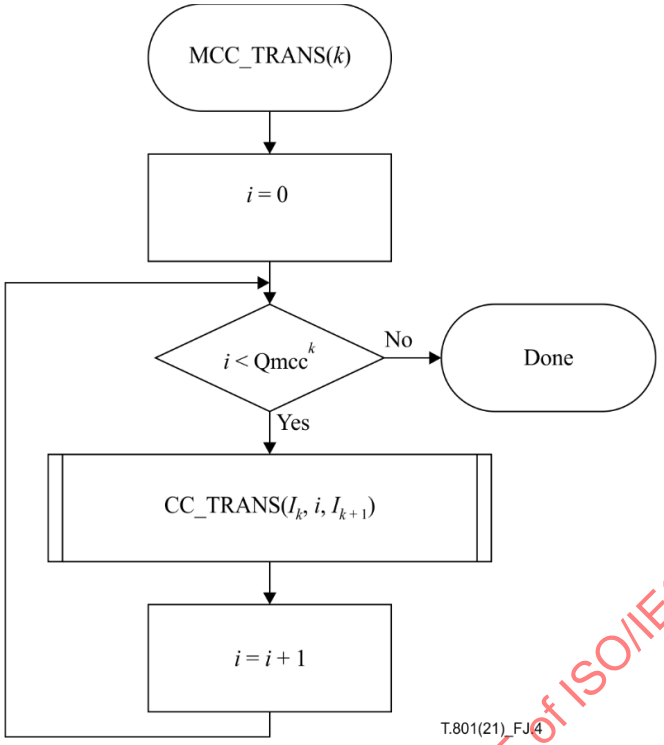

Figure J.4 – Procedure MCC_TRANS

J.2.3 Transformation component collection (CC_TRANS)

The processing flow to transform a given component collection is illustrated in Figure J.5. The figure shows the location of relevant MCC marker segment fields for the *i*th component collection within that marker segment. Each component collection within a transformation stage performs one of a number of different processing steps defined in subsequent clauses of this annex. The *i*th component collection operates on a subset, I_k^i, of the input intermediate components available at the current transformation stage, *k*, and it produces some subset, I_{k+1}^i, of the output intermediate components from the current transformation stage. In the figure, $\text{Cmcc}^i = \{\text{Cmcc}^{ij}\}$, $\forall\, j \in [0, 1, \ldots, \text{Nmcc}^i - 1]$, and $\text{Wmcc}^i = \{\text{Wmcc}^{ij}\}$, $\forall\, j \in [0, 1, \ldots, \text{Mmcc}^i - 1]$. For each of the component collections within the relevant MCC marker segment, the following subclauses, in order, describe the processing that occurs. (These subclauses parallel the functional blocks in Figure J.6.)

All of the transformation methods specified in this annex make use of component collections. A component collection consists of a list of input component indices along with a list of output component indices. Component collections are defined within the MCC marker segment (see clause A.3.8). The input component list of a collection specifies the order in which intermediate components input to the stage are accessed by the transform. In particular, for the *i*th component collection within the *k*th transformation stage, the *j*th input transformation component, C_j^i, is given by $I_k(\text{Cmcc}^{ij})$, where $0 \le j < \text{Nmcc}^i$. Similarly, the output component list of a collection specifies which intermediate components output by the stage are filled by an associated transform. Output component W_j^i (where $0 \le j < \text{Mmcc}^i$) from the transformation is assigned to intermediate component $I_{k+1}(\text{Wmcc}^{ij})$. The indices Cmcc^{ij} and Wmcc^{ij}, and the numbers of input and output components, Nmcc^i and Mmcc^i, all appear in an MCC marker segment. The component collection mechanism allows for permutation of components at both input and output of the associated transformation.

The number of output components may be greater than, less than, or equal to the number of input components. Some restrictions are placed on the relationship between the number of input and output components depending on the type of transformation associated with a collection, as is described later. Also, all components appearing on the input list of a collection shall have the same sample dimensions, as given in Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation B-13. This ensures that a sample from each component in the collection will be available at common locations on the reference grid by enforcing a registration of the input components in the collection.

The transformation that operates on the *i*th collection is identified in the Tmcc^i field of the active MCC marker segment. Additionally, the Tmcc^i field may reference arrays of transformation coefficients which are specified in MCT marker segments (see clause A.3.7) or may identify particular wavelet kernels for use (see clause A.3.5). The Omcc^i field may also be used to provide an offset in the component direction for a one-dimensional wavelet transform. In Figure J.6 and

subsequent subclauses of this annex, T^i will refer to the transformation arrays or wavelet transform information corresponding to the *i*th component collection in the current transformation stage.

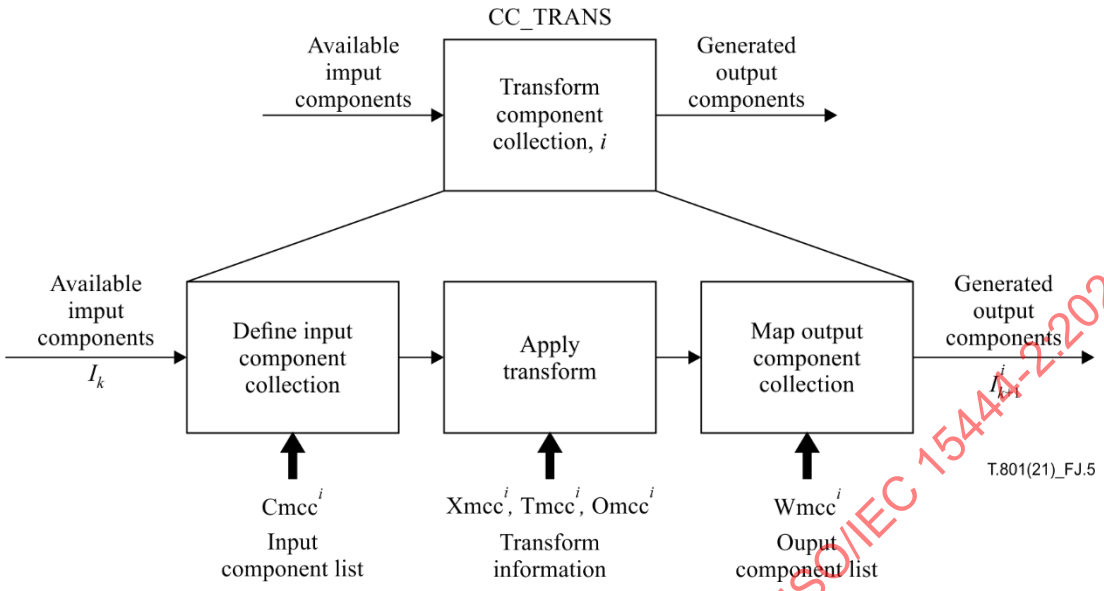

Figure J.5 – A single component collection transformation (CC_TRANS) stage

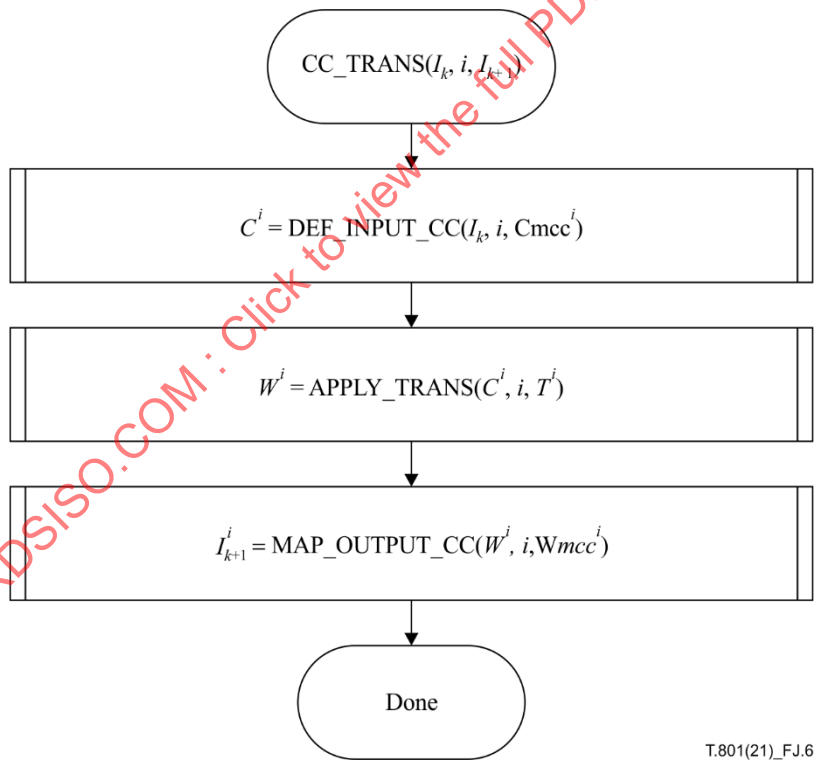

Figure J.6 – Procedure CC_TRANS

J.2.3.1 Define input component collection (DEF_INPUT_CC)

For the current collection, *i*, the set C^i of input components for the transformation is formed by selecting a subset of the available intermediate components, I_k. This set contains Nmcc^i components. The *j*th component, C_j^i, in the input set is given by intermediate component I_k(Cmcc^{ij}), where $0 \le j <$ *N*mcc^i and Cmcc$^{ij} \in$[0, 1, ... , NI$_k$ – 1]. The index *k* is the transformation stage number, and the values of Nmcc^i and Cmcc^{ij} are defined in the *i*th collection of the MCC marker segment corresponding to transformation stage *k*. This MCC marker segment has the same index value, Imcc, as that given for the *k*th transformation stage, Imco^k, in the MCO marker segment.

It is required that the component collections comprising a transformation stage "touch" every available input intermediate component (i.e., the set of output intermediate components from the previous transformation stage). This means that every intermediate component index from 0 to $NI_k - 1$ shall be present in the input component list of at least one component collection. If a given input intermediate component is not used in any transformation process in the current transformation stage, then it shall appear in a null transformation (see clause J.3). As stated above, there is no general requirement that the number of output components from the transformation process equal the number of input components. It is therefore possible that additional components are inserted (or created) during the multiple component transformation process. This processing step does not necessarily occur explicitly during the transformation process. It may occur implicitly through the use of null intermediate components.

A null component is one whose output was left undefined by the previous transformation stage (i.e., the (k – 1)th stage) in the formation of the intermediate components, I_k. This can occur when the union of all output intermediate component lists from the previous transformation stage does not include every component number between 0 and the largest input component number, $NI_k - 1$, of the current transformation stage. In other words, there may be gaps in the set of output intermediate component numbers from the previous transformation stage. When a null component is accessed in a given input intermediate component collection, it is treated by the transformation process as a component with values that are identically zero. (The informative example in clause O.3 illustrates a potential use of null components.).

When forming a series of transformation stages that work together, care shall be taken in the generation of the output and input component lists between successive transformation stages. It is not permitted for an output component number in stage k–1 to exceed the value $NI_k - 1$ for the current transformation stage k.

J.2.3.2 Apply transformation (APPLY_TRANS)

Given the input component set C^i, which contains $Nmcc^i$ components, the selected inverse multiple component transformation is applied. The $Xmcc^i$ field of the active MCC marker segment indicates the type of transformation used to transform the ith component collection. The allowed transformations and their application are discussed in clause J.3. The $Tmcc^i$ field of the active MCC marker segment provides additional information that is required for the particular transformation, such as pointers to transformation array coefficients contained in an MCT marker segment (see clause A.3.7), wavelet transform kernels contained in an ATK marker segment (see clause A.3.5), number of wavelet transform levels, and indicators of the reversibility of the transform. In the case of the wavelet transform, the $Omcc^i$ field also provides the equivalent of a tile offset for the one-dimensional wavelet transform. Application of the transformation results in a set of output components, W^i, which contains $Mmcc^i$ components.

J.2.3.3 Assign output components (MAP_OUTPUT_CC)

The transformation for the ith component collection produces the set of $Mmcc^i$ output components, W^i. The W^i are then assigned to a subset of the output intermediate components from the stage, I_{k+1}. Specifically, output component W_j^i (where $0 \le j < Mmcc^i$) from the transformation is assigned to intermediate component $I_{k+1}(Wmcc^{ij})$. This subset of the output intermediate component is also referred to as I_{k+1}^i. The $Mmcc^i$ and $Wmcc^{ij}$ values are found in the ith component collection of the active MCC marker segment for the kth transformation stage.

It is required that no output intermediate component appear more than once in the union of all output component collection lists within a particular MCC marker segment (i.e., $Wmcc^i \cap Wmcc^j = \varnothing, \forall\ i \ne j$). This rule implies that all component collections may be transformed in parallel without danger of overwriting previously computed results. As noted above, the output component list may be incomplete, thus allowing the transformation stage to create null components. The maximum output component number may not exceed $NI_{k+1} - 1$, the maximum input component number of the succeeding transformation stage. There is no requirement that the output component collection list be complete for the final transformation stage. However, the utility of such null image components is dubious, and their use is not recommended.

J.3 Transformations

This clause details the mathematics involved in the application of an inverse multiple component transformation. It also describes the location and interpretation of the remaining fields in the i^{th} component collection of the active MCC marker segment that are required for application of the specific transformation.

For each of the transformations discussed in this clause, it is assumed that the input component collection of the transformation, $C = C^i$, has already been formed. Individual components within the set are denoted as C_j, where $j = [0, 1, \ldots, N - 1]$, and $N = Nmcc^i$. All of the described transformations are assumed to produce a set of transformed components, $W = W^i$, with members denoted by W_j, where $j = [0, 1, \ldots, M - 1]$, and $M = Mmcc^i$. This set of transformed components comprises the output component collection. These two sets of components will be referred to generically as input and output components of the transform. The equations presented in this clause are normative only in the sense that

they describe a result that the decoder shall achieve; different implementations of these equations may exist for fully compliant decoders.

J.3.1 Array-based transforms

Array-based transformations are those that can be described by a set of equations that are linear in the input components. The transformation coefficients applied to the components in the following equations, as well as additive component offsets, are referred to as arrays. These arrays are stored within the codestream in MCT marker segments (see clause A.3.7). For array-based transformations, the $Tmcc^i$ field of *i*th component collection in the active MCC marker segment contains the index of a transformation array and the index of an offset array. The $Xmcc^i$ field for the *i*th component collection in the active MCC marker segment defines the type of array-based transformation to be applied (decorrelation or dependency, reversible or irreversible) to the component collection.

If the $Tmcc^i$ array indices are non-zero, these indices, along with knowledge of the type of array-based transformation being applied, are used to select the appropriate MCT marker segments from which the coefficients are extracted. For the transformation array, the appropriate MCT marker segment is found by matching the MCT marker segment index found in the Imct parameter against the $Tmcc^i$ transformation array index, and then matching the $Xmcc^i$ transformation type against the array type found in the Imct parameter. For the offset array, the appropriate MCT marker segment is found by matching the MCT marker segment index found in the Imct parameter against the $Tmcc^i$ offset array index, and then checking that the array type found in the Imct parameter indicates the array is an offset array. An index of zero in a $Tmcc^i$ field indicates a null transformation or offset array, so that particular transformation steps may be skipped. For each of the array-based transformations discussed in this clause, the number and storage order of the coefficients within the MCT marker segment is defined.

J.3.1.1 Decorrelation transformation

The decorrelation transformation type provides for an unconstrained linear combination of input components with additive offsets for each result. This transformation structure enables full matrix transformations such as the KLT.

J.3.1.1.1 Irreversible decorrelation transformation

The irreversible decorrelation transformation consists of a matrix multiplication of the input components followed by application of an additive offset. The transformation is applied by using Equation J-2.

$$\begin{aligned} W_0 &= t_{00}C_0 + t_{01}C_1 + t_{02}C_2 + t_{03}C_3 + \ldots + t_{0(N-1)}C_{N-1} + o_0 \\ W1 &= t_{10}C_0 + t_{11}C_1 + t_{12}C_2 + t_{13}C_3 + \ldots + t_{1(N-1)}C_{N-1} + o_1 \\ W2 &= t_{20}C_0 + t_{21}C_1 + t_{22}C_2 + t_{23}C_3 + \ldots + t_{2(N-1)}C_{N-1} + o_2 \\ W3 &= t_{30}C_0 + t_{31}C_1 + t_{32}C_2 + t_{33}C_3 + \ldots + t_{3(N-1)}C_{N-1} + o_3 \\ &\vdots \end{aligned} \tag{J-2}$$

If the decorrelation transformation array index provided by the $Tmcc^i$ field for this component collection is zero, then the coefficients t_{ij} are given by $t_{ij} = 1$ for $i = j$ and $t_{ij} = 0$ for $i \neq j$. If the decorrelation transformation array index is not zero, then the referenced MCT marker segment contains $M \cdot N$ elements. The coefficients t_{ij} are stored in the marker segment in the following order: t_{00}, t_{01}, ... , $t_{0(N-1)}$, t_{10}, t_{11}, ... , $t_{1(N-1)}$, ... $t_{(M-1)(N-1)}$.

If the offset array index provided by the $Tmcc^i$ field for this component collection is zero, then the coefficients o_i are given by $o_i = 0$. If the offset array index is not zero, then the referenced MCT marker segment contains M elements. The coefficients o_i are stored in the marker segment in the following order o_0, o_1, ... , o_{M-1}.

For an irreversible decorrelation transformation, the number of input components, N, is not required to equal the number of output components, M.

J.3.1.1.2 Forward irreversible decorrelation transformation (informative)

At a particular spatial location, (x, y), the M image components to be transformed are denoted by W_0, W_1, ... , W_{M-1}. The component dc offsets are given by o_0, o_1, ... , o_{M-1}, and the N components that result from the transformation are denoted by C_0, C_1, ... , C_{N-1}. The forward irreversible decorrelation transformation is applied by using Equation J-3.

$$\begin{aligned} C_0 &= t_{00}(W_0 - o_0) + t_{01}(W_1 - o_1) + t_{02}(W_2 - o_2) + \ldots + t_{0(M-1)}C_{M-1} \\ C_1 &= t_{10}(W_0 - o_0) + t_{11}(W_1 - o_1) + t_{12}(W_2 - o_2) + \ldots + t_{1(M-1)}C_{M-1} \\ C_2 &= t_{20}(W_0 - o_0) + t_{21}(W_1 - o_1) + t_{22}(W_2 - o_2) + \ldots + t_{2(M-1)}C_{M-1} \\ &\vdots \end{aligned} \tag{J-3}$$

The offsets o_0, o_1, ... , o_{M-1} are included in an MCT marker segment. The offset array index provided by the Tmcc^i field for this component collection should match the MCT array index. If the offsets are all equal to zero, then the Tmcc^i field for this component collection can be set to zero and the offsets do not need to be included in an MCT marker segment.

The t_{ij} coefficients for the inverse transformation that are included in the MCT marker segment are in general not the same as those that appear in forward transformation equation. For example, if the transformation for this component collection is unitary, then the inverse transformation coefficients are the matrix transpose of the forward transformation coefficients. It is the responsibility of the encoder to form correctly the inverse transformation information required by the decoder.

J.3.1.1.3 Reversible decorrelation transform

The reversible decorrelation transformation consists of a set of single element linear transformations followed by application of an additive offset. For the reversible decorrelation transformation, the number of input components, N, is required to equal the number of output components, M. This is true even though the number of single element linear transformations applied in the processing is $N + 1$. The transformation is applied by the following set of equations:

Let temporary variable P be defined as:

$$\begin{aligned} P_0 &= C_0 \\ P_1 &= C_1 \\ P_2 &= C_2 \\ P_3 &= C_3 \\ &\vdots \end{aligned}$$

then form the following sequence of single element linear transformations using Equation J-4 and the given rounding rule.

$$\left.\begin{aligned} S_l &= \sum_{i=0, i\neq(N-1-l)}^{N-1} t_{li} P_i + \frac{t_{l(N-1-l)}}{2} \\ PT_{N-1-l} &= -\left\lfloor \frac{S_1}{t_{1(N-1-l)}} \right\rfloor + P_{N-1-l} \\ P_{N-1-l} &= PT_{N-1-l} \end{aligned}\right\} l = 0,1,\ldots,N-1 \qquad \text{(J-4)}$$

Next compute the final single element transformation using Equation J-5 and apply the additive offset in Equation J-6 to form the output intermediate components. For the reversible decorrelation transformation, the sums S_l, l = [0, 1, ... , N], shall be generated in order and the single output term corresponding to that sum, P_{N-1-l}, shall be adjusted before the next sum is computed.

$$\begin{aligned} S_N &= \sum_{i=0}^{N-1} t_{Ni} P_i + \frac{|t_{N(N-1)}|}{2} \\ PT_{N-1} &= sgn(t_{N(N-1)}) \cdot \left[-\left\lfloor \frac{S_N}{|t_{N(N-1)}|} \right\rfloor + P_{N-1} \right] \\ P_{N-1} &= PT_{N-1} \end{aligned} \qquad \text{(J-5)}$$

$$\begin{aligned} W_0 &= P_0 + o_0 \\ W_1 &= P_1 + o_1 \\ W_2 &= P_2 + o_2 \\ W_3 &= P_3 + o_3 \\ &\vdots \end{aligned} \qquad \text{(J-6)}$$

Figure J.7 illustrates the computations described in Equations J-4 through J-6. Each stage of the transform adjusts exactly one output component. A linear combination of the unaltered components at each stage is first formed. This partial sum is rounded by a reversible rounding rule. The integer result is then added to the component that is adjusted at that stage (and the result is possibly negated at the last stage). These operations result in all-integer component values at each stage. Furthermore, since only a single component value is altered by an integer addition at each stage, the transform can be reversed simply by reversing the order of the single element transform steps. Also shown in the figure are two permutation matrices $P_R J$ and P_L which are incorporated into the structure of the reversible decorrelation transform. Here they explicitly indicate that the SERM factorization may not produce the transformed output components in the same order as the decorrelation matrix it is approximating. Given that the SERM factorization may not generate transformed components in the same order as the unitary transform it approximates, an encoder may choose to rectify this situation via the use of the output intermediate component collection index list.

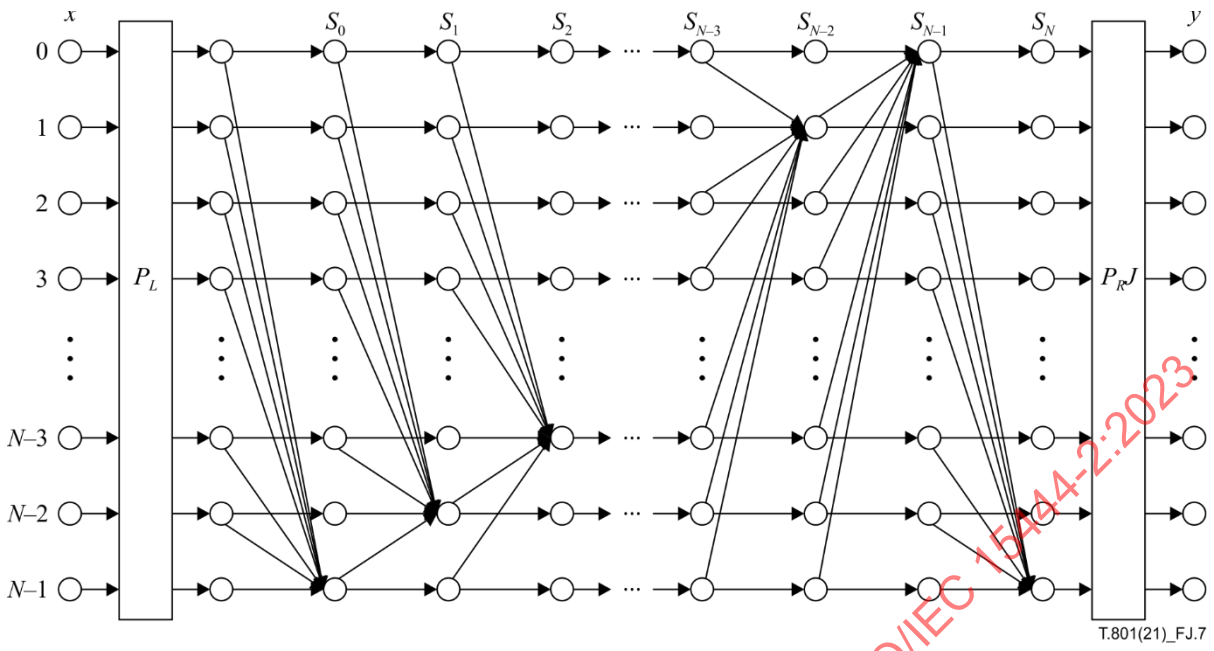

Figure J.7 – SERM implementation of reversible decorrelation transformation

NOTE – (informative): This reversible decorrelation transform structure accommodates a reversible factorization of a unitary $N \times N$ transform matrix that approximates its decorrelation properties. One such factorization technique, known as the single-row elementary reversible matrix (SERM) factorization is, described in [14]. Unitary transforms comprise a large class of multiple component transforms (examples include the KLT). An informative example is given in clause O.3 based upon the technique presented in [14].

If the decorrelation transformation array index provided by the Tmcc^i field for this component collection is zero, then the coefficients t_{ij} are given by $t_{N(N-1)} = 1$, $t_{i(N-1-i)} = 1$ for $i \in [0, \ldots, N-1]$, and $t_{ij} = 0$ for all other i,j. If the decorrelation transformation array index is not zero, then the referenced MCT marker segment contains $(N + 1) \cdot N$ elements. The coefficients t_{ij} are stored in the marker segment in the following order: $t_{00}, t_{01}, \ldots, t_{0(N-1)}, t_{10}, t_{11}, \ldots, t_{1(N-1)}, \ldots t_{N(N-1)}$. The coefficients are constrained to be integers for the reversible decorrelation transformation. Furthermore, the coefficients $t_{N(N-1)}$ and $t_{i(N-1-i)}$ for $\in [0, \ldots, N-1]$ are constrained to be exact positive integer powers of 2, while $t_{N(N-1)}$ is constrained to have an absolute value equal to an exact positive integer power of 2. This subset of the coefficients is interpreted as a set of scaling factors for each of the partial sums that are formed. This allows real-valued coefficients to be approximated to the nearest desired fractional bit. The restrictions on the coefficient values ensure that the sums can be carried out, if desired, with all-integer mathematical operations.

If the offset array index provided by the Tmcc^i field for this component collection is zero, then the coefficients o_i are given by $o_i = 0$. If the offset array index is not zero, then the reference MCT marker segment contains M elements. The coefficients o_i are stored in the marker segment in the following order: $o_0, o_1, \ldots, o_{M-1}$. For the reversible decorrelation transformation, the o_i are required to be integers.

J.3.1.1.4 Forward reversible decorrelation transformation (informative)

At a particular spatial location, (x, y) the N image components to be transformed are denoted by $W_0, W_1, \ldots, W_{N-1}$. The component dc offsets are given by $o_0, o_1, \ldots, o_{N-1}$, and the components that result from the transformation are denoted by $C_0, C_1, \ldots, C_{N-1}$. The forward irreversible decorrelation transformation is applied by using Equations J-7 through J-10. The SERM implementation of a forward reversible decorrelation transformation is shown in Figure J.8.

$$\begin{aligned} P_0 &= W_0 - o_0 \\ P_1 &= W_1 - o_1 \\ P_2 &= W_2 - o_2 \\ P_3 &= W_3 - o_3 \\ &\vdots \end{aligned} \tag{J-7}$$

$$S_0 = \sum_{i=0}^{N-2} t_{0i} P_i + \frac{|t_{0(N-1)}|}{2}$$

$$PT_{N-1} = \left\lfloor \frac{S_0}{|t_{0(N-1)}|} \right\rfloor + sgn(t_{0(N-1)}) P_{N-1}$$

$$P_{N-1} = PT_{N-1} \tag{J-8}$$

$$\left.\begin{aligned} S_l &= \sum_{i=0, i\neq(l(-1))}^{N-1} t_{li} P_i + \frac{t_{l(l-1)}}{2} \\ PT_{l-1} &= -\left\lfloor \frac{S_l}{t_{l(l-1)}} \right\rfloor + P_{l-1} \\ P_{l-1} &= PT_{l-1} \end{aligned}\right\} l = 1,2,\ldots,N \tag{J-9}$$

$$\begin{aligned} C_0 &= P_0 \\ C_1 &= P_1 \\ C_2 &= P_2 \\ C_3 &= P_3 \\ &\vdots \end{aligned} \tag{J-10}$$

The input component values are first shifted by a dc offset. The results are assigned to temporary variables P_i. The N input components are then transformed in a series of $N + 1$ steps. In each step, exactly one of the input values is altered. The transformation steps are carried out sequentially, and the temporary values, P_i, are updated during each step and used in subsequent steps. In the first step, given in Equation J-8, the last temporary value is altered. Equation J-9 is then applied N times, with the value of l running from 1 to N. In the lth of these steps, temporary value P_l is altered. The final set of temporary values becomes the output of the transformation.

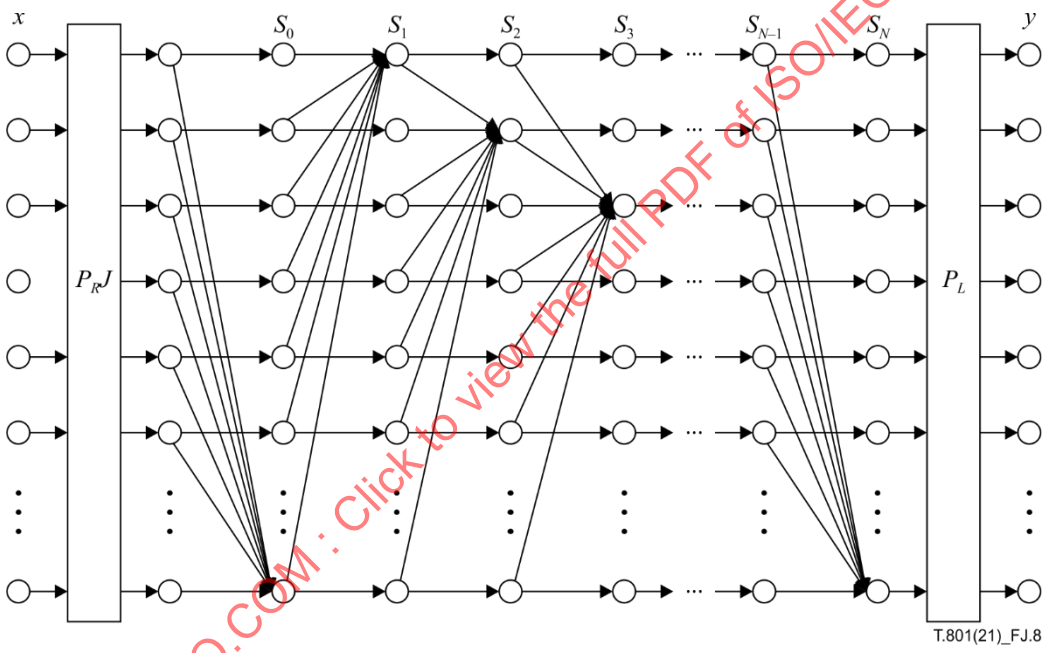

Figure J.8 – SERM implementation of forward reversible decorrelation transformation

The offsets $o_0, o_1, \ldots, o_{M-1}$ are integer valued and are included in an MCT marker segment. The offset array index provided by the Tmcc^i field for this component collection match the MCT array index. If the offsets are all equal to zero, then the Tmcc^i field for this component collection can be set to zero and the offsets do not need to be included in an MCT marker segment.

The t_{li} coefficients for the inverse transformation that are included in the MCT marker segment are in general not the same as those that appear in forward transformation equation. In general, it will be true that t_{ij} in Equations J-4 and J-5 is equal to $t_{(N-i)j}$ in Equations J-8 and J-9. There are some additional constraints on the coefficient values for the forward reversible decorrelation transformation. All of the t_{ij} are integer valued, $t_{0(N-1)}$ has an absolute value that is a power of 2, and $t_{i(i-1)}$ for i = 1, 2, ... , N shall be an exact power of 2. The coefficients that are powers of 2 can be interpreted in the equations as scale factors for each step in the transformation. Clause O.3 provides an informative example illustrating a decorrelation transformation, its SERM factorization, and associated reversible implementation.

J.3.1.2 Dependency transformation

The dependency transformation type allows for predictive transformations. Inherent in the dependency transformation is the concept that the $(j + 1)^{th}$ output component can be computed only after the jth output component is decoded. The dependency transformation structure enables usage of prediction based DPCM-like transforms. An example of this transformation type is given in clause O.3.

J.3.1.2.1 Irreversible dependency transformation

The irreversible dependency transformation consists of an additive offset followed by a constrained linear combination of components. The dependency transformation is defined by the following set of equations:

$$\begin{aligned} Y_0 &= C_0 + o_0 \\ Y_1 &= C_1 + o_1 \\ Y_2 &= C_2 + o_2 \\ Y_3 &= C_3 + o_3 \\ &\vdots \end{aligned} \tag{J-11}$$

$$\begin{aligned} W_0 &= Y_0 \\ W_1 &= t_{10}W_0 + Y_1 \\ W_2 &= t_{20}W_0 + t_{21}W_1 + Y_2 \\ W_3 &= t_{30}W_0 + t_{31}W_1 + t_{32}W_2 + Y_3 \\ &\vdots \end{aligned} \tag{J-12}$$

Equation J-12 implies a particular structure in the irreversible dependency transformation array. Specifically, the irreversible dependency transformation array shall be lower left triangular with zeros on and above the main diagonal. This particular structure guarantees that the array may be processed from top to bottom and causality will be preserved. This places a responsibility upon the encoder to form correctly the irreversible dependency transformation array in conjunction with the input and output intermediate component collection index lists. Figure J.9 illustrates the processing steps in the irreversible dependency transformation. The box labelled, $P(W_0, W_1, \ldots, W_{j-1})$, in the diagram is where the predictions for the current component being decoded is formed from previously decoded components.

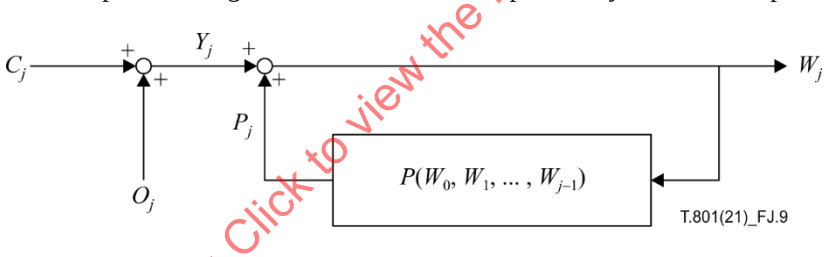

Figure J.9 – Irreversible dependency transformation

If the dependency transformation array index provided by the Tmcc^i field for the i^{th} component collection is zero, then the coefficients t_{ij} are given by $t_{ij} = 0$ for all i, j. If the dependency transformation array index is not zero, then the referenced MCT marker segment contains $M \cdot (M - 1)/2$ elements. The coefficients t_{ij} are stored in the marker segment in the following order: $t_{10}, t_{20}, t_{21}, t_{30}, t_{31}, t_{32}, \ldots, t_{(M-1)0}, \ldots, t_{(M-1)(M-2)}$.

If the offset array index provided by the Tmcc^i field for the ith component collection is zero, then the coefficients o_i are given by $o_i = 1$. If the offset array index is not zero, then the referenced MCT marker segment contains M elements. The coefficients o_i are stored in the marker segment in the following order: $o_0, o_1, \ldots, o_{M-1}$.

For a dependency transformation, the number of input components, N, is required to equal the number of output components, M.

J.3.1.2.2 Forward irreversible dependency transformation (informative)

The forward irreversible dependency transformation consists of a constrained linear combination of components and an additive offset. At a particular spatial location, (x, y), the N image components to be transformed are denoted by W_0, W_1, ... , W_{N-1}. The component dc offsets are given by o_0, o_1, ... , o_{N-1}, and the components that result from the transformation are denoted by C_0, C_1, ... , C_{N-1}. The forward dependency transformation is defined by the following set of equations and illustrated in Figure J.10:

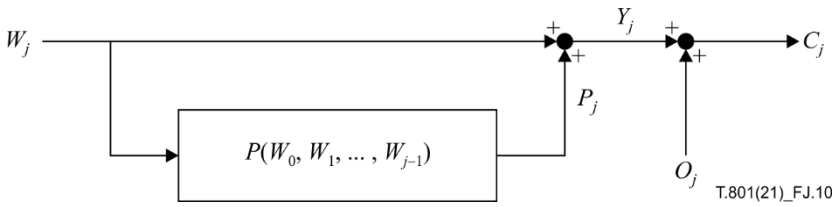

Figure J.10 – Forward irreversible dependency transformation

$$\begin{aligned} C_0 &= W_0 - o_0 \\ C_1 &= W_1 - o_1 - t_{10}W_0 \\ C_2 &= W_2 - o_2 - t_{20}W_0 - t_{21}W_1 \\ C_3 &= W_3 - o_3 - t_{30}W_0 - t_{31}W_1 - t_{32}W_2 \\ &\vdots \end{aligned} \tag{J-13}$$

The offsets o_0, o_1, ... , o_{M-1} are included in an MCT marker segment. The offset array index provided by the Tmcc^i field for this component collection should match the MCT array index. If the offsets are all equal to zero, then the Tmcc^i field for this component collection can be set to zero and the offsets do not need to be included in an MCT marker segment.

The t_{ij} coefficients for the inverse transformation that are included in the MCT marker segment are in general the same as those that appear in forward transformation equation. In Equation J-12, all operations are additions, whereas in the forward transformation, all operations are subtractions.

J.3.1.2.3 Reversible dependency transformation

The reversible dependency transformation consists of an additive offset followed by a constrained linear combination of components. The reversible dependency transformation is defined by the following set of equations:

$$\begin{aligned} Y_0 &= C_0 + o_0 \\ Y_1 &= C_1 + o_1 \\ Y_2 &= C_2 + o_2 \\ Y_3 &= C_3 + o_3 \\ &\vdots \end{aligned} \tag{J-14}$$

$$\begin{aligned} W_0 &= Y_0 \\ S_1 &= t_{10}W_0 + \left\lfloor \frac{t_{11}}{2} \right\rfloor \\ W_1 &= \left\lfloor \frac{S_1}{t_{11}} \right\rfloor + Y_1 \\ S_2 &= t_{20}W_0 + t_{21}W_1 + \left\lfloor \frac{t_{22}}{2} \right\rfloor \\ W_2 &= \left\lfloor \frac{S_2}{t_{22}} \right\rfloor + Y_2 \\ S_3 &= t_{30}W_0 + t_{31}W_1 + t_{32}W_2 + \left\lfloor \frac{t_{33}}{2} \right\rfloor \\ W_3 &= \left\lfloor \frac{S_3}{t_{33}} \right\rfloor + Y_3 \\ &\vdots \end{aligned} \tag{J-15}$$

Equation J-15 implies a particular structure in the reversible dependency transformation array. Specifically, the reversible dependency transformation array shall be lower left triangular with zeros above the main diagonal. The coefficients on the main diagonal of the array (except t_{00}) are scaling factors that may be used to scale real-valued dependency transformation arrays and represent them with a desired number of fractional bits. This particular structure guarantees that the array may be processed from top to bottom and causality will be preserved. This places a responsibility upon the encoder to properly form the reversible dependency transformation array in conjunction with the input and output

intermediate component collection index lists. Figure J.11 illustrates the reversible dependency transformation processing. The box labelled "R" represents a rounding rule, which has been defined as the floor function in this Recommendation | International Standard.

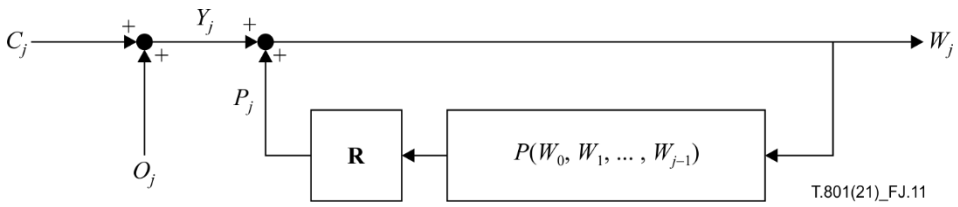

Figure J.11 – Reversible dependency transformation

If the dependency transformation array index provided by the Tmcc^i field for this component collection is zero, then the coefficients t_{ij} are given by $t_{ij} = 0$ for all $i \neq j$, and $t_{ij} = 1$ for $i = j$. If the dependency transformation array index is not zero, then the referenced MCT marker segment contains $\frac{M(M+1)}{2} - 1$ elements. The coefficients t_{ij} are stored in the marker segment in the following order: t_{10}, t_{11}, t_{20}, t_{21}, t_{22}, t_{30}, t_{31}, t_{32}, t_{33}, ... , $t_{(M-1)0}$, ... , $t_{(M-1)(M-1)}$. For $i \neq j$, the t_{ij} coefficients have the same interpretation as in the irreversible dependency transformation. The additional $M - 1$ coefficients, t_{ij} where $i = j$ and $i > 0$, are interpreted as scaling factors for the partial sums S_l. The t_{ij} coefficients are constrained to be integers in the reversible dependency transformation. Furthermore, the t_{ij} coefficients for $i = j$ and $i > 0$ are constrained to be exact positive integer powers of 2. These restrictions ensure that the partial sums can be formed, if desired, with all-integer mathematical operations.

If the offset array index provided by the Tmcc^i field for this component collection is zero, then the coefficients o_i are given by $o_i = 0$. If the offset array index is not zero, then the referenced MCT marker segment contains M elements. The coefficients o_i are stored in the marker segment in the following order: o_0, o_1, ... , o_{M-1}. These coefficients are constrained to be integers for the reversible dependency transformation.

For a reversible dependency transformation, the number of input components, N, is required to equal the number of output components, M.

J.3.1.2.4 Forward reversible dependency transformation (informative)

The forward reversible dependency transformation consists of a constrained linear combination of components and an additive offset. At a particular spatial location, (x, y), the N image components to be transformed are denoted by W_0, W_1, ... , W_{N-1}. The component dc offsets are given by o_0, o_1, ... , o_{N-1}, and the components that result from the transformation are denoted by C_0, C_1, ... , C_{N-1}. The forward dependency transformation is defined by the following set of equations. The forward dependency transformation is defined by the following set of equations and illustrated in Figure J.12:

$$
\begin{aligned}
C_0 &= W_0 - o_0 \\
S_1 &= t_{10} W_0 + \left\lfloor \frac{t_{11}}{2} \right\rfloor \\
C_1 &= \left\lfloor \frac{S_1}{t_{11}} \right\rfloor + W_1 - o_1 \\
S_2 &= t_{20} W_0 + t_{21} W_1 + \left\lfloor \frac{t_{22}}{2} \right\rfloor \\
C_2 &= -\left\lfloor \frac{S_2}{t_{22}} \right\rfloor + W_1 - o_2 \\
S_3 &= t_{30} W_0 + t_{31} W_1 + t_{32} W_2 + \left\lfloor \frac{t_{33}}{2} \right\rfloor \\
C_3 &= \left\lfloor \frac{S_3}{t_{33}} \right\rfloor + W_3 - o_3 \\
&\vdots
\end{aligned}
\tag{J-16}
$$

The offsets o_0, o_1, ... , o_{M-1} are integer valued and are included in an MCT marker segment. The offset array index provided by the Tmcc^i field for this component collection match the MCT array index. If the offsets are all equal to zero, then the Tmcc^i field for this component collection can be set to zero and the offsets do not need to be included in an MCT marker segment.

The t_{ij} coefficients for the inverse transformation that are included in the MCT marker segment are in general the same as those that appear in forward transformation equation. There are constraints on these coefficient values, however. In particular, all t_{ij} are integer valued, and t_{ii} for $i = 1, 2, \ldots, N-1$ are exact powers of 2. (There is no t_{00} coefficient; the transformation equation implicitly assumes it to be equal to 1.) The t_{ii} can be interpreted as scale factor for the steps in the transformation.

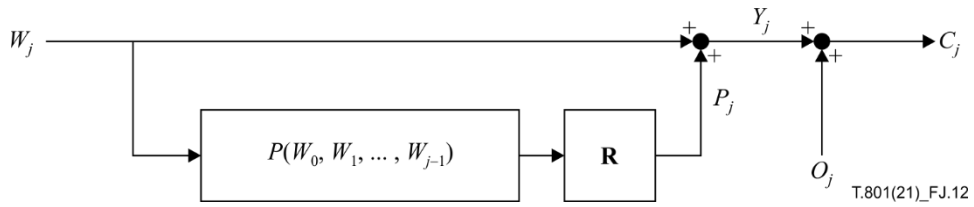

Figure J.12 – Forward reversible dependency transformation

J.3.2 Wavelet-based transformation

This clause describes a wavelet-based decorrelation process. The MCC marker segment allows for the specification of a wavelet transformation to be applied to i^{th} input component collection via the Xmcc^i marker parameter (see clause A.3.8). In fact, it is possible to use wavelet-based decorrelation on one component collection and array-based decorrelation on another within the same MCC marker segment. When wavelet-based decorrelation is used, an ATK marker segment (see clause A.3.5) specifies the wavelet transformation kernel used in processing the component collection. The Tmcc^i parameter in the MCC marker segment contains an index that specifies which ATK marker segment is to be used for the i^{th} component collection. The Tmcc^i parameter also includes an index that points to an MCT marker segment containing additive offsets to apply to the input intermediate components after inverse wavelet transformation processing. The MCC marker segment signals the number of wavelet transformation levels in Tmcc^i, and a component offset Omcc^i to be employed during inverse wavelet transformation processing.

For a wavelet-based decorrelation transform, the number of input components, N = Nmcc^i, is required to equal the number of output components, M = Mmcc^i.

J.3.2.1 Inverse multi-dimensional wavelet transformation

The samples of component C_j are denoted by $C_j(x, y)$. At each spatial location, (x, y), $N_L + 1$ two-dimensional arrays are formed. Here, N_L is the number of wavelet transformation levels applied to the ith collection as determined from the Tmcc^i marker parameter. The arrays are denoted by $a_{N_L\mathrm{LL}}(u, 0)$, $a_{N_L\mathrm{HL}}(u, 0)$, $a_{(N_L-1)\mathrm{HL}}(u, 0)$, ... , $a_{1\mathrm{HL}}(u, 0)$. In each case, $tbx_0 \le u < tbx_1$, where tbx_0 and tbx_1 are determined appropriately for sub-band a_b as specified in Annex B of Rec. ITU-T T.800 | ISO/IEC 15444-1. For the purposes of this annex, it is assumed that the transformation is applicable to a tile-component with upper left corner at (tcx_0, tcy_0) = (Omcc^i,0) and lower right corner at $(tcx_1 - 1, tcy_1 - 1)$ = (Omcc^i + M – 1, 0).

The (tbx_1, tbx_0) samples of each array, $a_b(u, 0)$, are taken from the $N = M$ samples $C_j(x, y)$, $j = [0, 1, \ldots, N-1]$. These are taken in order of the arrays as given above. Specifically, let $b = N_L\mathrm{LL}$. Then the first $tbx_1 - tbx_0 = tN_L\mathrm{LL}x_1 - tN_L\mathrm{LL}x_0$ samples of $C_j(x, y)$ (i.e., $j = [0, 1, \ldots, tbx_1 - tbx_0 - 1]$) become the samples $a_b(u, 0)$, $u = [tbx_0, \ldots, tbx_1 - 1]$. Similarly, let $b = N_L\mathrm{HL}$. Then the next $tbx_1 - tbx_0 = tN_L\mathrm{HL}x_1 - tN_L\mathrm{HL}x_0$ samples of $C_j(x, y)$ become the samples of $a_b(u, 0)$, $u = [tbx_0, \ldots, tbx_1 - 1]$, and so on.

The a_b arrays defined above are treated as sub-bands to be inverse wavelet transformed using the two-dimensional IDWT as described in Annex F of Rec. ITU-T T.800 | ISO/IEC 15444-1 (with extensions specified in Annexes G and H). The result of this inverse transform process is the two-dimensional array $I(z, 0)$, where z = [Omcc^i, Omcc^i + 1, ... , Omcc^i + M – 1]. This array represents the inverse wavelet transformation of the ith component collection in the k^{th} transform stage. The values of $I(z, 0)$ are then the samples (at spatial location (x, y)) of the inverse transformed components W_j, where j = [0, 1, ... , M – 1]. Specifically, $W_j(x, z) = I(j + \mathrm{Omcc}^i, 0)$.

J.3.2.2 Forward multi-dimensional wavelet transformation (informative)

In what follows, we assume a specific component collection, i, to be transformed by the forward multiple component wavelet transform. The components in this collection are denoted by W_j, $j = [0, 1, \ldots, M-1]$. These components are transformed to obtain a collection of components, C_j, $j = [0, 1, \ldots, N-1]$. For the wavelet decorrelation transform, it is required that $N = M$.

The components of this collection are transformed spatial location by spatial location. This transformation process is one-dimensional. However, it can be described using the two-dimensional wavelet transformation of Annex F of Rec. ITU-T T.800 | ISO/IEC 15444-1 with modifications and extensions as noted in Annex H of this Recommendation | International Standard.

The samples of component W_j are denoted as $W_j(x, y)$. At each spatial location (x, y), a two-dimensional array is formed. This array is constructed as $I(z + \text{Omcc}^i, 0) = W_z(x, y)$, where $z = [0, 1, \ldots, M - 1]$. This array is one sample high, thus it is effectively a one-dimensional array. However, it may be treated as two-dimensional for purposes of a compact description of the multiple component wavelet transform.

The transform procedure FDWT (from Annex F of Rec. ITU-T T.800 | ISO/IEC 15444-1 with modifications and extensions specified in Annexes G and H) is applied to the array $I(z, 0)$. For this purpose, $I(z, 0)$ should be treated as a tile-component with upper left corner at $(tcx_0, tcy_0) = (\text{Omcc}^i, 0)$ and lower right corner at $(tcx_1 - 1, tcy_1 - 1) = (\text{Omcc}^i + M - 1, 0)$.

Using the notation of Rec. ITU-T T.800 | ISO/IEC 15444-1, the output of the FDWT procedure is the $N_L + 1$ sub-bands $a_{N_L\text{LL}}$, $a_{N_L\text{HL}}$, $a_{(N_L-1)\text{HL}}$, ... , $a_{1\text{HL}}$ (All sub-bands of the form a_{levHH} and a_{levLH} are empty.) Here, N_L is the number of decomposition levels to be applied. This value would be placed in the Tmcc^i field for the ith component collection in the current (i.e., kth) transformation stage.

Each such sub-band has samples $a_b(u, 0)$, $tbx_0 \le u < tbx_1$ with tbx_0 and tbx_1 determined as in Annex B of Rec. ITU-T T.800 | ISO/IEC 15444-1. These sub-band samples are then interpreted as samples of the transformed components, C_j, $j = [0, 1, \ldots, N - 1]$, at spatial location (x,y). That is, $[C_0(x, y), C_1(x, y), \ldots, C_{N-1}(x, y)] = a_{N_L\text{LL}}(tbx_0, 0), \ldots, a_{N_L\text{LL}}(tbx_1 - 1,0), a_{N_L\text{HL}}(tbx_0, 0), \ldots, a_{N_L\text{HL}}(tbx_1 - 1,0), \ldots, a_{1\text{HL}}(tbx_0, 0), \ldots, a_{1\text{HL}}(tbx_1 - 1,0)$. In each case, tbx_0 and tbx_1 are the values applicable to the sub-band indicated in the subscript of a_b.

Annex K

Non-linear transformation

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 5444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

This annex specifies three non-linear point transformations that are used after the decoding processes and inverse multiple component transformations to map reconstructed values back to their proper range. The forward non-linear transformation may be employed by encoders prior to the application of the forward multiple component transformation in order to increase compression efficiency of linear or near-linear original image data. For example, an application capturing original image data from a sensor with a linear response may apply the forward non-linear transformation to correct the data such that the bit allocation matches visual sensitivity.

K.1 Signalling the use of the non-linear transformations

The use of the non-linear transformation is signalled in the Rsiz parameter (see clause A.2.1).

K.1.1 Decoded component reconstruction

The non-linear transformations specified in this annex are "point" transformations. These transformations are applied to each sample ("point") in a given component. The transformations do not span components like the multiple component transformations described in Annex J. They may be used, however, in conjunction with the multiple component transformations. Figure K.1 shows where in the decoder's processing chain the non-linear transformation is applied. Conversely, on the encoder side, non-linear transformations would be applied prior to decorrelation and dependency transformations and wavelet processing.

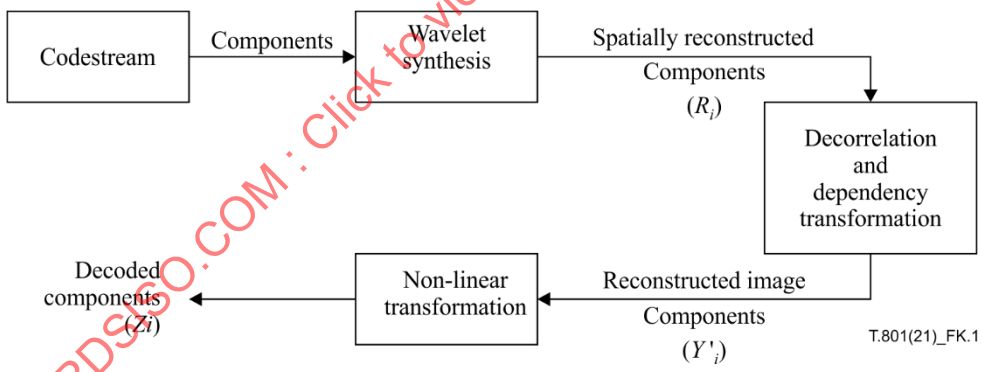

Figure K.1 – Non-linear transformation application during decoding

The forward transformation (on encoding) transforms the original image components (Z_i) into the inputs to the multiple component transformation (Y_i'). The reverse transformation (on decoding) transforms the reconstructed image components (Y_i') into the fully decoded components (Z_i).

K.1.2 Bit depth and interaction with the multiple component transformation

If a multiple component transformation is used in a codestream, the SIZ (see clause A.2.1) marker segment no longer carries the number of reconstructed image components and their bit depths. Instead, the SIZ marker segment contains the number of codestream components and their bit depths (R_i components in Figure K.1). Processing of the codestream components with the inverse decorrelation and dependency transformations may increase or decrease the number of reconstructed image components relative to the number of codestream components. Additionally, the bit depths of the reconstructed image components may be quite different from those of the codestream components. For these reasons, a CBD marker segment (see clause A.3.6) is always included in the codestream whenever a multiple component

transformation is used. The CBD marker segment contains the number of reconstructed image components and their bit depths.

If a multiple component transformation is used in conjunction with non-linear point transformations, the CBD marker segment placed in the codestream shall specify the number of reconstructed image components and their bit depths prior to processing through the non-linear transformation. If no multiple component transformation is used, then the SIZ marker segment shall indicate the number of reconstructed image components and their bit depths.

K.1.3 Marker interpretation

Regardless of the use of the multiple component transformation, an NLT marker segment (see clause A.3.10) is placed in the codestream whenever a non-linear transformation is used. The NLT marker segment indicates the bit depths of the decoded components that result from the application of a non-linear transformation to a reconstructed image component. Those reconstructed image components that do not undergo a non-linear transformation shall have no change in their bit depth.

K.2 Non-linear transformation specifications

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 clause 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 clause 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 clause O.6 for further information and its recommended use. The transformation itself is specified in clause K.2.3.

K.2.1 Gamma-style non-linearity

The gamma-style transformation specifies the non-linear transformation using mathematical equations. That transformation is specified using two functional segments: a linear region for small component values, and a power-law exponential region for large component values. The actual transformation stored within the NLT marker segment is normalized such that the maximum value of the input and output components is +1.

K.2.1.1 Forward gamma-style non-linearity (encoding, informative)

The relationship between a normalized value z of original input component Z_i and normalized value y′ of the gamma-adjusted component Y_i' is given by Equation K-1. An encoder using the non-linear transformation extension shall use the Y_i' component as input to the forward multiple component transformation, or directly to the forward wavelet transformation if the multiple component transformation is not used. Values z and y' are normalized such that the maximum value of the Z_i and Y_i' components, respectively, are +1.

$$y' = \begin{cases} -(A|z|^E - B) & z < -\frac{T}{S} \\ Sz & -\left(\frac{T}{S} \le z \le \frac{T}{S}\right) \\ Az^E - B & z > \frac{T}{S} \end{cases} \quad \text{(K-1)}$$

In Equation K-1 S is the toe-slope, T is the toe-slope threshold, E is the gamma exponent, and A and B are continuity parameters. These parameters are stored in the NLT marker segment in normalized form such that the maximum value of the input and output components is +1.

Figure K.2 shows an example forward gamma non-linearity transformation that might be used by an encoder. This particular gamma-style transformation is from Rec. ITU-R BT.709 (HDTV).

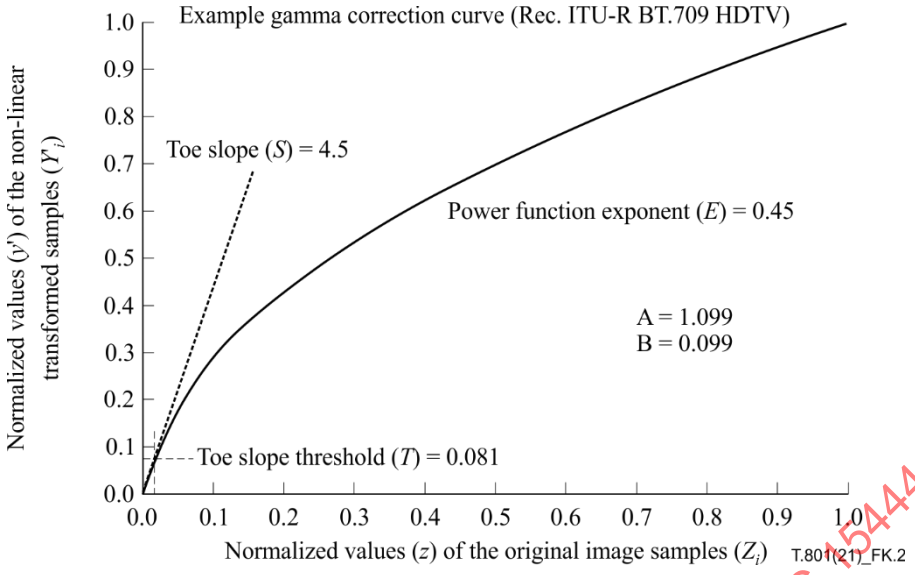

Figure K.2 – Example gamma-type forward non-linear transformation

K.2.1.2 Reverse gamma-style non-linearity (normative)

When decoding a codestream, a conforming reader shall apply the transformation specified by Equation K-2 to any component that indicates, through the NLT marker segment, that it should be processed using a gamma-style non-linearity. These equations take a normalized value y' value of component Y_i' as input and produce a normalized value z of the fully decoded Z_i component as output. The bit depth of the non-normalized Y_i' values is specified by the CBD marker segment, or by the SIZ marker segment if the multiple component extension is not used. The bit depth of the non-normalized Z_i values is specified by the BDnlt field in the NLT marker segment.

$$z = \begin{cases} -\left(\frac{|y'|+B}{A}\right)^{\frac{1}{E}} & y' < -T \\ \frac{y'}{S} & -T \le y' \le T \\ \left(\frac{y'+B}{A}\right)^{\frac{1}{E}} & y' > T \end{cases} \tag{K-2}$$

The parameters T, S, E, A, and B are all communicated in the NLT marker segment. The transformation may be applied to the non-normalized component values by first scaling the coefficients of Equation K-2 to match the actual bit depths of the Z_i and Y_i' components. To scale the normalized coefficients to non-normalized form to convert an input component Y_i' of bit depth b_y directly to an output component Z_i of bit depth b_z without first normalizing the Y_i' component values and then denormalizing the Z_i values, the coefficients shall be transformed as follows in Equations K-3 and K-4.

$$\begin{aligned} f_y &= \begin{cases} 2^{b_y} - 1 & Y_i' \text{ is unsigned} \\ 2^{b_y} - 1 & Y_i' \text{ is signed} \end{cases} \\ f_z &= \begin{cases} 2^{b_z} - 1 & Z_i \text{ is unsigned} \\ 2^{b_z} - 1 & Z_i \text{ is signed} \end{cases} \end{aligned} \tag{K-3}$$

$$\begin{aligned} T_S &= f_y \cdot T \\ S_S &= \frac{f_y}{f_z} \cdot S \\ A_S &= \frac{f_y}{f_z^E} \cdot A \\ B_S &= f_y \cdot B \end{aligned} \tag{K-4}$$

The value b_y is taken from the SIZ or CBD marker segment as appropriate. The value b_z is signalled in the BDnlt field of the NLT marker segment. Once the coefficients have been scaled appropriately, Equation K-5 can be applied directly to the integer image data for component Y_i' and produces the integer image data for component Z_i.

$$Z_i = \begin{cases} -\left(\frac{|Y_i'|+B_S}{A_S}\right)^{\frac{1}{E}} & Y_i' < -T_S \\ \frac{Y_i'}{S_S} & -T_S \le Y_i' \le T_S \\ \left(\frac{Y_i'+B_S}{A_S}\right)^{\frac{1}{E}} & Y_i' > T_S \end{cases} \quad \text{(K-5)}$$

K.2.2 LUT-style reverse non-linearity transformation

A non-linear transfer curve can often be approximated by a piece-wise linear function. The NLT marker segment provides a mechanism to specify such a non-linearity. This method is referred to as a "LUT-style non-linearity" for two reasons. First, from the specified information it is possible to build a look-up table to perform the transformation. Second, if the number of table values specified is equal to the number of possible decoded values, then a LUT is explicitly specified. While it is possible to approximate the gamma-style non-linearities specified in clause K.2.1, this mechanism allows for other transformations that cannot be specified using a simple gamma function.

The LUT-style non-linearity requires that a list of table values be supplied. The maximum possible number of LUT values for a table to produce component Z_i from component Y_i' is $N = 2^{b_y}$, where b_y is the number of bits used to represent the component Y_i' (as specified by the CBD marker if the multiple component transformation is used, or the SIZ marker segment (see clause A.2.3) if the multiple component transformation is not used). The minimum and maximum normalized LUT input values are D_{min} and D_{max}, respectively, where normalization includes shifting and linear scaling so that the range of the Y_i' component values is 0 to +1, regardless of whether component Y_i' is signed or unsigned.

The NLT marker specifies a total of N_{points} values, where $2 \le N_{points} \le N$. Those values are evenly distributed across the input range of D_{min} to D_{max}. Table value T_k (where k = 0, 1, ... , $N_{points} - 1$) represents the output of the reverse transformation for an input value of D_k. The D_k values are implicitly determined by D_{min}, D_{max}, and N_{points} through the following equation:

$$D_k = D_{min} + k\Delta$$
$$\text{where } \Delta = \frac{D_{max} - D_{min}}{N_{points} - 1}$$
$$\text{and } D_{max} > D_{min} \quad \text{(K-6)}$$

The actual LUT table values are specified in a normalized, shifted form, with the range of both the input and output components being 0 to +1, regardless of whether the components are signed or unsigned.

Values of the Y_i' input component (y') are normalized as follows before being processed through the LUT:

$$y'_{norm} = \begin{cases} \frac{y'}{2^{b_y}-1} & Y_i' \text{ is unsigned} \\ \frac{y'+2^{b_y-1}}{2^{b_y}-1} & Y_i' \text{ is signed} \end{cases} \quad \text{(K-7)}$$

If $y'_{norm} < D_{min}$, then y'_{norm} shall be clipped to D_{min}. If $y'_{norm} > D_{max}$, then y'_{norm} shall be clipped to D_{max}. The normalized values of the output component Z_i are then specified by the following equation:

$$z_{norm} = t_k + \left(\frac{y'_{norm} - D_k}{\Delta}\right)(t_{k+1} - t_k)$$
$$t_k = \frac{T_k}{2^{b_t}-1} \quad \text{(K-8)}$$

where k is the first integer such that $D_k \le y'_{norm} < D_{k+1}$. The final non-normalized value of the output component Z_i is specified by the following equation:

$$z = \begin{cases} z_{norm} \cdot (2^{b_z} - 1) & Z_i \text{ is unsigned} \\ z_{norm} \cdot (2^{b_z} - 1) - 2^{b_z-1} & Z_i \text{ is signed} \end{cases} \quad \text{(K-9)}$$

where b_z is the bit depth of the output component Z_i and is signalled in the BDnlt parameter of the NLT marker segment. The parameters, N_{points}, D_{min}, D_{max}, the array of T_k values and b_t are signalled in the NLT marker segment. b_t is stored in the NLT marker segment in the PTval field. The array of T_k values is stored in the Tvalues field.

K.2.3 Binary complement to sign-magnitude conversion transformation

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 ISO/IEC/IEEE 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 1 – 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 ISO/IEC/IEEE 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 clause 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 $BDcbd_i$ if a multi-component transformation is run (see clause A.3.6, Table A.31), or shall be equal to $Ssiz_i$ (see Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.11) if no such transformation is used.

Let Z_i be the input sample value of the component i to which this transformation is to be applied, b_i its bit depth (i.e., $b = (Ssiz_i \text{ AND } 0x7f) + 1$ or $b_i=(BDcbd_i \text{ AND } 0x7f) + 1$, and 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 2 – 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 Rec. ITU-T 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 clause O.6.

Annex L

Region of interest coding and extraction, extensions

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see clause A.2.1).

This annex describes the region of interest (ROI) technology. An ROI is a part of an image that is encoded with higher fidelity than the rest of the image (the background). The encoding is also done in such a way that the information associated with the ROI precedes the information associated with the background. The method used (and described in this annex) is the Scaling based method.

L.1 Decoding of ROI

The procedure specified in this clause is applied only in the case of the presence of an RGN marker segment, see clause A.2.5 (indicating the presence of an ROI coded with the Scaling based method).

The procedure realigns the significant bits of ROI coefficients and background coefficients. It is defined using the following steps:

1) Get the corresponding shape information and the scaling value, *s*, from the RGN marker segment for each ROI. The following steps 2-6 are applied to each coefficient (*u*, *v*) of sub-band *b*.
2) Generate the ROI mask $\{M_i(u, v)\}$ for all ROI, see L.3 for details on how to generate the ROI mask.
3) For each coding block find the largest scaling value, s_{max} for any coefficient (*u*, *v*).
4) For each coefficient in each coding block find the highest scaling value and set *s*(*u*, *v*) to:

$$s(u,v) = s_{Max} - max(s_i \cdot M_i(u,v)) \quad \text{(L-1)}$$

where i = *0... Number of ROI*–1

5) For each coefficient (*u*, *v*) discard the first *s(u, v)* MSBs and shift up the remaining MSBs *s(u, v)* places, as described in Equation L-2, for *i* = 1, ... , M_b

$$MSB_i(b,u,v) = \begin{cases} MSB_{i+s(u,v)}(b,u,v) & \text{if } i + s(u,v) \le N_b(u,v) \\ 0 & \text{if } i + s(u,v) > N_b(u,v) \end{cases} \quad \text{(L-2)}$$

6) Update the value of N_b(*u*, *v*) as given in Equation L-3.

$$N_b(u,v) = max(0, N_b(u,v) - s(u,v)) \quad \text{(L-3)}$$

L.2 Description of the Scaling based method

This clause describes how to encode an image with one or more ROI. The encoding is given here as an informative section. However, failure to generate the correct ROI mask at the encoder side will greatly reduce the quality of the decoded image and will not allow lossless decoding.

L.2.1 Encoding with ROI (informative)

At the encoder side an ROI mask is created describing which quantized transformation coefficients are encoded with better quality (up to lossless). The ROI mask is a bit map describing these coefficients for one ROI. See L.3 for details on how the mask is generated.

The quantized transformation coefficients are scaled in such a manner that the relative significance of each transformation coefficient is equal to the specified scaling value, s, of the ROI to which it applies. If a transformation coefficient belongs to several ROI the largest *s* value is chosen. If a transformation coefficient belongs to the Background, the scaling value *s* equals 0. Before scaling the quantized transformation coefficients of one code-block, the highest, s_{Max}, and lowest, s_{Min}, scaling value for the coding block are found.

Consider a quantized transformation coefficient, $q_b(u,v)$, in the current coding block with corresponding scaling value, *s* (where $s_{Min} \le s \le s_{Max}$). After scaling, the individual bits of $q_b(u, v)$ end up $abs(s_{Max}–s)$ bit planes lower than the

corresponding bits of a coefficient with $s = s_{Max}$. The number of magnitude bits for this coding block will hence increase by $(s_{Max}–s_{Min})$.

Since the coding blocks are treated independently, quantized transformation coefficients belonging to the same ROI might end up having different levels of significance in different coding blocks. This difference between coding blocks are taken care of by the rate allocator. An example of this would be if an entire coding block belongs to the image background and another coding block has both ROI and background coefficients. In this case, the background coefficients in the second coding block would be downshifted by $s–0$ steps whereas in the first coding block no shifting is done. In this case it is up to the rate allocation algorithm to make sure that the bit planes from the two coding blocks are put in the bit stream in the correct order.

When the entropy coder encodes the quantized transformation coefficients, the bit planes associated with the ROI are coded before or at the same time as the information associated with the background. The scaling value, s_i, for each ROI is specified by the user/application.

The method can be described using the following steps for a set of n ROI:

– For each coding block in each component:

1) Generate ROI mask for all ROI i, $\{M_i(u, v)\}$, see L.3.

2) Find s_{Min} and s_{Max}, where s_{Min} and s_{Max} are the smallest and largest scaling value in the current coding block, respectively.

3) Add $s_{Block} = s_{Max}–s_{Min}$ LSBs to each coefficient $|q_b(u, v)|$. The number M_b' of magnitude bit-planes for sub-band, b, will then be:

$$M_b' = M_b + s_{block} \tag{L-4}$$

where M_b is given by Rec. ITU-T T.800 | ISO/IEC 15444-1, Equation E-2, and the new value of each coefficient is given by:

$$|q_b(u,v)| = |q_b(u,v)| - 2^{s_{Block}} \tag{L-5}$$

4) For each coefficient in each coding block find the highest scaling value and set $s(u, v)$ to:

$$s(u,v) = s_{Max} - max(s_i - M_i(u,v)) \tag{L-6}$$

where i = *0... Number of ROI*–1

5) Scale down all coefficients so that:

$$|q_b(u,v)| = \frac{|q_b(u,v)|}{2^{s(u,v)}} \tag{L-7}$$

6) For each ROI write the scaling value, s, shape, and reference points into the codestream using the RGN marker segment as described in clause A.2.5.

L.3 Region of interest mask generation

To achieve an ROI with better quality than the rest of the image while maintaining a fair amount of compression, bits need to be saved by sending less information for the background. To do this an ROI mask is calculated. The mask is a bit-plane indicating a set of quantized transformation coefficients whose coding is sufficient in order for the receiver to reconstruct the desired region with better quality than the background (up to lossless).

To illustrate the concept of ROI mask generation, let us restrict ourselves to a single ROI and a single image component, and identify the samples that belong to the ROI in the image domain by a binary mask, $M(x, y)$, where:

$$M(u,v) = \begin{cases} 1 & \text{wavelet coefficient } (u,\ v) \text{ is needed} \\ 0 & \text{accuracy on } (u,v) \text{ can be sacrified without affecting ROI} \end{cases} \tag{L-8}$$

The mask is a map of the ROI in the wavelet domain so that it has a non-zero value inside the ROI and 0 outside. In each step, each sub-band of the mask is then updated line by line and then column by column. The mask will then indicate which coefficients are needed at this step so that the inverse transformation will reproduce the coefficients of the previous mask.

For example, the last step of the inverse transformation is a composition of two sub-bands into one. Then to trace this step backwards, the coefficients of both sub-bands that are needed are found. The step before that is a composition of four sub-bands into two. To trace this step backwards, the coefficients in the four sub-bands that are needed to give a perfect reconstruction of the coefficients included in the mask for two sub-bands are found.

All steps are then traced backwards to give the mask. If the coefficients corresponding to the mask are transmitted and received, and the inverse transformation calculated on them, the desired ROI will be reconstructed with better quality than the rest of the image (up to lossless if the ROI coefficients were coded losslessly).

Given below are descriptions of how the expansion of the mask is acquired in the rectangular and elliptic case and also how this is done for the various filters. Similar methods can be used for other filters.

L.3.1 Rectangular mask generation on the reference grid

The rectangular mask described in this clause is generated on the reference grid. When generated on the reference grid the methods described in clauses L.3.4 and L.3.5 are used for mask generation in the wavelet domain. A rectangle is described by four parameters, see Figure L.1, all signalled in the RGN marker, see clause A.2.5. The parameters are (*XArgn*, *YArgn*, *XBrgn*, *YBrgn*) where *XArgn* and *YArgn* are the *x* offset and *y* offset of the upper left corner of the rectangle from the reference grid origin, respectively; *YBrgn* is the height of the rectangle; *XBrgn* is the width of the rectangle.

The correct mask for the reference grid is given by Equation L-9.

$$XArgn \le x < XArgn + XBrgn$$
$$YArgn \le y < YArgn + YBrgn \qquad \text{(L-9)}$$

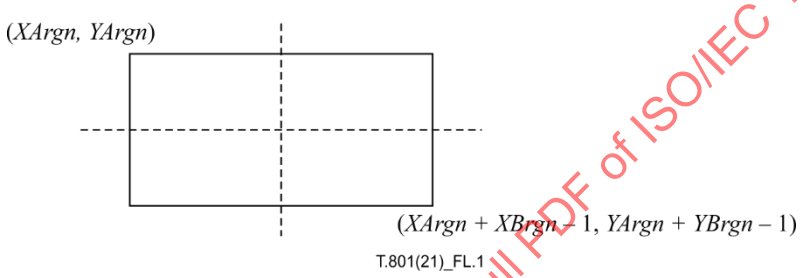

Figure L.1 – Rectangular mask on the reference grid

L.3.2 Elliptic mask generation on the reference grid

The elliptic mask described in this clause is generated on the reference grid. When generated on the reference grid, the methods described in clauses L.3.4 and L.3.5 are used for mask generation in the wavelet domain. An ellipse is described by four parameters, see Figure L.2, all signalled in the RGN marker, see clause A.2.5. The parameters are (*XArgn*, *YArgn*, *XBrgn*, *YBrgn*) where *XArgn* and *YArgn* are the *x* offset and *y* offset of the center of the ellipse from the reference grid origin, respectively; *YBrgn* is the height of the ellipse; *XBrgn* is the width of the ellipse.

The correct mask for the reference grid is given by Equation L-10.

$$YBrgn^2 \cdot (x - XArgn)^2 + XBrgn^2 \cdot (y - YArgn)^2 \le XBrgn^2 \cdot YBrgn^2 \qquad \text{(L-10)}$$

Thus, a coordinate on the reference grid belongs to the ROI if and only if Equation L-10 is true.

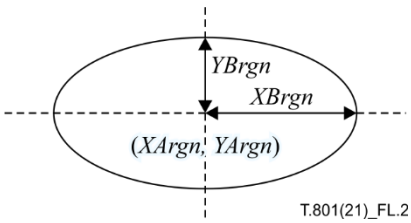

Figure L.2 – Elliptic mask on the reference grid

L.3.3 Region of Interest mask generation of whole-sample symmetric filter banks

The whole-sample symmetric filter banks are a subset of the arbitrary optional filter banks, the ROI mask generation can be described in the same way as for the arbitrary optional filter banks, see clause L.3.4. When using whole-sample symmetric, use the parameters defined in clause H.1.2.

L.3.4 Region of Interest mask generation of arbitrary optional filter banks

The generation of the ROI mask follows the arbitrary decomposition of tile-components as described in Annex F. However, instead of decomposing a tile-component, an ROI mask is decomposed. An ROI mask for the tile-component is derived first, starting from the ROI mask defined on the reference grid. The tile-component mask for component i has non-zero values at tile-component locations (x, y) for which the corresponding reference grid location $(x \cdot XRSiz^i,\ y \cdot YRSiz^i)$ belongs to the ROI and has zero values otherwise.

Instead of calculating the wavelet coefficients using the lifting steps, as described in Annex H, the lifting steps for the inverse discrete wavelet transformation are followed in reverse order and in each lifting step the wavelet coefficients that are used to reconstruct wavelet coefficients that correspond to non-zero samples in the ROI mask are found. For each lifting step, the ROI mask is updated so that all samples that correspond to wavelet coefficients that would have been used to reconstruct the wavelet coefficients that correspond to non-zero samples in the ROI mask are set to non-zero values. The mask generation shall take into account what type of filter that has been used by the transformation and also whether or not the SSO transformation has been used. The different cases are described below.

This is done by replacing Equations H-4, H-5 and H-7 by:

Let the 1D mask, to be decomposed, be R_{ext}.

For each lifting step s where s ranges from 0 to $N_{LS} - 1$,

a)

if$(R_{ext}(2n + m_s) == 1)$
then
$\left(R'_{ext}(2n + 1 - m_s + 2(k + off_s))\right) = 1$, for all $k = 0, \ldots, L_s - 1$
and

$$R'_{ext}(2n + m_s) = 1 \tag{L-11}$$

where R_{ext} are the samples in the ROI mask, $M_i(u,v)$ corresponding to V_{ext} in Equations H-4, H-5 and H-7. R'_{ext} is R_{ext} after lifting step s. Moreover, $m_s = 1 - m_{s-1}$ indicates whether the sth lifting step applies to even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), and where L_s is the number of lifting coefficients for lifting step s.

b)

$$R_{ext} = R'_{ext} \tag{L-12}$$

L.3.4.1 Single sample overlap

This is done by replacing Equations I-12 and I-14 by:

For each lifting step s where s ranges from 0 to $N_{LS} - 1$,

a)

for all n except n_p, $p = 0,1, \ldots, N_I$:
if $(R(2n + m_s) == 1)$:
$R'(PSE_{O,p}(2n + m_s - (2k + 1))) = R'(PSE_{O,p}(2n + m_s + (2k + 1))) = 1$,for$k = 0, \ldots, L_s - 1$
for all n_p, $p = 0,1, \ldots, N_I$:
if $(R(n_p) == 1)$:
$R'(n_p) = 1$ (L-13)

where R are the samples in the ROI mask, $M_i(u,v)$ corresponding to V in Equations I-12 and I-14. R' is R after lifting step s. Moreover, $m_s = 1 - m_{s-1}$ indicates whether the sth lifting step applies to even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), and where L_s is the number of lifting coefficients for lifting step s. N_I, p, n_p and $PSE_{O,p}()$ are defined as in Annex I.

b)

$$R = R' \tag{L-14}$$

After doing this for all the lifting steps, the ROI mask samples are separated into sub-bands the same way as the wavelet coefficients are separated in using the deinterleave procedure described in clause F.4.5 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

This ensures that each coefficient that has affected a coefficient in the ROI during the inverse wavelet transformation will have a '1' in the corresponding place in the ROI mask

L.3.5 Fast generation of a rectangular mask (informative)

In the case of a rectangular ROI, the mask can be derived more quickly than for arbitrary shapes. In this case, instead of tracing how each coefficient and pixel value is reconstructed in the inverse transform, only two positions need to be studied, namely the upper left and the lower right corners of the mask. The top-left corner, (x_1, y_1) on the reference grid will be given in the RGN marker segment as XArgn, YArgn, and the bottom right corner (x_2, y_2), on the reference grid will be given by the parameters in the RGN marker segment as (XArgn + XBrgn–1), (YArgn + YBrgn–1), respectively (see clause A.3.8). The mask generation shall take into account what type of filter that has been used by the transform. This can be combined with the single sample overlap transform.

In each level of decomposition, the steps described in the previous subclause are followed to see how the mask expands.

Let the 1D mask, to be decomposed, be R_{ext}, and let x_1 and x_2 and be the lowest and highest indices of non-zero samples in R_{ext}.

1) For each lifting step s where s ranges from 0 to $N_{LS} - 1$,

   a) Find the lowest sample index ($2n + m_s \geq x_1$) that is in the mask

$$x_1' = 2n + 1 - m_s + 2off_s \tag{L-15}$$

$$\text{if}(x_1' > x_1): \quad x_1' = x_1 \tag{L-16}$$

   b) Find the highest sample index $2n + m_s \leq x_2$

$$x_2' = 2n + 1 - m_s + 2(L_s - 1 + off_s) \tag{L-17}$$

$$\text{if}(x_2' > x_2): \quad x_2' = x_2 \tag{L-18}$$

   c) Set $x_1 = x_1', \quad x_2 = x_2'$

   where $m_s = 1 - m_{s-1}$ indicates whether the sth lifting step applies to even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), and where L_s is the number of lifting coefficients for lifting step s.

Let all samples between x_1 and x_2, inclusive, be non-zero and then separate the ROI mask samples into sub-bands the same way as the wavelet coefficients are separated in using the deinterleave procedure described in clause F.4.5 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

L.3.5.1 Single Sample Overlap

In each level of decomposition, the steps described in clause L.3.4.1 are followed to see how the mask expands.

Let the 1D mask, to be decomposed, be R and let x_1 and x_2 be the lowest and highest indices of non-zero samples in R.

1) For each lifting step s where s ranges from 0 to $N_{LS} - 1$,

   a) Find the lowest sample index ($2n + m_s \geq x_1$) that is in the mask

$$\text{if} x_1 = n_p, p = 0,1,\ldots,N_I: \quad x_1' = x_1 \quad else$$
$$x_1' = min(PSE_{O,p}(2n + m_s - (2k + 1))), k = 0,\ldots,L_s - 1 \tag{L-19}$$

$$\text{if}(x_1' > x_1): \quad x_1' = x_1 \tag{L-20}$$

   b) Find the highest sample index $2n + m_s \leq x_2$

$$\text{if} x_2 = n_p, p = 0,1,\ldots,N_I: \quad x_2' = x_2 \quad else$$
$$x_2' = max(PSE_{O,p}(2n + m_s + (2k + 1))), k = 0,\ldots,L_s - 1 \tag{L-21}$$

$$\text{if}(x_2' < x_2): \quad x_2' = x_2 \tag{L-22}$$

c) Set $x_1 = x_1'$, $x_2 = x_2'$

where $m_s = 1 - m_{s-1}$ indicates whether the sth lifting step applies to even-indexed coefficients ($m_s = 0$) or odd-indexed coefficients ($m_s = 1$), and where L_s is the number of lifting coefficients for lifting step s, and where off_s is the offset for lifting step s. N_I, p, n_p and $PSE_{O,p}()$ are defined as in Annex I.

Let all samples between x_1 and x_2, inclusive, be non-zero and then separate the ROI mask samples into sub-bands the same way as the wavelet coefficients are separated in using the deinterleave procedure described in clause F.4.5 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

L.4 Remarks on region of interest coding

L.4.1 Usage together with Maxshift method described in ITU-T T.800 | ISO/IEC 15444-1

The Maxshift method described in ITU-T T.800 | ISO/IEC 15444-1 shall not be used together with the method described in this Recommendation | International Standard.

L.4.2 Multi-component remark (informative)

For the case of colour images, the method applies separately in each colour component. If some of the colour components are down-sampled, the mask for the down-sampled components is created in the same way as the mask of the non-down-sampled components.

L.4.3 Implementation Precision remark (informative)

This ROI coding method might in some cases create situations where the dynamic range is exceeded. This is however easily solved by simply discarding the least significant bit planes that exceed the limit due to the downscaling operation. The effect will be that the ROI will have better quality compared to the background, even though the entire bit stream is decoded. It might however create problems when the image is coded with ROI's in a lossless mode. Discarding least significant bit-planes for the background might have the result that the background is not coded losslessly; and in the worst case the background may not be reconstructed at all. This depends on the dynamic range available.

Annex M

JPX extended file format syntax

(This annex forms an integral part of this Recommendation | International Standard.)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to Rec. ITU-T T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard.

M.1 File format scope

This annex defines an optional file format that applications may choose to use to contain JPEG 2000 compressed image data. This format is an extension to the JP2 file format defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex I. While not all applications will use this format, many applications will find that this format meets their needs. However, those applications that do implement this file format shall implement it as described in this entire annex.

This annex:

- specifies a binary container for both image and metadata;
- specifies a mechanism to indicate image properties, such as the tonescale or colourspace of the image;
- specifies a mechanism by which readers may recognize the existence of intellectual property rights information in the file;
- specifies a mechanism by which metadata (including vendor specific information) can be included in files specified by this Recommendation | International Standard;
- specifies a mechanism by which multiple codestreams can be combined into a single work, by methods such as compositing and animation.

M.2 Introduction to JPX

As defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Annex I, the JP2 file format provides a method by which applications can interchange images files in such a way that all conforming readers can properly interpret and display the image. However, some applications require extensions to the JP2 file format that would prevent the file from being properly interpreted by a conforming reader. For example, an image encoded in a CMYK colourspace will not be properly interpreted by a conforming JP2 reader.

Placing these non-compatible extensions into a JP2 file will introduce confusion in the marketplace, as a situation will exist where some readers can interpret some JP2 files but not others. While this confusion is inevitable when you consider the global set of all applications profiles, it shall be avoided in some applications, such as on the consumer desktop.

Thus this annex defines a second file format to be used in applications that require functionality or data structures beyond those defined in the JP2 file format. This file format is called JPX.

M.2.1 File identification

JPX files can be identified using several mechanisms. When stored in traditional computer file systems, JPX files should be given the file extension ".jpf" (readers should allow mixed case). On Macintosh file systems, JPX files should be given the type code 'jpx\040'.

However, if a particular JPX file is compatible with the JP2 reader specification (as indicated by placing the code 'jp2\040' in the compatibility list in the File Type box), then the writer of that file may choose to use the extensions for the JP2 file format, as specified in clause I.2.1 of the JP2 file format, for that particular file. This will maximize the interoperability of that file without sacrificing file indication (as the BR field in the File Type box shall be 'jpx\040' for files completely defined by this Recommendation | International Standard).

In the JP2 file format, the File Type box provides information that a file reader can use to determine if it is capable of reading the file. This box is also present in other JPEG 2000 family file formats.

M.2.2 File organization

As in JP2 files, a JPX file represents a collection of boxes. The binary structure of a file is a contiguous sequence of boxes. The start of the first box shall be the first byte of the file, and the last byte of the last box shall be the last byte of the file.

Many boxes are defined by this Recommendation | International Standard. In addition, other Recommendations | International Standards may define other boxes for use within JPX files. However, all information contained within a JPX file shall be in the box format; byte-streams not in the box format shall not be found in the file.

The binary structure of a box in a JPX file is identical to that defined in the JP2 file format (Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause I.4).

M.2.3 Greyscale/Colour/multi-component specification

The JP2 file format allowed the colourspace of the image to be specified in two ways (Enumerated or Restricted ICC methods). The JPX file format expands on this by:

- defining additional colourspaces for the Enumerated method (clause M.11.3.2);
- defining a method for vendors and other standards bodies to define additional Enumerated colourspaces (clause M.7.3);
- defining a new method to allow the use of any input ICC profile (the Any ICC method, clause M.3.3);
- defining a new method to allow vendors to define unique codes for colourspaces without consultation with a third-party (the Vendor Colour method, clause M.11.7.3);
- allowing the use of extensions of the multiple component transformation and non-linearity transformation extensions within the codestream (see Annexes J and K).

M.2.4 Specification of opacity information

The JPX file format specifies two extensions for specifying opacity information. First, the file format allows for opacity channels to be stored in a separate codestream from the colour channels of the image. In many image editing applications, the colour data and the opacity data are edited separately, and thus it is useful to allow those channels to be stored in separate codestreams. This is described in the compositing layer architecture description in clause M.5.

Secondly, the JPX file format allows for the specification of fully transparent samples through a chroma-key. The file format specifies an array of sample values, one from each colour channel. Image locations that contain that combination of sample values shall be considered fully transparent. For example, the chroma-key may be specified as red = 134, green = 92 and blue = 47. Any location with this combination of red, green and blue sample values shall be considered fully transparent. This is defined in the specification of the Opacity box in clause M.11.7.6.

M.2.5 Metadata

In addition to specifying how the image data shall be stored, this Recommendation | International Standard defines, in Annex M, a number of metadata elements. These elements specify information such as how the image was created, captured or digitized, or how the image has been edited since it was originally created. This Recommendation | International Standard also includes the ability to specify intellectual property rights information, as well as the content of the image, such as the names of the people and places in the image.

In addition to the metadata defined within this Recommendation | International Standard, other forms of XML-based metadata and descriptions may be embedded in a JPX file within XML boxes, such as those defined in Annex N.

Furthermore, the MPEG-7 binary box (as defined in clause M.11.19) may be used to store MPEG-7 binary (BiM) format metadata.

M.2.6 Storage of a codestream within JPX

In JP2, the entire codestream is required to be stored in a contiguous portion of the file. However, this restriction can be problematic for some applications. Image editing applications, for example, may desire to modify a single tile of the image and to write the modified tile to the end of the file without rewriting the file. Image servers or internet applications may desire to split the image up into multiple files on different disks or spread the codestream across the internet. The JPX file format allows these features by allowing the codestream to be divided into fragments. The fragmentation of codestreams is described in clause M.4.

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.

M.2.7 Combining multiple codestreams

In addition to specifying the rendered result as a result of decompressing a single codestream and properly interpreting the colourspace of that codestream as specified in the JP2 file format, the JPX file format allows for multiple codestreams to be combined to produce the rendered result. These codestreams can be combined in a combination of two ways: compositing and animation. This is further described in clause M.5.

M.2.8 Support for various pixel formats

In Rec. 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 clause 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 Rec. ITU-T 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, ISO/IEC/IEEE 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 clause 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, ISO/IEC/IEEE 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 clause M.11.5.1) if the channel depth is identical for all channels, or in the Bits Per Component box (see clause 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 I.5.3.4 of Rec. ITU-T 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 ISO/IEC/IEEE 60559.

Fixed point and floating point samples are mapped to device colours by means of the Colour Specification box (see clause M.11.7.2) in the same way as integer samples are mapped to device colours. For that, the Channel Definition box (see clause 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 clause M.11.7.2.

M.2.9 Support for JPEG XR codestreams

This Recommendation | International Standard also allows the encapsulation of codestreams of other image compression Recommendations | Standards, such as JPEG XR (Rec. ITU-T 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 clause O.5.

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

Table M.1bis – Brand Values for JPEG XR Codestreams

Value	Meaning
'jxrc'	JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) compliant bitstream is present
'jxr0'	JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) sub-baseline profile bitstream present and the file conforms to the JPEG XR sub-baseline profile defined in clause M.9.2.
'jxr1'	JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) baseline profile bitstream present and the file conforms to the JPEG XR baseline profile defined in clause M.9.2.
'jxr2'	JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) main profile bitstream present and the file conforms to the JPEG XR main profile defined in clause M.9.2.
'jxr3'	JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-3) advanced profile bitstream and the file conforms to the JPEG XR advanced profile defined in clause M.9.2.
NOTE – Brand values 'jxr0' to 'jxr3' indicate JPEG XR profiles of this Recommendation | International Standard. The corresponding profiles are defined in clause M.9.2.	

M.3 Greyscale/Colour/Palette/multi-component specification architecture

The JPX file format builds on the flexible colour architecture defined in the JP2 file format. Colourspace specifications are specified within Colour Specification boxes, as originally defined in JP2. However, JPX extends this box (within the binary structure limitations defined in JP2) to allow other methods to be used to specify the colourspace, and to allow a reader to select from the different colourspace specifications found in a single file when interpreting an image.

M.3.1 Extensions to the Colour Specification box header

In JP2, the APPROX and PREC fields were reserved for future use; JP2 writers are required to write default values into these fields. In JPX, these fields are defined and can be used by a JPX reader to make intelligent processing choices.

In both JP2 and JPX, a single file may contain multiple representations of the colourspace of the image. For example, a JPX image may contain an enumerated value, a complex ICC profile, and a Restricted ICC profile. These multiple methods are included to maximize interoperability as well as to provide optimized access. In this example, the enumerated value allows quick recognition, the complex ICC profile allows accurate interpretation using a full ICC engine, and the Restricted ICC profiles allows less complex readers to get a "good enough" result. Specifically, the Restricted ICC profile in this example is an approximation of the complex ICC profile. This approximation is specified within the Colour Specification box, and allows a reader to make trade-offs when interpreting the image.

The Colour Specification box also allows the writer to associate a precedence with each method. This information specifies a default priority in which the multiple colourspace specifications should be considered when selecting which specification will be used to interpret the decompressed code values.

However, the use of both the approximation and precedence information is beyond the scope of this Recommendation | International Standard. Applications are free to consider both pieces of information together and to define their own priorities for selecting which colourspace method to use when interpreting the image. For example, in a fast-preview mode, where speed is more important then quality, an application may desire to use the Restricted ICC profile even if it can use the more complex profile.

M.3.2 Extensions to the Enumerated method

The JPX format defines enumerated values for several additional colourspaces. In addition, this Recommendation | International Standard defines a mechanism by which vendors or other standards bodies can register additional values for the EnumCS field in the Enumerated Method. In general, there are no implementation requirements for these additional defined or registered colourspaces. Requirements for interpretation of specific spaces are defined within the file conformance definition.

In addition, the data structures for the enumerated method have been extended to allow for the specification of parameters that define exactly how that particular colourspace was encoded in the file. For example, the CIELab colourspace, as defined by Rec. ITU-T T.42 specifies six parameters that specify the exact encoding range and offsets of the stored data. To properly interpret a CIELab image, the decoder is required to know this information and apply those parameters to the decoded image data. Enumerated parameters are specified individually for each enumerated colourspace that requires parameters. However, many colourspaces do not require additional parameters, and thus additional parameters are not defined for those colourspaces. For colourspaces that do specify additional parameters, default values may be defined (as for example are done for the definitions of CIELab and CIEJab). If the entire block of additional parameters are not contained within the Colour Specification box, then the default values shall be used; however, if any additional parameter is specified for a particular Colour specification box, then all additional parameters shall be specified for that Colour Specification box.

M.3.3 Any ICC method

In the JP2 file format, the Restricted ICC method was defined, allowing images to be encoded in a wide range of RGB and greyscale spaces. However, many colourspaces, such as CMYK and CIELab spaces, cannot be represented using the restricted set of ICC profiles allowed by the Restricted ICC method. JPX lifts this restriction by defining a separate method to allow any legal ICC input profile to be embedded in the file. This is a separate colour method than the Restricted ICC method, which is also legal in a JPX file. Applications shall not use the Restricted ICC METH value for embedding non-Restricted ICC profiles.

M.3.4 Vendor Colour method

While this Recommendation | International Standard defines a method by which new colourspaces can be registered, the registration method is not appropriate for use for defining codes for vendor-specific or private colourspaces. To allow the quick identification of these colourspaces, the JPX standard defines an additional colourspace specification method, called the Vendor Colour method. This method is very similar to the Enumerated method, except that instead of using 4-byte integer codes, the Vendor Colour method uses universal unique identifiers (UUIDs). These UUID values are generated by application developers when the definition of a particular colourspace is created.

It is legal to specify a Vendor Colour value in every JPX file. However, no reader is required to correctly interpret the image based solely on the Vendor Colour method. If an image writer desires to maximize interoperability outside the scope of the target application, it should use additional colour methods in the file (such as the Any ICC and Restricted ICC colour methods).

M.3.5 Palettized colour

Palettized colour is specified and works exactly as defined in the JP2 file format. A palettized image would contain a Palette box, which specifies the transformation from one to many components. The many components generated by the palette are then interpreted by the rest of the colour architecture as if they had been stored directly in the codestream.

M.3.6 Using multiple methods

The JPX file format allows for multiple methods to be embedded in a single file (as in the JP2 file format) and allows other standards to define extensions to the enumerated method and to define extended methods. This provides readers conforming to those extensions a choice as to what image processing path should be used to interpret the colourspace of the image.

If the file is to be JP2 compliant, the first method found in the file (in the first colourspace specification box in the JP2 Header box) shall be one of the methods as defined and restricted in the JP2 file format. However, a conforming JPX reader may use any method found in the file.

M.3.7 Interactions with the decorrelating multiple component transformation

The specification of colour within the JPX file format is independent of the use of a multiple component transformation or non-linearity correction within the codestream (the MCT, MCC, MCO and NLT markers specified in clauses A.3.7, A.3.8, A.3.9 and A.3.10, respectively). The colourspace transformations specified through the sequence of Colour Specification boxes shall be applied to the image samples after the reverse multiple component transformation and reverse non-linearity correction has been applied to the decompressed samples. While the application of these decorrelating component transformations is separate, the application of an encoder-based multiple component transformation will often improve the compression of colour image data.

M.4 Fragmenting the codestream between one or more files

Another important feature of the JPX file format is the ability to fragment a single codestream within a single file or across multiple files. This allows applications to implement such features as:

- edit an image, resaving the changed tiles to the end of the file;
- distribute the image across several disks for faster access;
- distribute the image across the internet, allowing only certain customers access to the high quality or high resolution portions of the codestream;
- reuse of headers from within a codestream across multiple codestreams (to minimize file overhead when storing similar codestreams within the same JPX file).

Fragmentation in JPX works by specifying a table of pointers to the individual fragments. Each pointer specifies three things:

- The file in which the fragment is contained. Because multiple fragments across multiple codestreams may be stored in the same file, the format encapsulates all filename/URL data into a table (the Data Reference box). Each fragment specification then references an entry in the data reference table.
- The offset of the first byte of the fragment within the file specified. This offset is with respect to the first byte of the file (byte 0) and points directly to the first byte of codestream data for that fragment; it does not point to the start of a box containing that fragment.
- The length of the fragment, in bytes.

While the fragment offset does not point to the start of the box, any codestream data contained within a JPX file shall be encapsulated in a box. If a codestream is contained within the JPX file in contiguous form, then it shall be encapsulated within a Contiguous codestream box as specified in the JP2 file format and M.11.8; the file shall not also contain a Fragment table representing that contiguous codestream. If the codestream is contained within the JPX file in multiple fragments, then the codestream shall be encapsulated within one or more Media Data boxes (defined in clause M.11.9).

Figure M.1 shows how a fragment table is used to specify a complete codestream in an example JPX file when all fragments are stored within the file itself. Because the data reference table was empty (no external references), it may not exist in the JPX file. Boxes other than the fragment related boxes are not explicitly shown.

In this example, the codestream is divided into four fragments. The dark lines on the bottom of the Media Data boxes show the portion of the contents of that Media Data box that represents the fragment. Two of those fragments are contained within the same Media Data box. For each fragment, the fragment list specifies the offset and the length of each fragment. The offset values point to the first byte of the codestream data, relative to the beginning of the file. For example, the first fragment is at the start of the contents of a Media Data box. The offset to that fragment is to the first byte of the contents of the box, not to the start of the box header. The length values specify the length only of the actual codestream data for that fragment.

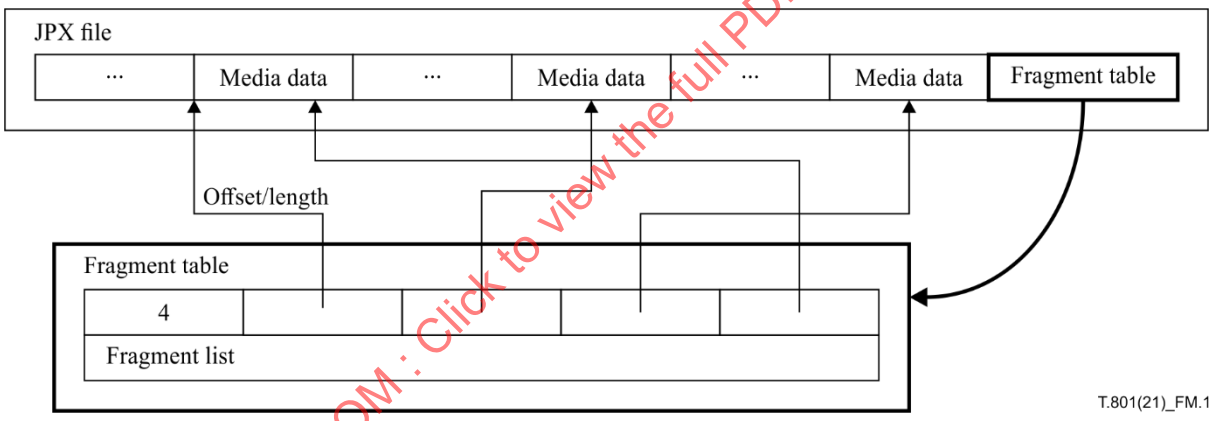

Figure M.1 – Example fragmented JPX file where all fragments are in the same file

To extract the complete codestream from the file, an application locates the fragment table for that codestream in the file, and then parses the offsets and lengths from the fragment list. The application could then simply seek to the locations specified by the offsets and read the amount of data specified by the length.

Figure M.2 shows how a fragment table is used to specify a complete codestream in an example JPX file when some of the fragments are stored outside the file. In this case, the file shall contain a Data Reference box.

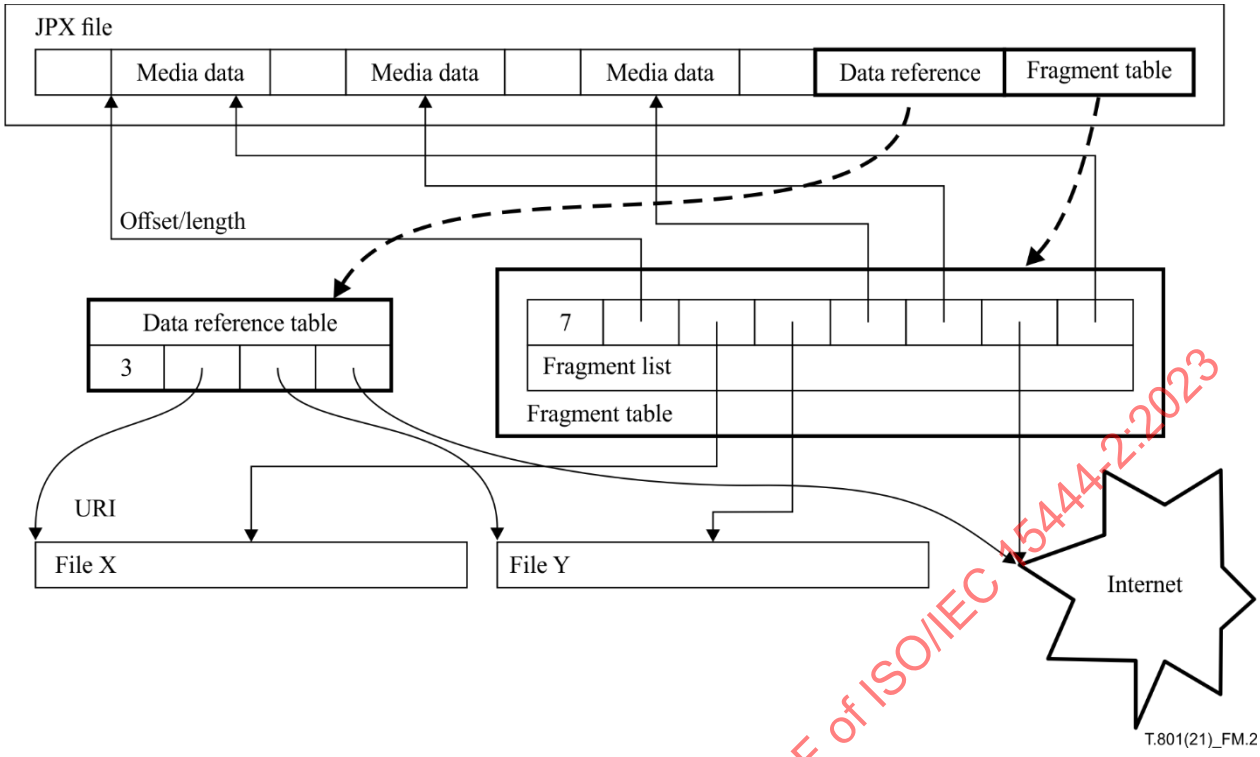

Figure M.2 – Example fragmented JPX file where some fragments are stored in other files or resources

In this example, two of the fragments are stored in separate but locally accessible files, and one of the fragments is stored across the internet.

M.5 Combining multiple codestreams

In the simplest JPX file, a rendered result is generated by decompressing a single codestream to one or more image channels and properly interpreting them in the context of the associated colourspace specification and optional opacity specification. This mode of operation is identical to that offered by JP2 except that JPX offers a wider range of colourspaces and specification methods. In addition to this, JPX offers a rich set of methods for combining multiple codestreams to form the rendered result.

In a JPX file it is possible to store multiple JP2 style "images." In the context of a single JPX file, these separate images are referred to as compositing layers. Each compositing layer comprises a set of channels that an application should treat as a unit for the purpose of rendering. The JPX file format includes syntax for specifying how the compositing layers in a file should be combined by the reader application to produce the rendered result. Both simple still image compositing and animation are supported.

For example, one layer may be a simple RGBA codestream. Another layer may consist of R, G and B channels generated by the application of a palette to one component from codestream 1, and an opacity channel directly extracted from codestream 2.

In a JPX file, it is additionally possible to store a single image (or compositing layer) using multiple codestreams. This, for example, allows the separation of RGB components from an opacity channel component. This would permit a single opacity channel to be reused in other compositing layers in the JPX file. The file format also includes syntax for specifying how codestreams are combined to form compositing layers including how the codestreams should be spatially registered against each other.

In a JPX file, metadata can be associated with codestreams and compositing layers independently. Metadata may be shared between multiple codestreams.

M.5.1 Mapping codestreams to compositing layers

To facilitate the mapping of multiple codestreams to single compositing layers, the JPX format separates header fields defined for JP2 into two logical groups: those specific to a single codestream are grouped into a Codestream Header box (M.11.6) and those specific to a compositing layer are grouped into a Compositing Layer Header box (M.11.7). The

process of mapping codestream components to channels is exemplified in Figure M.3. Multiple codestreams are combined via a Codestream Registration box (M.11.7.7) to provide the complete set of components for a compositing layer. A Component Mapping box (Rec. ITU-T T.800 | ISO/IEC 15444-1, clause I.5.3.5) in the Codestream Header box is used to specify how the components of any given codestream are mapped to channels. Interpretation of these channels is specified in the Compositing Layer Header box either using a Channel Definition box (M.11.7.5) or an Opacity box (M.11.7.6). The Opacity box is a new option in the JPX file format that provides an additional method for specifying compositing layers with simple compositing or that use chroma-key opacity.

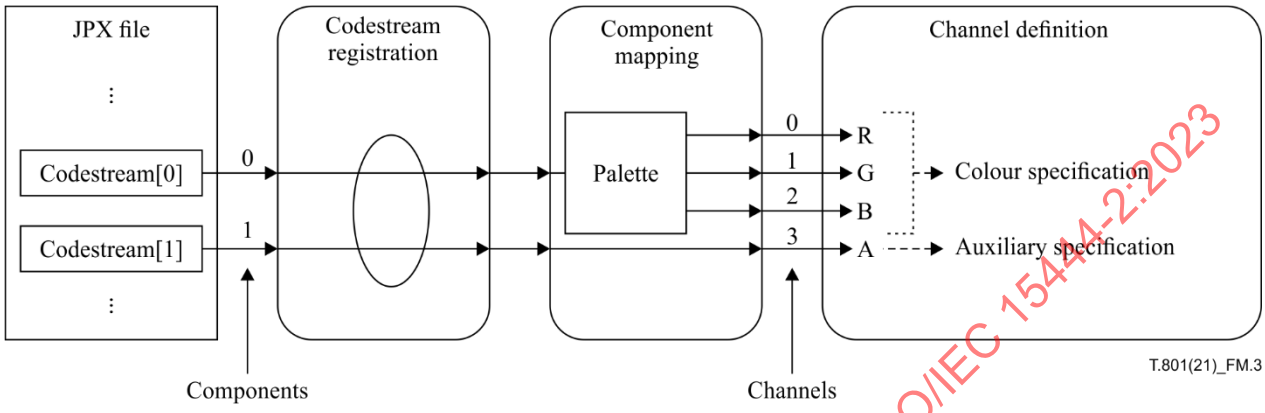

Figure M.3 – Example combination of two codestreams into a single compositing layer

M.5.1.1 Establishing a sequence order for compositing layers

A sequence order for compositing layers is required for any subsequent rendering or animation of the file. In the simplest case there are no Codestream Registration boxes in the file. In this case, which includes the case where all of the headers are present as global defaults, codestreams map directly to compositing layers and the compositing layer sequence order is given by the sequence order of the codestreams in the file.

If a Component Registration box is present in any Compositing Layer Header box, then there shall be one Compositing layer header box for every Compositing Layer in the file, each containing at least one Component Registration box. In this case, the order of compositing layers is given by the sequence order of compositing layer header boxes in the file.

M.5.1.2 Establishing an order for channels in a compositing layer

Where multiple codestreams are combined it is necessary to establish a sequence order over the combined set of channels generated from them. This sequence order is required so that specific channel numbers can be associated with channel definitions when using a Channel Definition box.

Channel ordering is zero based and performed independently for each compositing layer present in the file. The first n channels (numbered 0 through to $n - 1$) within the scope of a particular compositing layer are contributed by channels defined by the first Codestream Header box referenced in the layer's Component Registration box (where we assume this codestream generates n channels), the next m by the next codestream referenced in the layer's Component Registration box, and so on. Within each codestream, channel ordering is determined by the order of entries in the codestream's Component Mapping box, or by the order of components in the codestream if no Component Mapping box is present.

M.5.1.3 Establishing a sequence order for codestreams

A sequence order for codestreams is required for the subsequent ordering of compositing layers and in particular for the case where one or more compositing layers comprises more than one codestream.

Codestreams are assigned a sequence order equal to their position in the file, starting with zero. In the case where a Fragment Table box is used, the Fragment Table box is considered equivalent to a Contiguous Codestream box for the purpose of establishing a codestream sequence order for the referenced codestream.

M.5.2 Sharing header and metadata information between codestreams and compositing layers

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.

M.5.2.1 Default headers and metadata

When a JP2 Header box is found in a JPX file, the header information in that box shall be used as global default header information for all codestreams and compositing layers within the file. If a Codestream Header box includes boxes that also appear in the JP2 Header box, then these headers shall override the global headers for that specific codestream. If a Codestream Header box contains other boxes that do not appear in the global JP2 Header box, then these boxes shall augment the global header information for that specific codestream. Similarly, if a Compositing Layer Header box includes boxes that appear in the JP2 Header box, then these shall override the global headers for that specific compositing layer. If a Compositing Layer Header box contains other boxes that do not appear in the global JP2 Header box, then these boxes shall augment the global header information for that specific compositing layer.

Any metadata box, including IPR, XML and UUID boxes defined by the JP2 specification as well as additional metadata boxes defined by this specification, found at the file level (not contained within any other superbox) shall also be considered as containing global default information. As with header boxes, these global defaults may be overridden or augmented on a per codestream or per compositing layer basis by the inclusion of corresponding boxes in the codestream or layer header superboxes.

M.5.2.2 Cross-referencing headers and metadata

A Codestream Header box or Compositing Layer Header box may also contain a cross-reference to a box stored in another location. This cross-reference is very similar to the Fragment Table box used to specify the location of a fragmented codestream. In fact, a Cross-Reference box uses the same data structures as the Fragment Table box, with the addition of a field to specify the type of box being referenced. If a Codestream Header box or Compositing Layer Header box contains a Cross-Reference box, the reader shall consider the box pointed to by the reference as if it had been physically contained with the header. Cross-Reference boxes may be used equally for header and metadata.

M.5.2.3 Labelling and association

The Association box may be used to share a label (or other metadata) between codestreams and compositing layers by the inclusion of a Number List box within the Association box. A Number List box refers, by number, to a set of entities in the file. If the first box within an Association box is a Number List box, then any other boxes within the Association box will be associated with all of the entities referred to by the Number List box.

M.5.2.4 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 shall 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 principal 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 Rec. ITU-T T.808 | ISO/IEC 15444-9.

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

M.5.3 Composition

Composition data is divided into fixed options, contained in the Composition Options box (clause M.11.10.1), and a sequence of instructions contained in one or more Instruction Set boxes (clause 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 constitutes the primary presentation, while the latter constitutes 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.

M.5.3.1 Composition rendering

The composition data defines the width and height of a render area into which the compositing layers are to be rendered. The size of the render area is the size of the rendered result and can be thought of as the overall image size. Parameters in each render instruction may specify:

- a rectangular region to crop from the source compositing layer;
- the location to place the top left corner of the (possibly cropped) compositing layer with respect to the top left corner of the render area;
- the width and height of the region within the render area into which the (possibly cropped) compositing layer is to be rendered.

For example, in a composite image using an RGBA colour space for all compositing layers, the current compositing layer (R_t, G_t, B_t, A_t) is ideally rendered over the background (R_b, G_b, B_b, A_b) to form the composed image (R_c, G_c, B_c, A_c) according to the following equations:

$$\begin{aligned} A_c &= 1 - (1 - A_t) \cdot (1 - A_b) \\ s &= \frac{A_t}{A_c} \\ t &= \frac{(1 - A_t) \cdot A_b}{A_c} \\ R_c &= sR_t + tR_b \\ G_c &= sG_t + tG_b \\ B_c &= sB_t + tB_b \end{aligned} \tag{M-1}$$

In the case where the bottom sample is fully opaque, this simplifies to:

$$\begin{aligned} R_c &= A_t R_t + (1 - A_t) R_b \\ G_c &= A_t G_t + (1 - A_t) G_b \\ B_c &= A_t B_t + (1 - A_t) B_b \end{aligned} \tag{M-2}$$

However, the above equations require access to the background pixel, and for a variety of reasons, individual applications may not be willing or able to support such a rendering process. It is possible to emulate continuous alpha blending even in these cases by thresholding or dithering the provided alpha channel in order to generate a set of completely transparent or completely opaque pixels which can be rendered using simple pixel replacement over an unknown background. Specification of such methods is however outside the scope of this Recommendation | International Standard.

M.5.3.2 Animation model

In addition to the basic cropping and positioning parameters, each render instruction may include LIFE, PERSIST and NEXT-USE parameters. The LIFE parameter assigns a temporal duration to the instruction. This is the period of time that the reader should aim to place between the appearance of any screen update resulting from execution of the current instruction and any screen update resulting from the execution of the next instruction. PERSIST is a binary field indicating whether or not the compositing layer rendered by the current instruction should be treated as part of the background for the next instruction. If an instruction specifies false for PERSIST, then the reader shall save the background prior to execution and use this saved background when executing the next instruction.

M.5.3.2.1 Special cases of life and persistence

There are a number of special combinations of LIFE and PERSIST parameters that require specific treatment by the reader.

- When PERSIST is false and LIFE is zero, no action should be performed by the reader. This combination might be used for example to force a reader to step over a thumbnail or print frame that would be displayed by a reader not capable of displaying the file as an animation.
- When PERSIST is true and LIFE is zero then this instruction should be executed together with the next instruction. In practice this combination may occur for a sequence of more than two instructions and shall place the reader into a frame composition mode. This mode is exited when an instruction with non-zero LIFE is encountered or when the end of the animation is reached. The set of instructions executed whilst in frame composition mode is referred to as a frame composition sequence. In frame composition mode, a virtual compositing layer is created (off-screen) by executing the instructions in the frame definition sequence. The PERSIST and LIFE parameters for the closing instruction of a frame definition are applied to the virtual compositing layer as a whole. That is, upon close of frame composition mode, all of the compositing layers involved in the frame composition sequence acquire the PERSIST and LIFE values of the closing frame in the sequence.
- When LIFE is the maximum value that can be stored the reader shall interpret this as a request for indefinite life. If the driving application has the capacity, it shall progress to the next instruction upon completion of some predetermined user interaction such as a mouse click. In addition to its use in animations, this feature may be used in files that store multi-page documents in order to force the reader to pause after composition of each page.

In general, screen updates shall not be performed after an instruction which has zero LIFE unless it is the last instruction in a frame composition sequence.

M.5.3.2.2 Assigning compositing layers to instructions and layer reuse

Compression of animated sequences is considerably improved if compositing layers can be reused in multiple frames. At the same time it is desirable that instructions only reference compositing layers that have already been decoded or are sequentially next in the file. Further, decoders can better optimize their caching of compositing layers if they can tell which layers are to be reused ahead of time. These policies are enforced in JPX via the discipline used to associate compositing layers with instructions.

The first instruction is always associated with the first compositing layer in the file. This instruction may specify a value for NEXT-USE. This value is interpreted as the number of instructions, including the current instruction, until the current compositing layer is reused. A value of zero indicates to the reader that the current compositing layer shall not be used again and may be forgotten. A value of one implies that the current compositing layer is to be used with the next instruction and so on. The reader shall maintain a record of which instructions have been assigned compositing layers in this fashion. Whenever an instruction is encountered that does not have a compositing layer assigned to it, the next unused compositing layer defined in the file in sequence order shall be used; the use of NEXT-USE does not specify a loop. A single compositing layer may be reused any number of times in any given animation.

M.5.3.2.3 Looping animations

It is possible to specify that an animation should be looped. That is, when the animation has been fully rendered, the reader resets the display to its initial state and displays the animation over again. A loop count is optionally specified as part of the composition options. As with life, the maximum value for the loop parameter is used to indicate indefinite looping. Looping impacts the caching strategy used by the reader as many readers will not wish to free any compositing layer once decoded.

M.6 Using reader requirements masks to determine how a file can be used

The JPX format defines a file architecture rather than a specific, fixed set of data structures that will be found in a file. This architecture is complex enough to permit quite distinct file structures. For example, a JPX file may include:

- animation;
- image collections;
- redundant image sets (e.g., print and display versions of the same scene); or
- single images in special colour spaces (e.g., Parameterized CIELab).

As a result, the JPX brand tells a reader little about what capabilities it will require in order to correctly read an arbitrary JPX file. Instead, JPX conveys this information using three expressions contained within the header of the file. These expressions describe:

1) the full set of technologies/features present in the file;
2) the set of technologies/features required by a decoder in order to read the file in a form consistent with the intent of the files creator;
3) a fallback mode which can be used to display a minimally acceptable result (usually a thumbnail or preview).

The fallback mode is communicated using the File Type box and is primarily intended to indicate whether or not the file can be read by a reader with specifically standardized capabilities (such as a conforming JP2 or Baseline JPX reader). However, the combination of technologies required may be too complex for a simple listing of the functionalities in the file. For this reason, the technology/feature set information takes the form of encoded logic expressions and are contained in a Reader Requirements box.

In general, a reader need only identify that it satisfies the requirements outlined in point 2 in order to be assured of being able to read enough of the file to fulfil the creator's intent. On the other hand, an editor may want to inform a user if there is any aspect of the file that it does not know how to support. Because all JPEG 2000 family files are not necessarily interoperable with all JPX readers, these expressions describe each aspect of the file, and the combination of features that shall be supported to interpret the file correctly.

M.6.1 Types of expressions

M.6.1.1 Fully understand aspects

An encoded expression is used to describe all of the items contained within the file, and the combinations of functionality required to read these items. This expression describes each major option a reader has for processing the features of the file, regardless of whether support for a particular feature is required to make use of that aspect of the file. For example, a file may contain metadata describing an original file from which it was created; however, a reader is not required to understand this metadata in order to correctly use the file.

M.6.1.2 Display contents

A second expression is used to describe the functionality required to display the contents of the file as desired. Files may contain several representations of a single image, so that the expression to display the contents correctly may include several options.

M.6.1.3 Fallback

In the event that a file cannot be displayed as desired by the writer, a fallback method for displaying the file is defined. The fallback methods are intended to be ultimately interoperable, as such, they may not generate the exact desired output.

A list of fallback methods is stored in the File Type box at the beginning of the JPEG 2000 family file: all files which start with a JPEG 2000 Signature box shall contain a File Type box. The File Type box contains a list of known methods of reading the file. For example, the JP2 file format defines the JP2 fallback position. This specifies that a reader conforming to the JP2 file format specification, as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause I.2.6, can read the file.

This Recommendation | International Standard defines one additional fallback position, JPX baseline, as defined in clause M.9. These define other minimal readers, and a set of file formats that are guaranteed to be readable by those readers.

Fallback methods are stored in the File Type box. The order within the box is of no significance. When a reader supports more than one of the fallback methods described in the file, it is up to the reader to determine which fallback methods to use.

M.6.2 Expression representation

The expressions of the requirements to fully understand all aspects, and to display the file as desired are stored in the Reader Requirements box, which is a mandatory feature of a JPX file format. If the Reader Requirements box is not present, the File Type box describes the full functionality of the file.

The Reader Requirements expressions are logical expressions involving functions which can be provided by the writer. These expressions may include vendor-specific options. The expression is factored into `AND`-separated sub-expressions, each containing only `OR` operations. These are then encoded into bitmasks which the reader can use to determine how to handle the file.

The Reader Requirements box includes two expressions, the Fully Understand Aspects expression, and the Display Contents expression. In general, these expressions will share several sub-expressions; shared sub-expressions are only stored once, and bitmasks are used to determine which sub-expressions belong to each expression.

M.6.2.1 Formulating requirements expressions

When formulating the requirements expressions describing a file, each expression is firstly factored into AND-separated sub-expressions, each sub-expression containing OR-separated options. Thus, an expression in the form:

$$(A \,\&\, B \,\&\, C \,\&\, E) \mid (D \,\&\, E) \tag{M-3}$$

is factored into the expression:

$$(A \mid D) \,\&\, (B \mid D) \,\&\, (C \mid D) \,\&\, E \tag{M-4}$$

Each sub-expression is expressed as a bit-array, with a flag set for each option appearing in that sub-expression. So, for example, the expression (A | D) becomes:

Table M.1 – Example expression

A	B	C	D	E
1	0	0	1	0

The full expression is written in table form (see Table M.2):

Table M.2 – Expanded expression

A | D	B | D	C | D	E	
1	0	0	0	A
0	1	0	0	B
0	0	1	0	C
1	1	1	0	D
0	0	0	1	E

with each sub-expression in a column of the table. Thus, to satisfy the requirements of this expression, a reader has to support one of the functionalities in each column of the table.

However, there are two expressions to be encoded, and they will, in general, share common factors (because the functionality required to display a file is part of the functionality required to fully understand its contents). Thus, the two expressions are combined into one table, and a bitmask is provided to determine which columns in the table belong to each expression. So, if the expression in Equation M-3 is Display Contents and if the expression for Fully Understand Aspects is:

$$((A \mid D) \,\&\, (B \mid D) \,\&\, (C \mid D) \,\&\, E) \,\&\, (F \mid G) \tag{M-5}$$

then, noting that the expression in Equation M-4 is a common factor, here, the resulting table is (see Table M.3):

Table M.3 – Example factored expression

1	0	0	0	0	A
0	1	0	0	0	B
0	0	1	0	0	C
1	1	1	0	0	D
0	0	0	1	0	E
0	0	0	0	1	F
0	0	0	0	1	G
1	1	1	1	0	Display contents
1	1	1	1	1	Fully understand

where the first four columns are the Display Contents requirement, and all five sub-expressions are required to Fully Understand Aspects. Thus the bitmask for Display Contents is 11110, and the bitmask for Fully Understand Aspects is 11111.

This table can be read in columns, as a set of sub-expressions defining the functionality required of a reader, or in rows, as a set of compatibility bitmasks which a reader can use to determine whether it can read the file. By obtaining the bitwise OR of the rows which correspond to the functionalities present, and comparing the result with the bitmasks for the two expressions, a reader can determine whether it can satisfy the requirements of each.

Thus, a writer can construct the table in columns, setting flags corresponding to the options in each sub-expression, and generating the bitmasks describing which sub-expressions are ANDed together to form the full expressions for Fully Understand Aspects, and Display Contents. It can then obtain the compatibility bitmasks for each function which a reader may use in reading the file, by extracting the row corresponding to each functionality present.

M.6.2.2 Encoding requirements expressions

The requirements expressions are encoded in the Reader Requirements box, starting with a mask length, indicating the width of the compatibility bitmasks, to byte precision. This is followed by the bitmasks for the Fully Understand Aspects and Display Contents expressions, and in turn by a list of the features used and their compatibility masks, obtained from the rows of the expression table.

The list includes a set of standard features used (as specified in Table M.14) and their compatibility bitmasks, followed by a list of vendor-specific features (represented as UUIDs), together with the compatibility bitmasks associated with these. Apart from the separation into standard and vendor-specific features, the order of presentation is unimportant. This structure is fully specified in clause M.11.1.

M.6.2.3 Examples

For example, consider an image processing program that produces a JPX file containing a single image in the sRGB colour space, and a multiple codestream version containing the compositing layers, to allow an editor to work with the image, and includes metadata containing the history of the file, then the requirement to display the file is:

sRGB & (single codestream | (multiple codestream & compositing)) (M-6)

and to fully understand the file requires:

sRGB & (single codestream | (multiple codestream & compositing)) & metadata (M-7)

Equation M-6 factors as:

sRGB & (single codestream | multiple codestream) & (single codestream | compositing) (M-8)

Equation M-7 factors similarly, so the sub-expressions are:

a) sRGB;

b) single codestream | multiple codestream;

c) single codestream | compositing;

d) metadata.

The resulting expression table and bitmasks is shown in Table M.4.

The bitmasks indicate which sub-expressions are required for each degree of functionality. Thus the expression for Display Contents is:

(Sub-expr a) & (Sub-expr b) & (Sub-expr d) (M-9)

Table M.4 – Example of a Reader Requirements expressions for Equations M-6 and M-7

	Sub-Exp a	Sub-Exp b	Sub-Exp c	Sub-Exp d
sRGB	1	0	0	0
single codestream	0	1	1	0
multiple codestream	0	1	0	0
compositing	0	0	1	0
metadata	0	0	0	1
Fully Understand Aspects bitmask	1	1	1	1
Display Contents bitmask	1	1	1	0

Thus, the above table is stored in the file as shown in Table M.5:

Table M.5 – Example of a Reader Requirements box for Equations M-6 and M-7

Mask Length (in bytes)	1[a]				
Fully Understand Aspects bitmask	1	1	1	1	
Display Contents bitmask	1	1	1	0	
Number of Standard Features	5				
Standard Feature Compatibility list	sRGB				
	1	0	0	0	
	single codestream				
	0	1	1	0	
	multiple codestream				
	0	1	0	0	
	compositing				
	0	0	1	0	
	metadata				
	0	0	0	1	
Number of Vendor-Specific features	0				

[a] 1 byte, because masks are 4 bits wide, which fits into 1 byte.

As a second example, suppose the ACME printer driver produces a JPX file which contains a single codestream sRGB image, for display, and a CMYK image which can be read by a printer driver using ACME's vendor-specific functions. For this file, the expression to Fully Display Contents is:

(sRGB & single codestream) | (CMYK & single codestream & ACME extensions) (M-10)

while the expression to Understand All Aspects is:

((sRGB & single codestream) | (CMYK & single codestream & ACME extensions))
& metadata & ACME print metadata (M-11)

Factorizing these into sub-expressions gives:

single codestream & (sRGB | CMYK) & (sRGB | ACME extensions) (M-12)

and:

single codestream & (sRGB | CMYK) & (sRGB | ACME extensions)
& metadata & ACME print metadata (M-13)

respectively.

The resulting file Reader Requirements table is shown in Table M.6:

Table M.6 – Reader Requirements table for Equations M-10 and M-11

sRGB	0	1	0	1	0
CMYK	0	1	0	0	0
single codestream	1	0	0	0	0
metadata	0	0	1	0	0
ACME extensions	0	0	0	1	0
ACME print metadata	0	0	0	0	1
Fully Understand Aspects bitmask	1	1	1	1	1
Display Contents bitmask	1	1	0	1	0

As always, each column represents a factor sub-expression, and each row provides a compatibility bitmask which a reader can use to determine whether it can read the file. This example includes vendor-specific features, and that sub-expressions can involve both standard and vendor-specific functionality.

These are stored in the file as shown in Table M.7:

Table M.7 – Reader Requirements box data for Equations M-10 and M-11

Mask Length	1					
Fully Understand Aspects bitmask	1	1	1	1	1	
Display Contents bitmask	1	1	0	1	0	
Number of Standard Features	4					
Standard Feature Compatibility list	sRGB					
	0	1	0	1	0	
	CMYK					
	0	1	0	0	0	
	single codestream					
	1	0	0	0	0	
	metadata					
	0	0	1	0	0	
Number of Vendor Features	2					
Vendor Feature Compatibility list	ACME extensions UUID					
	0	0	0	1	0	
	ACME print metadata UUID					
	0	0	0	0	1	

Also consider a JPX file that contains two compositing layers that are not combined by either animation or compositing; they are conceptually two separate rendered results. The first compositing layer contains a single codestream in the sRGB colourspace (specified using the Enumerated method). The second compositing layer contains a single codestream, for which the colourspace is specified using the Any ICC method. In addition, the second compositing layer contains vendor-specific metadata.

For this file, the expression to Fully Display Contents is:

(sRGB & single codestream) | (full ICC & single codestream & ACME extensions) (M-14)

while the expression to Understand All Aspects is:

((sRGB & single codestream) | (full ICC & sRGB & single codestream & ACME extensions)) (M-15)

Factorizing these into sub-expressions gives:

single codestream AND (sRGB | full ICC) AND (sRGB | ACME extensions) (M-16)

and:

single codestream AND sRGB AND full ICC AND ACME extensions (M-17)

respectively. The resulting file Reader Requirements table is shown in Table M.8:

Table M.8 – Reader Requirements box data for Equations M-16 and M-17

Mask Length	1						
Fully Understand Aspects bitmask	1	0	0	1	1	1	
Display Contents bitmask	1	1	1	0	0	0	
Number of Standard Features	3						
Standard Feature Compatibility list	sRGB						
	0	1	1	1	0	0	
	Any ICC						
	0	1	0	0	1	0	
	single codestream						
	1	0	0	0	0	0	
Number of Vendor Features	1						
Vendor Feature Compatibility list	ACME extensions UUID						
	0	0	0	1	0		

M.6.3 Testing an Implementation against requirements expressions

In order to determine whether it can read the file, the reader extracts the compatibility bitmask from the feature list entry corresponding to each functionality which it provides. If a flag is set in the bitmask, then this function is an option in the sub-expression corresponding to the flag.

Thus, if the reader performs a bitwise OR of the bitmasks for all of the functions which it provides, it can determine whether it can read the file by comparing the result with the Fully Understand Aspects and Display Contents bitmasks from the file. Also, by reconstructing the expression table and looking up the column (or columns) of the table where the file bitmask flag is set, and the reader's compatibility bitmask flag is not, the reader can determine which extra functionality is required to read the file.

If there is functionality provided by the reader, which is not in the feature list for the file, then the feature is not required to read the file (and the bitmask may be assumed to be all zeroes).

Consider the first example Reader Requirements box (Table M.9):

Table M.9 – Example Reader Requirements box to test

Mask Length (in bytes)	1				
Fully Understand Aspects bitmask	1	1	1	1	
Display Contents bitmask	1	1	1	0	
Number of Standard Features	5				
Standard Feature Compatibility list	sRGB				
	1	0	0	0	
	single codestream				
	0	1	1	0	
	multiple codestream				
	0	1	0	0	
	compositing				
	0	0	1	0	
	metadata				
	0	0	0	1	
Number of Vendor-Specific features	0				

In this example, if the reader supports the sRGB and single codestream functions, it looks up the bitmasks for these features (`1000` and `0110`, respectively). The bitwise OR gives the compatibility mask of this file for the reader, `1110`. Thus, this reader can fully display the contents of the file; however, it will not understand all aspects of the file.

Noting that the compatibility mask for the reader (DCM) is 1110, and the Fully Understand Aspects Mask (FUAM) mask is 1111, the reader can perform (FUAM & !DCM) bitwise, to get 0001. This tells it that the missing functionality's bitmask has bit 4 set, so it can search the list for this, and determine that the missing functionality is metadata support.

Table M.10 – *Table intentionally left blank*

M.7 Extensions to the JPX file format

M.7.1 General

In the JPX file format, several features of the file format (denoted user items) can be extended unilaterally, without consultation with ITU-T | ISO/IEC. For example, the format provides the Vendor Colour method to allow individual vendors to indicate custom colourspaces through a form of enumeration (using UUID's) without involving a third-party. Other features (denoted reserved items) require specification in a Recommendation | International Standard.

M.7.2 Reserved items

M.7.2.1 Definition

Reserved items are those elements of a JPX file that are restricted to a limited (albeit large in some cases) number of values. For these items, there exists the possibility that two vendors would use the same value for different meanings if there is no third-party mediating the use of the element values. Also, in most cases, there are additional criteria for the allocation of values to registered items. Because the number of available values for most registered items is limited, and given that most problems can be solved using publishable items rather than registered items, the allocation of a registered value is made through Recommendations | International Standards. For example, an Enumerated Colourspace is a reserved item; the value is indicated through the use of a 4-byte integer that is specified in Recommendations | International Standards.

Table M.11 lists reserved items.

Table M.11 – Items which can be extended through Recommendations | International Standards

Item	Purpose
Enumerated colourspaces	Define additional standard colourspaces
Desired reproduction boxes	Define additional reproduction scenarios and the data required to transform images for output in those scenarios
Compatibility modes	Define additional compatibility modes to promote interoperability in markets not explicitly addressed by the JPX baseline feature set
Standard feature list	Define additional standard feature codes for the Reader Requirements box

M.7.2.2 Enumerated colourspace

New values of the EnumCS field in the Colour Specification Box are specified in Recommendations | International Standards. Each enumerated colourspace contains a complete colourimetric definition of that colourspace, instructions on how to use images in that colourspace, any required enumerated parameters (for the EP field in the Colour Specification box) and any default values of those parameters.

NOTE – Non-standard colourspaces shall be specified using the Vendor Colourspace method.

M.7.2.3 Desired reproduction boxes

New box types for Desired Reproduction information (like the Graphics Technology Standard Output box in the JPX format) are specified in Recommendations | International Standards. Each desired reproduction contains a complete definition of the reproduction scenario, including the binary structure of the reproduction data as well as when an application uses the reproduction data.

NOTE – Reproductions can also be specified by embedding the data in a UUID box and placing that UUID box within the Desired Reproduction superbox.

M.7.2.4 Compatibility modes

New compatibility modes for the File Type box (values for the CLi fields) are specified in Recommendations | International Standards. Each compatibility mode defines a complete definition of the JPX reader requirements for that compatibility mode, as well as the definition of the 4-byte CLi field for this mode.

M.7.2.5 Standard feature codes

New values of the SFi field in the Reader Requirements box are specified in Recommendations | International Standards.

M.7.3 User items

User items are those elements of a JPX file that can be safely extended, generally through the use of URL's or UUID's, without risk of conflict with other users. Values can be assigned for user items without consultation with a third-party.

Table M.12. list user items.

Table M.12 – Items which can be extended by registration

Item	Purpose	Values
Vendor feature codes	Define additional vendor-specific features	VF^i field in the Application Profile box
Vendor colourspaces	Define additional vendor-specific colourspaces	VCLR field in the METHDAT field for Colour Specification boxes that use the Vendor Colour method
Binary filter algorithms	Define additional algorithms for use in the Binary Filter box	F field in the Binary Filter box
UUID metadata	Define additional metadata for use within UUID boxes	UUID's used in UUID boxes
XML metadata	Define additional metadata for use within XML boxes	Document Type Definitions (DTD's) and XML Schema's used in XML boxes

M.8 Differences from the JP2 binary definition

The box structure of a JPX file is identical to that of a JP2 file. A JPX file is a sequence of boxes, defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause I.6. However, many new boxes are defined, and the structures of several boxes are extended as follows:

- The BR field in the File Type box shall be "jpx\040" for files that are completely defined by this Recommendation | International Standard. In addition, a file that conforms to this Recommendation | International Standard shall have at least one CLi field in the File Type box, and shall contain the value 'jpx\040' in one of the CLi fields in the File Type box.
- The MinV field in the File Type box shall bet set to 1, indicating a minor revision to the standard.
- Additional forms of the Colour Specification box are defined (M.11.7.2).
- The JPEG 2000 compressed codestream may contain extensions as defined in Annex A.
- Under some circumstances, the JP2 header box may be found in anywhere in the file, provided that it is not encapsulated within another box (it shall always be at the top level of the file). See M.11.5 for a description of the storage of the JP2 header box within a JPX file.
- Additional box types are defined within the scope of this Recommendation | International Standard.

M.9 Conformance

M.9.1 Interpretation of JPX data structures

A conforming file shall contain the boxes that are labelled as required in Table M.13, and those boxes shall be as defined in this Recommendation | International Standard.

A JPX reader that supports a particular subset of JPX features is a conforming JPX reader if that reader properly supports all files that contain a Display Contents mask (in the Reader Requirements box) or a Fallback position (in the File Type box) indicating that the file can be read using only that particular subset of features; a conforming reader may fall back from any extended feature, as allowed by the Reader Requirements or File Type box, provided that the reader does not claim a higher level of conformance than it actually supports.

M.9.2 Support for JPX feature set

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 Rec. ITU-T 800 |

ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2 only, and four additional profiles describing images containing only codestreams conforming to Rec. 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 CL_i 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 clauses M.9.2.1 through M.9.2.9, the JPEG XR profiles based on 29199-2 codestreams are defined in clause M.9.2.10 and following.

The ITU-T 80x | ISO/IEC 15444-x based profile denoted JHX Baseline is identical to the JPX Baseline profile with the exception that the codestream in the first compositing layer shall belong to the HTONLY set specified in Rec. ITU-T T.814 | ISO/IEC 15444-15 and shall not require support for extensions other than the irreversible decorrelation transformation (specified in clause J.3.1.1.1) and non-linearity transformation (specified in Annex K) extensions. Files that are written in such a way as to allow a reader that supports only this JHX Baseline set of features to properly open the file shall contain a CL_i field in the File Type box with the value 'jhxb' (0x6a68 7862). All JHX baseline readers are required to properly support all files with this code in the compatibility list in the File Type box.

M.9.2.1 Compression types

Support for compression types other than JPEG 2000 (the C field in the Image Header box = 7) shall not be required to properly display the file.

M.9.2.2 Compositing layers

Support for multiple compositing layers is not required to properly display the file. However, the file may contain multiple compositing layers. If the file does contain compositing layers, the first compositing layer in the file (signalled by the first Compositing Layer Header box) shall be rendered. That compositing layer shall consist of one and only one codestream, which shall represent the rendered result as rendered into a single codestream. In addition, the codestream that shall be processed by a reader that only supports the JPX feature set shall be the first codestream in the file.

M.9.2.3 Codestreams

The codestream specified by the first compositing layer shall be compressed using the JPEG 2000 compression algorithm, as defined by Rec. ITU-T T.800 | ISO/IEC 15444-1 and shall not require support for extensions other than the irreversible decorrelation transformation (specified in clause J.3.1.1.1) and non-linearity transformation (specified in Annex K) extensions.

A conforming JPX baseline reader is not required to support other portions of the multiple component transformation extension. If support for the irreversible decorrelation transformation is required, then the first codestream shall be restricted as follows:

- The value of the Qmcc field in any MCC marker segment shall be 1.
- The $Xmcc^i$ field in any MCC marker segment shall indicate an array based decorrelation transformation.
- The $Tmcc^i$ field in any MCC marker segment shall indicate an irreversible transformation
- The Nmco field in any MCO marker segment shall be 1.

That codestream may contain other extensions provided that support for those extensions is not required to decode the codestream.

Other codestreams in the file may require support for other extensions in order to be decoded.

M.9.2.4 Colour specification

The first compositing layer shall contain at least one Colour Specification box from the following list:

- Enumerated method EnumCS values indicating either sRGB, sRGB-grey, ROMM-RGB, sYCC, e-sRGB, or e-sYCC.
- Enumerated method EnumCS value of CIELab using default values (EP fields are not specified).
- Enumerated method EnumCS value of CIELab using enumerated parameters (as specified in the EP fields in the Colour Specification box).
- Enumerated method EnumCS value of CIEJab using default values (EP fields are not specified).
- Enumerated method EnumCS value of CIEJab using enumerated parameters (as specified in the EP fields in the Colour Specification box).

– Restricted ICC method.
– Any ICC method.

A baseline JPX file may contain additional colourspace specifications, such as other enumerated values or vendor defined colourspace specifications. However, the file shall contain at least one colour specification method from the list above.

In addition, at least one Colour Specification box specified for the first compositing layer shall have an APPROX value of 3 or less (indicating a "reasonable" or better approximation of the true colourspace of the image).

M.9.2.5 Codestream fragmentation

The codestream used by the first compositing layer in a baseline JPX file may be fragmented. However, all fragments shall be in the JPX file itself and shall be found in the file in the order they are listed in the Fragment Table box, starting the search at byte 0 of the file and proceeding sequentially to the end of the file.

M.9.2.6 Cross-reference boxes

All Cross-Reference boxes that shall be parsed in order to properly interpret or decode the first compositing layer in the file shall only point to fragments that are contained within the JPX file itself. Those fragments shall be in the same order in the file as they are listed in the Fragment List box, starting the search at byte 0 of the file and proceeding sequentially to the end of the file. In addition, all fragments shall be found in the file before the data representing the codestream used by the compositing layer. If that codestream is specified by a Contiguous Codestream box, then all fragments for the cross-reference shall be found before that Contiguous Codestream box. If the codestream is specified by a Fragment Table box, then all fragments for the cross-reference shall be found before the Media Data box containing the first fragment from the codestream.

M.9.2.7 JP2 Header box location

The JP2 Header box shall be found in the file before the first Contiguous Codestream box, Fragment Table box, Media Data box, Codestream Header box, and Compositing Layer Header box. Any information contained within the JP2 Header box shall be applied to the first codestream, as well as being used as default information for all other codestreams and 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.

M.9.2.8 Opacity

A baseline JPX reader shall properly interpret opacity channels, through either direct mapping to a codestream component using either the Channel Definition box or the Opacity box, or by expansion from a palette. The use of opacity outside of the use of compositing layers within the JPX file indicates that the decoded image data shall be composited onto an application defined background.

M.9.2.9 Other data in the file

A baseline JPX 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 baseline JPX feature set. All baseline 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 behaviours and functions associated with that extended data.

M.9.2.10 JPEG XR profiles

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.

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.

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.

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.

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.

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.

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.

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.

M.9.2.18 Rotation

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

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 behaviours and functions associated with that extended data.

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 Rec. ITU-T 834 | ISO/IEC 29199-4.

M.10 Key to graphical descriptions (informative)

Each box is described in terms of its function usage and length. The function describes the information contained in the box. The usage describes the logical location and frequency of this box in the file. The length describes which parameters determine the length of the box.

These descriptions are followed by a figure that shows the order and relationship of the parameters in the box. Figure M.4 shows an example of this type of figure. A rectangle is used to indicate the parameters in the box. The width of the rectangle is proportional to the number of bytes in the parameter. A shaded rectangle (diagonal stripes) indicates that the parameter is of varying size. Two parameters with superscripts and a gray area between indicate a run of several of these parameters. A sequence of two groups of multiple parameters with superscripts separated by a gray area indicates a run of that group of parameters (one set of each parameter in the group, followed by the next set of each parameter in the group). Optional parameters or boxes will be shown with a dashed rectangle.

The figure is followed by a list that describes the meaning of each parameter in the box. If parameters are repeated, the length and nature of the run of parameters is defined. As an example, in Figure M.4, parameters C, D, E and F are 8, 16, 32 bit and variable length respectively. The notation G^0 and G^{N-1} implies that there are *n* different parameters, G^i, in a row. The group of parameters H^0 and H^{M-1}, and J^0 and J^{M-1} specify that the box will contain H^0, followed by J^0, followed by H^1 and J^1, continuing to H^{M-1} and J^{M-1} (M instances of each parameter in total). Also, the field E is optional and may not be found in this box.

After the list is a table that either describes the allowed parameter values or provides references to other tables that describe these values.

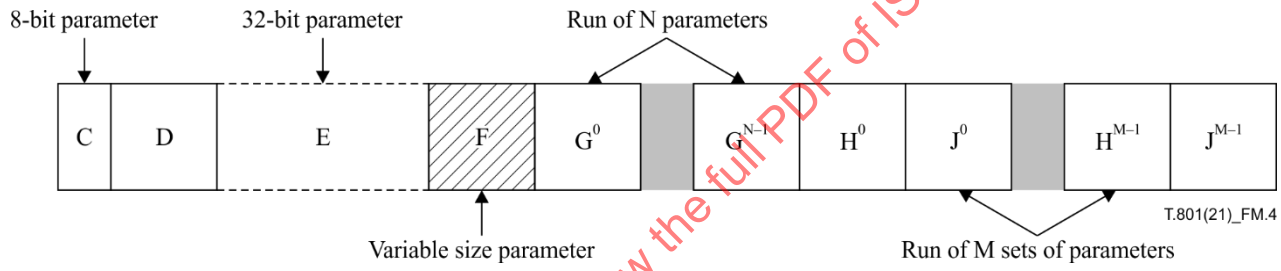

Figure M.4 – Example of the box description figures

In addition, in a figure describing the contents of a superbox, an ellipsis (…) will be used to indicate that contents of the file between two boxes is not specifically defined. Any box (or sequence of boxes), unless otherwise specified by the definition of that box, may be found in place of the ellipsis.

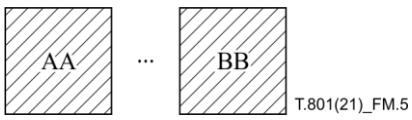

Figure M.5 – Example of the superbox description figures

For example, the superbox shown in Figure M.5 shall contain an AA box and a BB box, and the BB box shall follow the AA box. However, there may be other boxes found between boxes AA and BB. Dealing with unknown boxes is discussed in clause M.12.

M.11 Defined boxes

The following boxes are defined as part of JPX file format. In addition, any box defined as part of the JP2 file format that is not also listed here is also defined for use in a JPX file. However, this Recommendation | International Standard may redefine the binary structure of some boxes defined as part of the JP2 file format. For those boxes, the definition found in this Recommendation | International Standard shall be used for all JPX files.

Figure M.6 shows the hierarchical organization of the boxes in a JPX file. Several of these boxes are defined within the JP2 file format specification. This illustration does not specify nor imply a specific order to these boxes. In many cases, the file will contain several boxes of a particular box type. The meaning of each of those boxes is dependent on the placement and order of that particular box within the file.

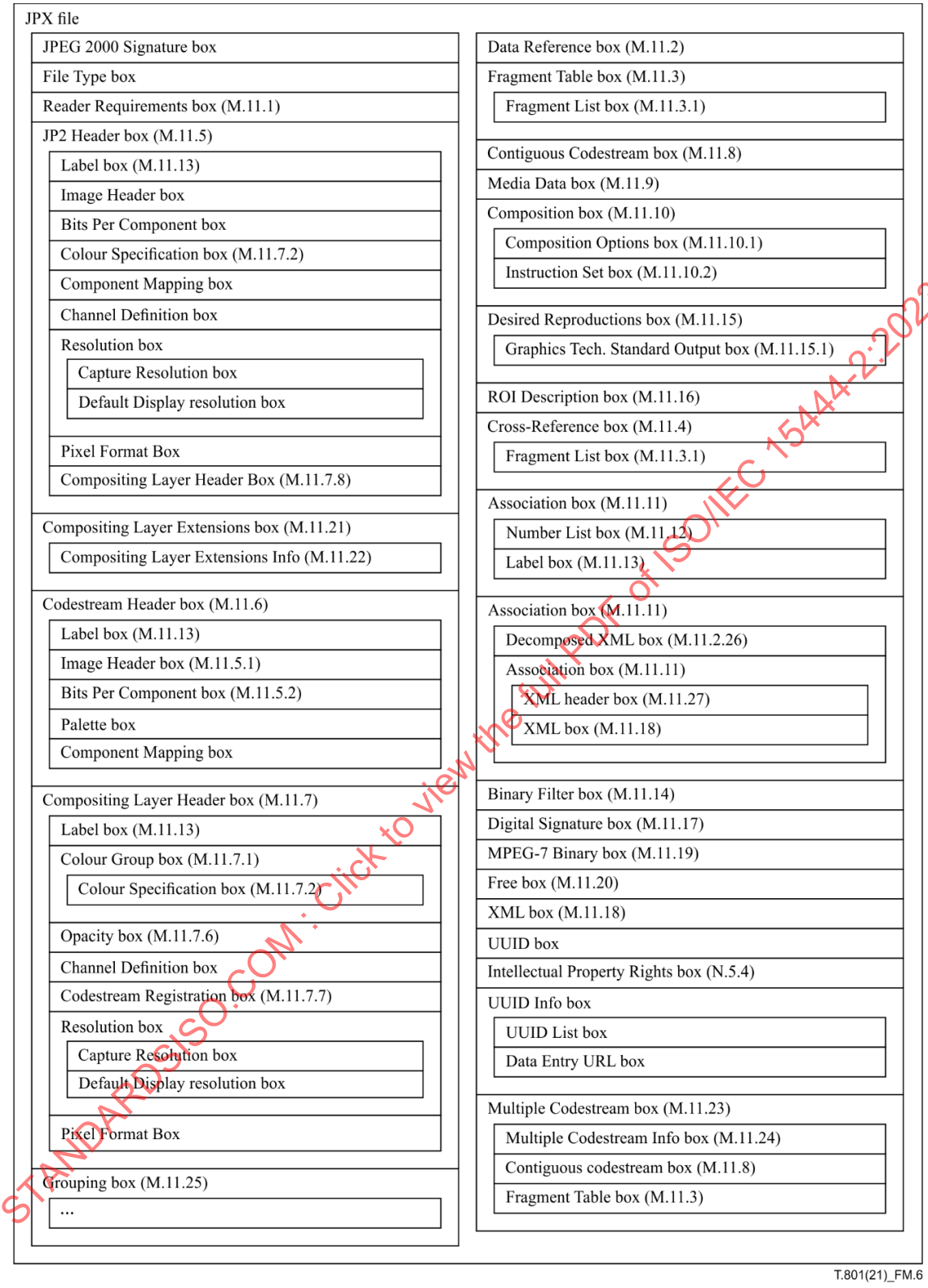

Figure M.6 – Boxes defined within a JPX file

Table M.13 lists all boxes defined as part of this Recommendation | International Standard. Boxes defined as part of the JP2 file format are not listed. A box that is listed in Table M.13 as "Required" shall exist within all conforming JPX files. For the placement of and restrictions on each box, see the relevant clause defining that box.

Table M.13 – Boxes defined within this Recommendation | International Standard

Box name	Type	Required ?	Comments
Reader Requirements box (M.11.1)	'rreq' (0x7272 6571)	Yes	This box specifies the different modes in which this file may be processed.
JP2 Header box (superbox) (M.11.5)	'jp2h' (0x6A70 3268)	No	This box specifies JP2 compatibility and default header information for the codestreams and compositing layers.
Image Header box (M.11.5.1)	'ihdr' (0x6968 6472)	Yes	This box specifies the size of the image and other related fields.
Bits Per Component box (M.11.5.2)	'bpcc' (0x6270 6363)	No	This box specifies the bit depth of the components in the file in cases where the bit depth is not constant across all components.
Codestream Header box (superbox) (M.11.6)	'jpch' (0x6A70 6368)	No	This box specifies general information, such as bit depth, height and width about one specific codestream in the file.
Compositing Layer Header box (superbox) (M.11.7)	'jplh' (0x6A70 6C68)	No	This box specifies general information, such as colourspace and resolution, about one specific compositing layer in the file.
Colour Group box (superbox) (M.11.7.1)	'cgrp' (0x6367 7270)	No	This box groups a sequence of Colour Specification boxes that specify the different ways that the colourspace of a layer can be processed.
Colour Specification box (M.11.7.2)	'colr' (0x636F 6C72)	Yes	This box specifies one way in which the colourspace of an image can be processed. The definition of this box is extended from the definition in the JP2 file format.
Opacity box (M.11.7.6)	'opct' (0x6F70 6374)	No	This box specifies how opacity information is contained within a set of channels.
Codestream Registration box (M.11.7.7)	'creg' (0x6372 6567)	No	This box specifies the alignment between the set of codestreams that make up one compositing layer.
Data Reference box (M.11.2)	'dtbl' (0x6474 626C)	No	This box contains a set of pointers to other files or data streams not contained within the JPX file itself.
Fragment Table box (superbox) (M.11.3)	'ftbl' (0x6674 626C)	No	This box specifies how one particular codestream has been fragmented and stored within this JPX file or in other streams.
Fragment List box (M.11.3.1)	'flst' (0x666C 7374)	No	This box specifies a list of fragments that make up one particular codestream within this JPX file.
Cross-Reference box (M.11.4)	'cref' (0x6372 6566)	No	This box specifies that a box found in another location (either within the JPX file or within another file) should be considered as if it was directly contained at this location in the JPX file.
Contiguous Codestream box (M.11.8)	'jp2c' (0x6A70 3263)	No	This box contains one codestream from the JPX file, stored contiguously in a single box.
Media Data box (M.11.9)	'mdat' (0x6D64 6174)	No	This box contains generic media data, which is referenced through the Fragment Table box.
Composition box (superbox) (M.11.10)	'comp' (0x636F 6D70)	No	This box specifies how a set of compositing layers shall be combined to create the rendered result.
Composition Options box (M.11.10.1)	'copt' (0x636F 7074)	No	This box specifies generic options for the composition of multiple compositing layers.
Instruction Set box (M.11.10.2)	'inst' (0x696E 7374)	No	This box specifies the specific instructions for combining multiple compositing layers to create the rendered result.
Association box (superbox) (M.11.11)	'asoc' (0x6173 6F63)	No	This box allows several other boxes (i.e., boxes containing metadata) to be grouped together and referenced as a single entity.
Number List box (M.11.12)	'nlst' (0x6E6C 7374)	No	This box specifies what entities are associated with the data contained within an Association box.
Label box (M.11.13)	'lbl\040' (0x6C62 6C20)	No	This box specifies a textual label for either a Codestream Header, Compositing Layer Header, or Association box.
Binary Filter box (M.11.14)	'bfil' (0x6266 696C)	No	This box contains data that has been transformed as part of the storage process (such as compressed or encrypted).

Table M.13 – Boxes defined within this Recommendation | International Standard

Box name	Type	Required ?	Comments
Desired Reproductions box (superbox) (M.11.15)	'drep' (0x6472 6570)	No	This box specifies a set of transformations that shall be applied to the image to guarantee a specific desired reproduction on a set of specific output devices.
Graphics Technology Standard Output box (M.11.15.1)	'gtso' (0x6774 736F)	No	This box specifies the desired reproduction of the rendered result for commercial printing and proofing systems.
ROI Description box (M.11.16)	'roid' (0x726F 6964)	No	This box specifies information about specific regions of interest in the image.
Digital Signature box (M.11.17)	'chck' (0x6368 636B)	No	This box contains a checksum or digital signature for a portion of the JPX file.
MPEG-7 Binary box (M.11.19)	'mp7b' (0x6D70 3762)	No	This box contains metadata in MPEG-7 binary format (BiM) as specified in ISO/IEC 23001-1.
Free box (M.11.20)	'free' (0x6672 6565)	No	This box contains data that is no longer used and may be overwritten when the file is updated.
Intellectual Property Rights (clause N.5.4)	'ipr' (0x6A70 3269)	No	This box contains intellectual rights information.
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 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.

M.11.1 Reader Requirements box

The Reader Requirements box specifies what features or feature groups have been used in this JPX file, as well as what combination of features shall be supported by a reader in order to fully use the file. The Reader Requirements box shall immediately follow the File Type box, and there shall be one and only one Reader Requirements box in the file.

All features specified are in addition to the features defined by the JP2 file format and JPEG 2000 codestream profile 0; it is assumed that any reader capable of reading a JPX file is also capable of understanding every feature defined in the JP2 file format and decoding a JPEG 2000 profile 0 codestream.

This box shall contain an accurate specification, to the extent as known by the writer, of all features in the file and an accurate specification of the set or sets of features required to display the image as intended by the writer.

NOTE – If a JPX file contains no features other than those defined by the JP2 file format and JPEG 2000 codestream profile 0, or if the reader does not know of any features contained in the file beyond those base features, the Reader Requirements box will list zero standard features and zero vendor features.

Many features from previous revisions of this Recommendation | International Standard have been deprecated. Writers shall not include these features when creating or updating files. Readers shall ignore the contribution of those features when determining whether they can or cannot read the file.

The type of a Reader Requirements box shall be 'rreq' (0x7272 6571'). The contents of the Reader Requirements box is as follows (see Figure M.7):

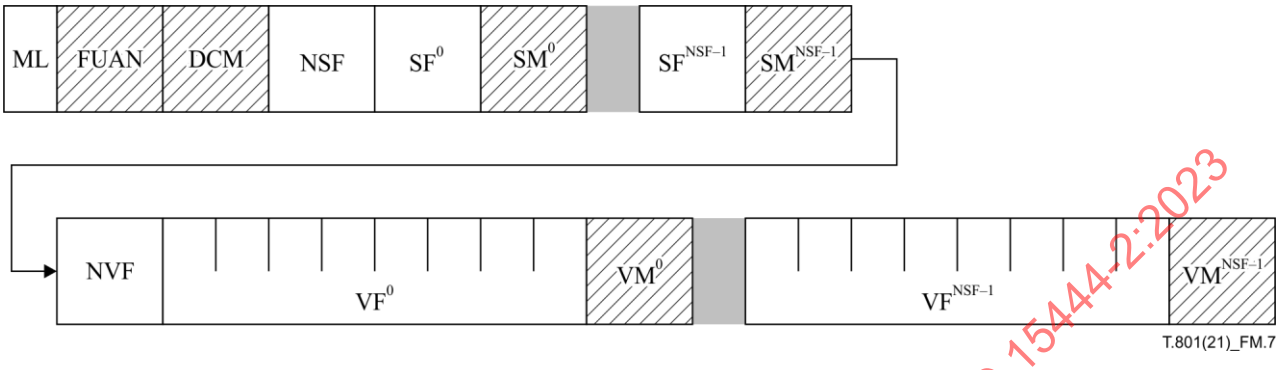

Figure M.7 – Organization of the contents of the Reader Requirements box

ML: Mask length. This field is a byte that specifies the number of bytes used for the compatibility masks. This field is encoded as a 1-byte unsigned integer.

FUAM: Fully Understand Aspects mask. This field is the mask describing the Fully Understand Aspects expression. This field is specified as a big endian integer of the size as specified by the ML field.

DCM: Display Contents mask. This field is the mask describing the expression to display the image correctly. This field is specified as a big endian integer of the size as specified by the ML field.

NSF: Number of standard flags. This field specifies the number of standard feature flags contained within the Reader Requirements box. The value of this field shall be equal to the number of SF^i fields found within the Reader Requirements box. This field is encoded as a 2-byte big endian unsigned integer.

SF^i: Standard flag. This field specifies a standard feature flag. The number of SFi fields shall be equal to the value of the NSF field. This field is encoded as a 2-byte big endian unsigned integer. Legal values of this field are specified in Table M.14 and in extensions to this standard.

SM^i: Standard mask. This field specifies the compatibility mask for the feature specified by SF^i. This field is specified as a big endian integer of the size as specified by the ML field.

NVF: Number of vendor features. This field specifies the number of vendor features specified in the Reader Requirements box. The value of this field shall be equal to the number of VF^i fields in the Reader Requirements box. This field is encoded as a 2-byte big endian unsigned integer.

VF^i: Vendor feature. This field specifies one vendor defined feature that is used in this JPX file. This field is encoded as a 128-bit UUID. Information about the feature specified by this UUID can be specified using the UUID Info box as defined in the JP2 file format.

VM^i: Vendor mask. This field specifies the compatibility mask for the feature specified by VF^i. This field is specified as a big endian integer of the size as specified by the ML field.

Table M.14 – Legal values of the SF^i field

Value	Meaning
0	File not completely understood
1	Codestream contains no extensions
2	Contains multiple composition layers
3	Deprecated
4	JPEG 2000 Core coding system Profile 1 codestream as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, Table A.45
5	Unrestricted JPEG 2000 Core coding system codestream as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1
6	Unrestricted JPEG 2000 Extensions codestream as defined in this Recommendation | International Standard
7	JPEG codestream as defined in ISO/IEC 10918-1
8	Deprecated

Table M.14 – Legal values of the SF^i field

Value	Meaning
9	Non-premultiplied opacity channel
10	Premultiplied opacity channel
11	Chroma-key based opacity
12	Deprecated
13	Fragmented codestream where all fragments are in the file and in order
14	Fragmented codestream where all fragments are in the file but are out of order
15	Fragmented codestream where not all fragments are within the file but all are in locally accessible files
16	Fragmented codestream where some fragments may be accessible only through a URL specified network connection
17	Compositing required to produce rendered result from multiple compositing layers
18	Deprecated
19	Deprecated
20	Deprecated
21	At least one compositing layer consists of multiple codestreams
22	Deprecated
23	Colourspace transformations are required to combine compositing layers; not all compositing layers are in the same colourspace
24	Deprecated
25	Animation
26	First animation layer does not cover entire rendered result area
27	Deprecated
28	Reuse of animation layers
29	Deprecated
30	Some animated frames are non-persistent
31	Deprecated
32	Rendered result involves scaling within a layer
33	Rendered result involves scaling between layers
34	ROI metadata
35	IPR metadata
36	Content metadata
37	History metadata
38	Creation metadata
39	JPX digital signatures
40	JPX checksums
41	Desired Graphic Arts reproduction specified
42	Deprecated
43	Deprecated
44	Compositing layer uses Any ICC profile
45	Deprecated
46	Deprecated
47	BiLevel 1 enumerated colourspace
48	BiLevel 2 enumerated colourspace
49	YCbCr 1 enumerated colourspace
50	YCbCr 2 enumerated colourspace
51	YCbCr 3 enumerated colourspace
52	PhotoYCC enumerated colourspace
53	YCCK enumerated colourspace
54	CMY enumerated colourspace
55	CMYK enumerated colourspace
56	CIELab enumerated colourspace with default parameters

Table M.14 – Legal values of the SF^i field

Value	Meaning
57	CIELab enumerated colourspace with non-default parameters
58	CIEJab enumerated colourspace with default parameters
59	CIEJab enumerated colourspace with non-default parameters
60	e-sRGB enumerated colourspace
61	ROMM–RGB enumerated colourspace
62	Non-square samples
63	Deprecated
64	Deprecated
65	Deprecated
66	Deprecated
67	GIS metadata XML box
68	JPSEC extensions in codestream as specified by ISO/IEC 15444-8
69	JP3D extensions in codestream as specified by ISO/IEC 15444-10
70	Deprecated
71	e-sYCC enumerated colourspace
72	JPEG 2000 Extensions codestream as restricted by baseline conformance requirements in clause M.9.2.3
73	YPbPr(1125/60) enumerated colourspace
74	YPbPr(1250/50) enumerated colourspace
75	Codestream contains a JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) compliant bitstream.
76	Codestream contains a Sub-baseline profile JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) compliant bitstream.
77	Codestream contains a Baseline profile JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) compliant bitstream.
78	Codestream contains a Main profile JPEG XR (Rec. ITU-T T.832 | ISO/IEC 29199-2) compliant bitstream.
79	Codestream contains an Advanced profile JPEG XR (Rec. ITU-T 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 (Annex P)
85	Compositing layer uses scRGB gray scale (IEC 61966-2-2 based) enumerated colourspace
86	JPEG 2000 codestream capabilities specified in Rec. ITU-T T.814 | ISO/IEC 15444-15

The format of the contents of the Reader Requirements box is given it Table M.15.

Table M.15 – Format of the contents of the Reader Requirements box

Field name	Size (bits)	Value
ML	8	1, 2, 4 or 8
FUAM	8 · ML	Variable
DCM	8 · ML	Variable
NSF	16	0-65 535
SF^i	16	0-65 535
SM^i	8 · ML	Variable
NVF	16	0-65 535
VF^i	128	Variable
VM^i	8 · ML	Variable

M.11.2 Data Reference box

The Data Reference box contains an array of URL's which are referenced by this file. Many of these references will be from Fragment Table boxes, specifying the location of the codestream fragments. Other references will be from Cross-Reference boxes. A JPX file shall contain zero or one Data Reference boxes, and that Data Reference box shall be at the top level of the file; it shall not be in any superboxes.

The Data Reference box is not a superbox because it does not contain only boxes.

The type of the Data Reference box shall be 'dtbl' (0x6474 626C), and its contents shall be as follows (see Figure M.8):

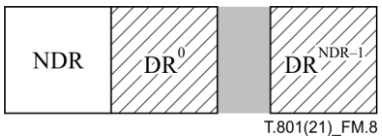

Figure M.8 – Organization of the contents of a Data Reference box

NDR: Number of data references. This field specifies the number of data references, and thus the number of URL boxes contained within this Data Reference Box.

DR^i: Data Reference URL. This field contains a Data Entry URL box, as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause I.7.3.2. However, in this context, the Location field in the box is not specific to UUID Info Boxes. The meaning of the URL is specified in the context of the box that refers to the particular entry in the Data Reference box.

The indices of the elements in the array of DR^i fields is 1 based; a data reference of 1 in a DR^i field within a Fragment List box specifies the first Data Reference URL contained within the Data Reference Box. A data reference value of 0 is a special case that indicates that the reference is to data contained within this JPX file itself.

The format of the contents of the Data Reference box is given in Table M.16.

Table M.16 – Format of the contents of the Data Reference box

Field name	Size (bits)	Value
NDR	16	0-65 535
DR^i	Variable	Variable

M.11.3 Fragment Table box (superbox)

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.

The type of the Fragment Table box shall be 'ftbl' (0x6674 626C), and its contents shall be as follows (see Figure M.9):

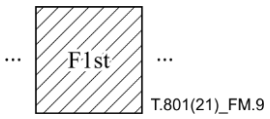

Figure M.9 – Organization of the contents of a Fragment Table box

Flst: Fragment List. This field contains a Fragment List box as specified in clause M.11.3.1.

M.11.3.1 Fragment List box

The Fragment List box specifies the location, length and order of each of the fragments that, once combined, form a valid and complete data stream. Depending on what box contains this particular Fragment List box, the data stream forms either a codestream (if the Fragment List box is contained in a Fragment Table box) or shared header or metadata (if the Fragment List box is contained in a Cross-Reference box).

If this Fragment List box is contained within a Fragment Table box (and thus specifies the location of a codestream), then the first offset in the fragment list shall point directly to the first byte codestream data; it shall not point to the header of the box containing the first codestream fragment.

If this Fragment List box is contained within a Cross-Reference box (and thus specifies the location of shared header or metadata), then the first offset in the fragment list shall point to the first byte of the contents of the referenced box; it shall not point to the header of the referenced box. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox.

For all other offsets in the Fragment List box, the offsets shall point directly to the first byte of the fragment data and not to the header of the box that contains that fragment.

In addition, an offset within any Fragment List shall not point into a Binary Filter box. If the JPX file does contain one or more Binary Filter boxes, then all offsets in all Fragment list boxes shall be interpreted with respect to the length of the Binary Filter boxes, as stored in the file, not the length of the data after the application of the filter.

The type of the Fragment List box shall be 'flst' (0x666C 7374) and it shall have the following contents (see Figure M.10):

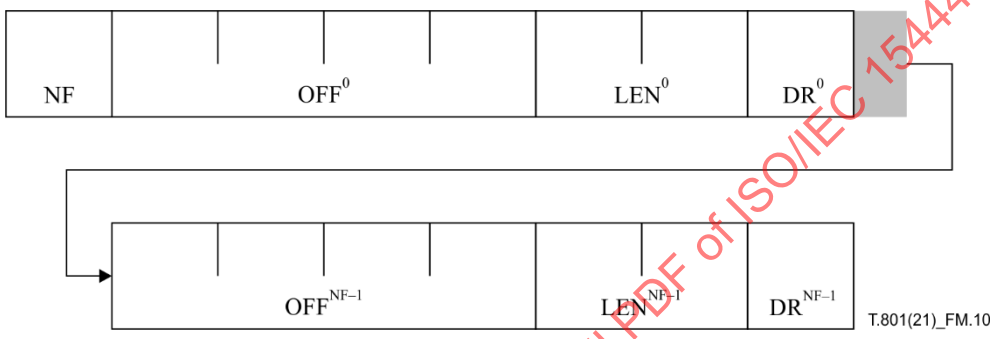

Figure M.10 – Organization of the contents of a Fragment List box

NF: Number of fragments. This field specifies the number of fragments used to contain the data stream. The number of {OFF, LEN, DR} tuples in the Fragment list box shall be the same number as the value of the NF field.

OFF^i: Offset. This field specifies the offset to the start of the fragment in the specified file. The offset is relative to the first byte of the file (for example, the first byte of the length field of the JPEG 2000 signature box header for a JPX file). This field is encoded as a 64-bit unsigned integer.

LEN^i: Length of fragment. This field specifies the length of the fragment. This value includes only the actual data and not any headers of an encapsulating box. This field is encoded as a 32-bit unsigned integer.

DR^i: Data reference. This field specifies the data file or resource that contains this fragment. If the value of this field is zero, then the fragment is contained within this file. If the value is not zero, then the fragment is contained within the file specified by this index into the array of DR^i fields in the Data Reference box, where an index value of 1 indicates the first element in the array. This field is encoded as a 16-bit unsigned integer.

The format of the contents of the Fragment List box is given in Table M.17.

Table M.17 – Format of the contents of the Fragment List box

Parameter	Size (bits)	Value
NF	16	0-65 535
OFF^i	64	12-(2^{64}–1)
LEN^i	32	0-(2^{32}–1)
DR^i	16	0-65 535

M.11.4 Cross-Reference box

If a JPX file contains multiple codestreams or compositing layers, it may be useful to share header and metadata information between those codestreams or compositing layers to minimize file size. One mechanism to share such data is to place a cross-reference to the actual metadata or header box into the Codestream Header or Compositing Layer Header

box in place of the actual data. This is done using a Cross-Reference box. A JPX file may contain zero or more Cross-Reference boxes, and the Cross-Reference boxes shall be found only within Codestream Header boxes, Compositing Layer Header boxes, or Association boxes. Also, a Cross-Reference box shall not point to another Cross-Reference box. Also, because the Cross-Reference box contains a field followed by a box, the Cross-Reference box is not a superbox.

The type of the Cross-Reference box shall be 'cref' (0x6372 6566) and it shall have the following contents (see Figure M.11):

Figure M.11 – Organization of the contents of a Fragment table box

Rtyp: Referenced box type. This field specifies the actual type (as would be found in the TBox field in an actual box header) of the box referenced by this Cross-Reference box. However, a reader shall not attempt to locate a physically stored box header for the box represented by this cross-reference box, as it is legal to use a Cross-Reference box to create a new box that is not contiguously contained in other locations within this or other files, and thus the box header will not exist.

flst: Fragment List box. This box specifies the actual locations of the fragments of the referenced box. When those fragments are concatenated, in order, as specified by the Fragment List box definition, the resulting byte-stream shall be the contents of the referenced box and shall not include the box header fields. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox. The format of the Fragment List box is specified in clause M.11.3.1.

The format of the contents of the Cross-Reference box is given in Table M.18.

Table M.18 – Format of the contents of the Cross-Reference box

Parameter	Size (bits)	Value
Rtyp	32	0-(2^{32}–1)
flst	Variable	Variable

M.11.5 JP2 Header box (superbox)

The JP2 Header box is syntactically unchanged from the structures defined in the JP2 file format. However, if the JPX file contains multiple codestreams or multiple compositing layers, then any boxes contained within the JP2 Header box shall be considered as defaults for all codestreams and compositing layers. For example, if a Compositing Layer Header box does not specify a Colourspace specification, then a reader shall apply the Colourspace Specification contained within the JP2 Header box to that particular compositing layer.

Also, if the JP2 Header box specified default information for any codestreams, then the semantic relationship of the Image Header box and Bits Per Component box contained within the JP2 Header box shall follow the rules defined in clause M.11.5.1 and M.11.5.2 respectively.

Also, the JPX file format allows the JP2 Header box to be located anywhere at the top-level of the file (but not within any superbox). However, certain fallback positions, such as the JPX baseline definition, may restrict the placement of this box. In addition, if this file does not require the JP2 Header box to meet the requirements of a fallback position, nor does it use the JP2 Header box to specify default information for multiple compositing layers or codestreams, then this box may be omitted from the file.

M.11.5.1 Image Header box

The format and structure of the Image Header box is identical to that defined in clause I.5.3.1 in Rec. ITU-T T.800 | ISO/IEC 15444-1 in the JP2 file format. However, the additional values of the fields within that box are defined for the JPX file format. In a JPX file, this box may be found either within the JP2 Header box or within a Codestream Header box.

The type of the Image Header box shall be 'ihdr' (0x6968 6472) and contents of the box shall have the following format (see Figure M.12):

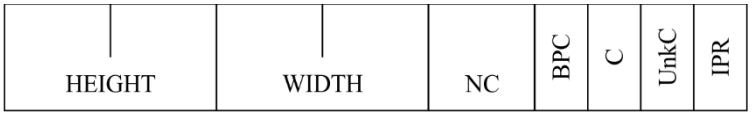

Figure M.12 – Organization of the contents of an Image Header box

HEIGHT: Image area height. The value of this field is identical to that defined for the JP2 file format.

WIDTH: Image area width. The value of this field is identical to that defined for the JP2 file format.

NC: Number of components. The value of this field is identical to that defined for the JP2 file format.

BPC: Bits per component. This parameter identifies the bit depths and signed/unsigned characteristics of the component sample values reconstructed from the codestream, stored as a 1-byte field as defined in Table M.20. If the components vary in bit depth or signed/unsigned characteristics, 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 clause M.11.5.2.

If the bit depth is the same for all components, then this parameter specifies that bit depth and shall be equivalent to the bit depth specified within the codestream using the data structures defined for that particular codestream format. If the components vary in bit depth, then the value of this field shall be 255, and the superbox that contains this Image Header box (either the JP2 Header box or a Codestream Header box) shall contain a Bits Per Component box defining the bit depth of each component (as defined in clause I.5.3.2 in Rec. ITU-T T.800 | ISO/IEC 15444-1 in the JP2 file format). Components should be considered to have different bit depths if either the magnitude or sign of the bit depth of the components differ.

The low 7-bits of the value indicate the bit depth of the components. The high-bit indicates whether the components are signed or unsigned. If the high-bit is 1, then the components contain signed values. If the high-bit is 0, then the components contain unsigned values.

C: See Table M.19.

Table M.19 – Legal C values

Value	Meaning
0	**Uncompressed.** Picture data is stored in component interleaved format, encoded at the bit depth as specified by the BPC field. This value is only permitted for codestreams where all components are encoded at the same bit depth. When the bit depth of each component is not a multiple of 8, component values shall be packed into bytes so that no bits are unused between components. However, the value of the first component of each sample shall begin on a byte boundary and padding bits having value zero shall be inserted after the last component of the sample as necessary to fill out the remaining bits to the next byte boundary. When multiple component values are packed into a byte, the first component shall appear in the most significant bits of the byte. When a component is larger than a byte, its most significant bits shall appear in earlier bytes.
1	**Rec. ITU-T T.4, the basic algorithm known as MH (Modified Huffman).** This value is only permitted for bi-level images.
2	**Rec. ITU-T T.4, commonly known as MR (Modified READ).** This value is only permitted for bi-level images.
3	**Rec. ITU-T T.6, commonly known as MMR (Modified Modified READ).** This value is only permitted for bi-level images.
4	**Rec. ITU-T T.82 | ISO/IEC 11544.** Commonly known as JBIG. This value is only permitted for bi-level images.
5	**Rec. ITU-T T.81 | ISO/IEC 10918-1 or Rec. ITU-T T.84 | ISO/IEC 10918-3**. Commonly known as JPEG. This compressed image stream shall conform to the syntax of interchange format for compressed image data as specified in the aforementioned standards. This value is only permitted for continuous tone, greyscale or colour images.
6	**JPEG-LS.**
7	**JPEG 2000 compression (as defined by ISO/IEC 15444).**
8	**JBIG2.**
9	**Rec. ITU-T T.82 | ISO/IEC 11544.** Commonly known as JBIG. This value is permitted for any image permitted by the JBIG standard.
10	**Rec. ITU-T T.45** (run length coding, used in Rec. ITU-T T.805 | ISO/IEC 15444-6)
11	**JPEG XR** (as defined by Rec. ITU-T T.832 | ISO/IEC 29199-2)
12	**JPEG XS** (as defined in ISO/IEC 21122-1)
	All other values reserved.

UnkC: Colourspace Unknown. The value of this field is identical to that defined for the JP2 file format.

IPR: Intellectual Property. This parameter indicates whether this JPX file contains intellectual property rights information that is associated with the codestream or codestreams described by this Image Header box. If the value of this field is 0, those codestreams do not have associated rights information. If the value is 1, then those codestreams do have associated rights information. Other values are reserved.

Table M.20 – 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

NOTE – For codestreams conforming to Rec. ITU-T 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 Rec. ITU-T 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 BPCi, 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

M.11.5.2 Bits Per Component box

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.

The structure of this box is identical to that defined in clause I.5.3.2 in the JP2 file format specified in Rec. ITU-T T.800 | ISO/IEC 15444-1.

The Format of the contents of the Image Header box is given in Table M.21.

Table M.21 – Format of the contents of the Image Header box

Field name	Size (bits)	Value
HEIGHT	32	1-(2^{32}–1)
WIDTH	32	1-(2^{32}–1)
NC	16	1-16 384
BPC	8	See Table M.20
C	8	Compression Type, see Table M.19
UnkC	8	0-1
IPR	8	0-1

M.11.6 Codestream Header box (superbox)

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 clause M.11.21). All top-level Codestream Header boxes shall 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 clause 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).

The type of a Codestream Header box shall be 'jpch' (0x6A70 6368). The contents of a Codestream Header box is as follows (see Figure M.13):

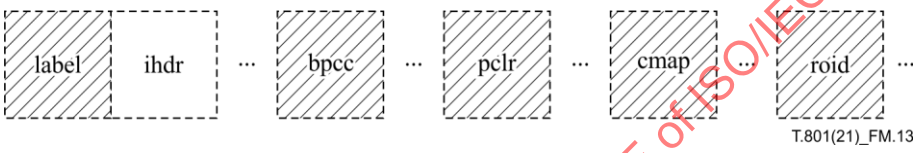

Figure M.13 – Organization of the contents of a Codestream Header box

label: Label box. This box specifies a label for this codestream. Its structure is specified in clause M.11.13.

ihdr: Image Header box. This box specifies information about this codestream, such as its height and width. Its structure is specified in clause M.11.5.1. If the JP2 Header box contains an Image Header box that accurately specifies this codetream, then it is not required that this Codestream Header box contain an Image Header box. Otherwise, this Codestream Header box shall contain an Image Header box. In addition, if the IPR flag in the Image Header box is set to 0, indicating no intellectual property rights information is specified for this codestream, then this Codestream Header box shall not contain an IPR box, and the reader shall not apply the contents of an IPR box at the top level of the file to this codestream.

bpcc: Bits Per Component box. This box specifies the bit depth of each component in the codestream after decompression. Its structure is specified in clause M.11.5.2.

pclr: Palette box. This box defines the palette to use to create multiple components from a single component. Its structure is specified in clause I.5.3.4 in Rec. ITU-T T.800 | ISO/IEC 15444-1 of the JP2 file format.

cmap: Component Mapping box. This box defines how image channels are identified from the actual components in the codestream. Its structure is specified in clause I.5.3.5 in Rec. ITU-T T.800 | ISO/IEC 15444-1 of the JP2 file format.

roid: ROI Description box. This box describes regions of interest within this codestream. These ROIs may or may not be directly associated with coded ROIs in the codestream. Its structure is defined in clause M.11.16.

The Codestream Header box may also contain other metadata boxes, including an IPR box, or cross-references to other boxes. If the Codestream Header contains a cross-reference, then the box pointed to by the cross-reference shall be considered as if it was physically stored in this Codestream Header box.

Also, if any of these boxes are contained within the JP2 header box and are not contained within this Codestream Header box, then those boxes should also be applied to this codestream.

M.11.7 Compositing Layer Header box (superbox)

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 clause M.11.21). All top-level Compositing Layer Header boxes shall 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 clause M.11.21.

The type of a Compositing Layer Header box shall be 'jplh' (0x6A70 6C68). The contents of a Compositing Layer Header box is as follows (see Figure M.14):

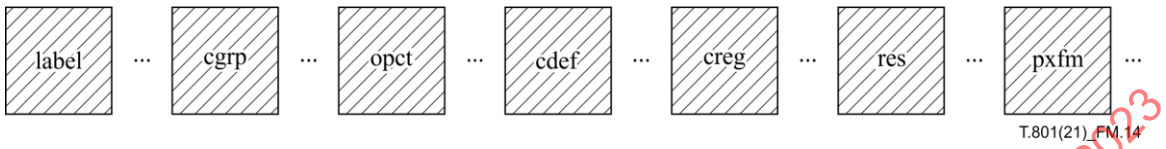

Figure M.14 – Organization of the contents of a Compositing Layer Header box

label: Label box. This box specifies a label for this compositing layer. Its structure is specified in clause M.11.13.

cgrp: Colour Group box. This box contains the complete colourspace specification (represented by a sequence of colour specification boxes) for this compositing layer. Its structure is specified in clause M.11.7.1. If neither this box nor a cross-reference to another Colour Group box are found within the Compositing Layer Header box, then the default value of the colourspace specification for this compositing layer shall be the set of individual Colour Specification boxes found within the JP2 Header box. These Colour Specification boxes shall not be encapsulated within a Colour Group box.

opct: Opacity box. This box specifies that this compositing layer uses a simple opacity mode. Its structure is specified in clause M.11.7.6. If the Compositing Layer Header box contains an Opacity box, then it shall not contain a Channel Definition box, and any default Channel Definition box in the JP2 Header box shall be ignored for this compositing layer.

cdef: Channel Definition box. This box defines the channels in the image. Its structure is specified in clause I.5.3.6 in Rec. ITU-T T.800 | ISO/IEC 15444-1 of the JP2 file format. This box shall not be found if this Compositing Layer Header box contains an Opacity box.

creg: Codestream Registration box. This box specifies the spatial registration between the codestreams in this compositing layer. Its structure is specified in clause M.11.7.7. If any Compositing Layer Header box contains a Codestream Registration box, then every Compositing Layer Header box shall contain a Codestream Registration box.

res: Resolution box. This box specifies the capture and default display resolutions of the image. Its structure is specified in clause I.5.3.7 in Rec. ITU-T T.800 | ISO/IEC 15444-1 of the JP2 file format.

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 clause clause M.11.7.8.

The Compositing Layer Header box may also contain other metadata boxes, including an IPR box, or cross-references to other boxes. If the Compositing Layer Header contains a cross-reference, then the box pointed to by the cross-reference shall be considered as if it was physically stored in this Compositing Layer Header box.

Also, if any of these boxes are contained within the JP2 header box and are not contained within this Compositing Layer Header box, then those boxes should also be applied to this compositing layer.

M.11.7.1 Colour Group box (superbox)

A Colour group box contains a set of related, equivalent, colour specification methods. When interpreting the colourspace of a codestream, any colour specification method contained within the specified Colour Group box may be used. This box shall be found only within a Compositing Layer Header box. This encapsulation reduces the storage overhead of sharing an entire set of colour specifications between layers.

A Colour Group box (or the JP2 Header box) shall not contain multiple Colour Specifications boxes with a METH value of 1 (Enumerated method), or multiple boxes with a METH value of 2 (Restricted ICC method). A single colour group may contain multiple Colour Specification boxes with a METH value of 3 (Any ICC method) or 4 (Vendor Colour

method). Multiple ICC profiles (of the unrestricted variety) may be used to specify a particular colourspace with varying degrees of complexity (1D LUT's vs 3D LUT's), and multiple Vendor Colour methods may be used to specify multiple non-ICC based representations of the colourspace.

The JPX file may contain zero Colour Group boxes, which indicates that all compositing layers are in the colourspace specified within the JP2 Header Box (through a set of Colour Specification boxes stored directly within the JP2 Header boxes and not encapsulated within a Colour Group box).

However, if the file does not contain a colourspace specification within the JP2 Header Box (or does not contain the JP2 Header Box), then the JPX file shall contain at least one Colour Group box.

The type of a Colour Group box shall be 'cgrp' (0x6367 7270). The contents of a Colour Group box is as follows (see Figure M.15):

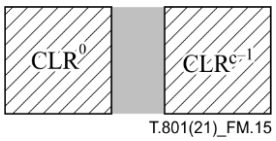

Figure M.15 – Organization of the contents of a Colour Group box

CLR^i: Colour Specification Box. This Colour Specification box specifies one method by which the colourspace of a particular codestream can be interpreted. The format of the Colour Specification box is specified in clause M.11.7.2

M.11.7.2 Colour Specification box

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 clause M.11.7.8) and associated to colours according to the Channel Definition Box (see clause M.11.7.5). This association is to be interpreted using the value MAX^i for each colour channel i, as given by Table M.26, in combination with the relevant colourspace definition.

In Table M.24bis, BPC^i is used to denote the value of the B^j field of the Palette Box (see clause I.5.3.4 of Rec. ITU-T 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 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 F^i in this table is either the format indicated in the Pixel Format Box (see clause 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.

$$d^i = Dmax^i \cdot x^i / MAX^i,$$

Here, Dmax^i is the maximum input value associated with the relevant ICC tone reproduction curve and MAX^i depends on the pixel format F^i as given by Table M.24bis.

For enumerated 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 clause M.11.7.4.

NOTE 1 – 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

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

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

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

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 achromatic level, if this interpretation is applicable and all channel values are uniquely defined in this case. Otherwise, the pixel format F^i and the channel precision 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 2 – The CIELab and CIEJab colourspaces uses both positive and negative values to represent colours. However, since clause 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 clause will not take place. Instead, clause 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 $Dzero^0$=0, and $Dzero^1$=$Dzero^2$=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.

Colour Specification boxes may be found in either the JP2 Header box or in Colour Group boxes. In total, a JPX file may contain multiple Colour Specification boxes, and either the JP2 Header box or a particular Colour Group box may contain multiple Colour Specification boxes. However, all JPX files shall contain at least one Colour Specification box.

The box type and binary structure of a Colour Specification box is identical to that defined in the JP2 file format. However, to clarify the extensibility of the box with respect to defining new colour specification methods, the way in which it is described is changed within JPX. The contents of a Colour Specification box is as follows (see Figure M.16):

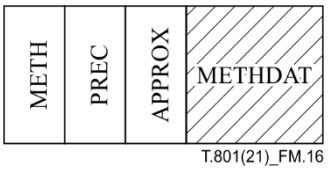

Figure M.16 – Organization of the contents of a Colour Specification box

METH: Specification method. This field specifies the method used by this Colour Specification box to define the colourspace of the decompressed image. This field is encoded as a 1-byte unsigned integer. The legal values of the METH field are as shown in Table M.22.

Table M.22 – Legal METH values

Value	Meaning
1	**Enumerated method**. This Colour Specification box indicates that the colourspace of the codestream is specified by an enumerated integer code. The definition of the format of this method is identical to the Enumerated Method in JP2. However, the JPX file format defines additional enumerated values as specified in clause M.11.7.3.1, as well as additional parameters for some enumerated colourspaces as specified in clause M.11.7.4.
2	**Restricted ICC method**. This Colour Specification box indicates that the colourspace of the codestream is specified by an embedded ICC profile of restricted type. The definition of and format of this method is identical to the Restricted ICC method defined in the JP2 file format, clause I.5.3.3 in Rec. ITU-T T.800 | ISO/IEC 15444-1.
3	**Any ICC method**. This Colour Specification box indicates that the colourspace of the codestream is specified by an embedded input ICC profile. Contrary to the Restricted ICC method defined in the JP2 file format, this method allows for any input ICC profile, defined by ISO/IEC 15076-1. The binary format of the METHDAT field is specified in clause M.11.7.3.2.
4	**Vendor Colour method**. This Colour Specification box indicates that the colourspace of the codestream is specified by a unique vendor defined code. The binary format of the METHDAT field is specified in clause M.11.7.3.3.
5	**Parameterized colourspace**. This Colour Specification box indicates that the colourspace of the codestream is parameterized as specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. The binary format of the METHDAT field is specified in clause M.11.7.3.4.
	All other values reserved. For any value of the METH field, the length of the METHDAT field may not be 0, and applications shall not expect that the APPROX field be the last field in the box if the value of the METH field is not understood. In this case, a conforming reader shall ignore the entire Colour Specification box.

PREC: Precedence. This field specifies the precedence of this Colour Specification box, with respect to the other Colour Specification boxes within the same Colour Group box, or the JP2 Header box if this Colour Specification box is in the JP2 Header box. It is suggested, but not required, that conforming readers use the colour specification method that is supported with the highest precedence. This field is specified as a signed 1-byte integer.

APPROX: Colourspace approximation. This field specifies the extent to which this colour specification method approximates the "correct" definition of the colourspace. An example of approximation of a colourspace specification may be increased quantization in look-up tables or rounding in matrix coefficients. This field is specified as 1-byte unsigned integer. Legal values of this field are as follows:

Contrary to the APPROX field in a JP2 file (a file with "jp2\040" in the BR field in the File Type box), a value of 0 in the APPROX field is illegal in a JPX file (a file with "jpx\040" in the BR field in the File Type box). JPX writers are required to properly indicate the degree of approximation of the colour specification to the correct definition of the colourspace. This does not specify if the writer of the file knew the actual colourspace of the image data. If the actual colourspace is unknown, then the value of the UnkC field in the Image Header box shall be set to 1 and the APPROX field shall specify the degree to which this Colour Specification box matches the correct definition of the assumed or target colourspace.

In addition, high values of the APPROX field (indicating poor approximation) shall not be used to hide that the multiple Colour Specification boxes in either a Colour Group box or the JP2 Header box actually represent different colourspaces; the specification of multiple different colourspaces within a single Colour Group box is illegal.

Table M.23 – Legal APPROX values

Value	Meaning
1	This colour specification method accurately represents the correct definition of the colourspace
2	This colour specification method approximates the correct definition of the colourspace with exceptional quality
3	This colour specification method approximates the correct definition of the colourspace with reasonable quality
4	This colour specification method approximates the correct definition of the colourspace with poor quality
	All other values reserved

The Format of the contents of the Colour Specification box is given in Table M.24.

Table M.24 – Format of the contents of the Colour Specification box

Field name	Size (bits)	Value
METH	8	1-4
PREC	8	–128-127
APPROX	8	1-4
METHDAT	Variable	Variable

Table M.24bis – Nominal maximum sample values

Pixel Format F^i and Channel Signedness	MAX^i
unsigned, integer ($BPC^i < 128$)	$2^{BPC+1} - 1$
signed, integer ($BPC^i \geq 128$)	$2^{BPC-128} - 1$
unsigned, fixed point ($BPC^i < 128$)	1.0
signed, fixed point ($BPC^i \geq 128$)	1.0
unsigned, floating point ($BPC^i < 128$)	1.0
signed, floating point ($BPC^i \geq 128$)	1.0
NOTE 1 – It is colourspace dependent 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 2 – Fixed point values can be interpreted as integer values after scaling by 2^{Qi}, where the value of Qi is given by the low 12 bits of the 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^{Qi}.	

M.11.7.3 METHDAT field specifications in the Colour Specification box

The following subclauses define the fields and values that make up the METHDAT field for each defined colour specification method.

M.11.7.3.1 METHDAT values for the Enumerated method

The contents of the METHDAT field for Colour Specification boxes using the Enumerated method is defined as follows (see Figure M.17):

Figure M.17 – Organization of the contents of the METHDAT field for the Enumerated method

EnumCS: Enumerated colourspace. This field specifies the colourspace of the image using an integer code. To correctly interpret the colour of an image using an enumerated colourspace, the application has to know the definition of that colourspace internally. This field contains a 4-byte big endian unsigned integer value indicating the colourspace of the image. Valid EnumCS values are those values defined for the Enumerated method in the JP2 file format and the values defined as follows (Table M.25).

Table M.25 – Additional legal EnumCS values

Value	Meaning
0	**Bi-level**: This value shall be used to indicate bi-level images. Each image sample is one bit: 0 = white, 1 = black.
1	**YC_bC_r(1)**: This is a format often used for data that originated from a video signal. The colourspace is based on Rec. ITU-R BT.709-4. The valid ranges of the YC_bC_r components in this space is limited to less than the full range that could be represented given an 8-bit representation. Rec. ITU-R BT.601-5 specifies these ranges as well as defines a 3 × 3 matrix transformation that can be used to convert these samples into RGB.
3	**YC_bC_r(2)**: This is the most commonly used format for image data that was originally captured in RGB (uncalibrated format). The colourspace is based on Rec. ITU-R BT.601-5. The valid ranges of the YC_bC_r components in this space is [0, 255] for Y, and [–128, 127] for C_b and C_r (stored with an offset of 128 to convert the range to [0, 255]). These ranges are different from the ones defined in Rec. ITU-R BT.601-5. Rec. ITU-R BT.601-5 specifies a 3 × 3 matrix transformation that can be used to convert these samples into RGB.
4	**YC_bC_r(3)**: This is a format often used for data that originated from a video signal. The colourspace is based on Rec. ITU-R BT.601-5. The valid ranges of the YC_bC_r components in this space is limited to less than the full range that could be represented given an 8-bit representation. Rec. ITU-R BT.601-5 specifies these ranges as well as defines a 3 × 3 matrix transformation that can be used to convert these samples into RGB.
9	**PhotoYCC**: This is the colour encoding method used in the Photo CD™ system. The colourspace is based on Rec. ITU-R BT.709 reference primaries. Rec. ITU-R BT.709 linear RGB image signals are transformed to non-linear R'G'B' values to YCC corresponding to Rec. ITU-R BT.601-5. Details of this encoding method can be found in Kodak Photo CD products, *A Planning Guide for Developers*, Eastman Kodak Company, Part No. DC1200R and also in Kodak Photo CD Information Bulletin PCD045.
11	**CMY**: The encoded data consists of samples of Cyan, Magenta and Yellow samples, directly suitable for printing on typical CMY devices. A value of 0 shall indicate 0% ink coverages, whereas a value of 2^{BPS}–1 shall indicate 100% ink coverage for a given component sample.
12	**CMYK**: As CMY above, except that there is also a black (K) ink component. Ink coverage is defined as above.
13	**YCCK**: This is the result of transforming original CMYK type data by computing R = (2^{BPS}–1)–C, G = (2^{BPS}–1)–M, and B = (2^{BPS}–1)–Y, applying the RGB to YCC transformation specified for YC_bC_r (2) above, and then recombining the result with the unmodified K-sample. This transformation is intended to be the same as that specified in Adobe Postscript.
14	**CIELab**: The CIE 1976 (L*a*b*) colourspace. A colourspace defined by the CIE (Commission Internationale de l'Eclairage), having approximately equal visually perceptible differences between equally spaced points throughout the space. The three components are L*, or Lightness, and a* and b* in chrominance. For this colourspace, additional Enumerated parameters are specified in the EP field as specified in clause M.11.7.4.1
15	**Bi-level(2)**: This value shall be used to indicate bi-level images. Each image sample is one bit: 1 = white, 0 = black.

Table M.25 – Additional legal EnumCS values

Value	Meaning
18	**sYCC** as defined by IEC 61966-2-1. NOTE – Use of the ICT or RCT specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 Annex G is unhelpful with sYCC image data. See Rec. ITU-T T.800 | ISO/IEC 15444-1, Clause J.15, for guidelines on handling YCC codestreams.
19	**CIEJab**: As defined by CIE Colour Appearance Model 97s, CIE Publication 131. For this colourspace, additional Enumerated parameters are specified in the EP field as specified in clause M.11.7.4.2.
20	**e-sRGB**: As defined by PIMA 7667.
21	**ROMM-RGB**: As defined by ISO 22028-2.
22	**YPbPr(1125/60)**: This is the well-known colour space and value definition for the HDTV (1125/60/2:1) system for production and international program exchange specified by Rec. ITU-R BT.709-3. The Recommendation specifies the colour space conversion matrix from RGB to YPbPr(1125/60) and the range of values of each component. The matrix is different from the 1250/50 system. In the 8-bit/component case, the range of values of each component is [1, 254], the black level of Y is 16, the achromatic level of Pb/Pr is 128, the nominal peak of Y is 235, and the nominal extremes of Pb/Pr are 16 and 240. In the 10-bit case, these values are defined in a similar manner.
23	**YPbPr(1250/50)**: This is the well-known colour space and value definition for the HDTV (1250/50/2:1) system for production and international program exchange specified by Rec. ITU-R BT.709-3. The Recommendation specifies the colour space conversion matrix from RGB to YPbPr(1250/50) and the range of values of each component. The matrix is different from the 1125/60 system. In the 8-bit/component case, the range of values of each component is [1, 254], the black level of Y is 16, the achromatic level of Pb/Pr is 128, the nominal peak of Y is 235, and the nominal extremes of Pb/Pr are 16 and 240. In the 10-bit case, these values are defined in a similar manner.
24	**e-sYCC**: e-sRGB based YCC colourspace as defined by PIMA 7667:2001, Annex B.
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.
	All other values reserved.

The generic RGB and grayscale spaces from the SPIFF file format are explicitly not included. Applications wishing to transcode SPIFF images using colourspaces 8 and 10 should specify, within the JPX file, the colourspace definition that a reader shall use to unambiguously interpret the image data. In many cases, this will be the sRGB or sRGB-greyscale spaces from JP2. In addition, the file writer should set the UnkC field in the Image Header box indicating that the actual colourspace is not known.

EP: Enumerated parameters. This field contains a series of parameters that augment the generic colourspace definition specified by EnumCS. Together, the EnumCS and EP fields describe the colourspace and how that colour data has been encoded in the JPX file. For example, the CIELab colourspace as described by Rec. ITU-T T.42 requires several parameters to describe the ITU encoding of the colour data. The format and value of the EP field is defined individually for each EnumCS as required. If a value of EP is not defined for a particular value of EnumCS, then the length of the EP field for that EnumCS value shall be zero, indicating that the EnumCS value alone describes the colourspace or default values are used as defined by the referenced colourspace definition. The format and values of the EP field are defined in clause M.11.7.4. However, the EP field shall be the last field in the Colour Specification box and shall be all bytes in the box following the EnumCS field to the end of the box.

Table M.26 – Format of the contents of the METHDAT field for the Enumerated method

Field name	Size (bits)	Value
EnumCS	32	0-$(2^{32}-1)$
EP	Variable	Variable

M.11.7.3.2 METHDAT values for the Any ICC method

The contents of the METHDAT field for Colour Specification boxes using the Any ICC method is defined as follows (see Figure M.18 and Table M.27):

Figure M.18 – Organization of the contents of the METHDAT field for the Any ICC method

Profile: ICC Profile. This field contains an ICC input profile as defined by ICC-1, specifying the transformation between the decompressed code values and the PCS. Any input ICC profile, regardless of profile class, may be contained within this field.

Table M.27 – Format of the contents of the METHDAT field for the Any ICC method

Field name	Size (bits)	Value
PROFILE	Variable	Variable

M.11.7.3.3 METHDAT values for the Vendor Colour method

The contents of the METHDAT field for Colour Specification boxes using the Vendor Colour method is defined as follows (see Figure M.19):

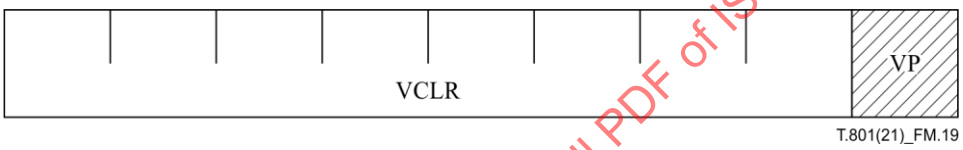

Figure M.19 – Organization of the contents of the METHDAT field for the Vendor Colour method

VCLR: Vendor Defined Code. This field specifies the colourspace of the image using a UUID. To correctly interpret the colour of an image using a Vendor defined colourspace, the application has to know the definition of that colourspace internally. This field contains a 16-byte UUID indicating the colourspace of the image. These values are defined and shared by individual vendors and are outside the scope of this Recommendation | International Standard.

VP: Vendor parameters. This field specifies a series of parameters that augment the generic colourspace definition specified by VCLR. Together, the VCLR and VP fields unambiguously describe the colourspace. The format and value of the VP field is defined individually for each VCLR value as required. If a value of VP is not defined for a particular value of VCLR, then the length of the VP field for that VCLR value shall be zero, indicating that the VCLR value alone unambiguously describes the colourspace, or default values are used as defined by the referenced colourspace definition. The format and values of the VP field are defined by each individual vendor colourspace definition, and are outside of the scope of this Recommendation | International Standard. However, the VP field shall be the last field in the Colour Specification box and shall be all bytes in the box following the VCLR field to the end of the box.

Table M.28 – Format of the contents of the METHDAT field for the Vendor Colour method

Field name	Size (bits)	Value
VCLR	128	Variable
VP	Variable	Variable

M.11.7.3.4 METHDAT values for the Parameterized method

The contents of the METHDAT field for Colour Specification boxes using the Parameterized method is defined as follows (see Figure M.19bis and Table M.28bis):

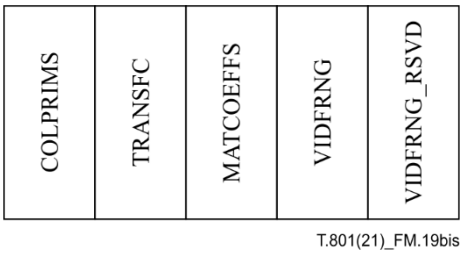

Figure M.19bis – Organization of the METHDAT field for the Parameterized method

Table M.28bis – Format of the METHDAT field for the Parameterized method

Field name	Size (bits)	Value
COLPRIMS	16	One of the ColourPrimaries enumerated values specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. This field is encoded as a big-endian unsigned integer.
TRANSFC	16	One of the TransferCharacteristics enumerated values specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. This field is encoded as a big-endian unsigned integer.
MATCOEFFS	16	One of the MatrixCoefficients enumerated values specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. This field is encoded as a big-endian unsigned integer.
VIDFRNG	1	Value of the VideoFullRangeFlag specified in Rec. ITU-T H.273 | ISO/IEC 23091-2. This field is the most-significant bit of the last byte of METHDAT.
VIDFRNG _RSVD	7	Reserved for future use by ITU-T | ISO/IEC. This field are the 7 least-significant bits of the last byte of METHDAT.

M.11.7.4 EP field format and values

This field defines the format and values of the EP fields for Colour Specification boxes using the Enumerated method. If an EP field is not defined for a particular value of the EnumCS field, then the length of the EP field shall be zero.

M.11.7.4.1 EP field format for the CIELab colourspace

If the value of EnumCS is 14, specifying that the layer is encoded in the CIELab colourspace, then the format of the EP field shall be as follows (see Figure M.20):

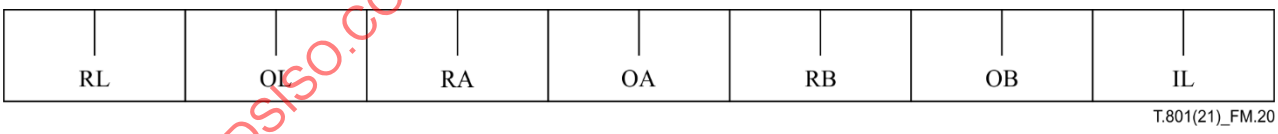

Figure M.20 – Organization of the contents of the EP field for the CIELab (EnumCS = 14)

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 Rec. ITU-T 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. 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:

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

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 clause M.11.7.2.

The IL field specifies the illuminant data used in calculating the CIELab values.

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 Rec. ITU-T 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:

Table M.29 – Standard illuminant values for CIELab

Illuminant	Standard IL field value
CIE Illuminant D50	0x0044 3530
CIE Illuminant D65	0x0044 3635
CIE Illuminant D75	0x0044 3735
CIE Illuminant SA	0x0000 5341
CIE Illuminant SC	0x0000 5343
CIE Illuminant F2	0x0000 4632
CIE Illuminant F7	0x0000 4637
CIE Illuminant F11	0x0046 3131

When the illuminant is specified by a colour temperature, then the 4 bytes consist of the string 'CT', followed by two unsigned bytes representing the temperature of the illuminant in degrees Kelvin as a 2-byte big endian unsigned integer. For example, a 7500K illuminant is represented by the 4 bytes 0x4354 1D4C.

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.29bis – Default Offset Values and Encoding of Offsets for the CIELab Colourspace

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

Table M.29bis – Default Offset Values and Encoding of Offsets for the CIELab Colourspace

unsigned, fixed point ($BPC^i < 128$)	0	2^{BPCi}	$2^{BPCi-1}+2^{BPCi-2}$
signed, fixed point ($BPC^i \geq 128$)	0	0	0
unsigned, floating point ($BPC^i < 128$)	0	2^{BPCi}	$2^{BPCi-1}+2^{BPCi-2}$
signed, floating point ($BPC^i \geq 128$)	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 the definition of the $Ssiz^i$ parameter in Rec. ITU-T T.800 | ISO/IEC 15444-1.			

NOTE – Other applications can 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.

Table M.30 – Format of the contents of the EP field for CIELab (EnumCS = 14)

Field name	Size (bits)	Value
RL	32	$0-(2^{32}-1)$
OL	32	$0-(2^{32}-1)$
RA	32	$0-(2^{32}-1)$
OA	32	$0-(2^{32}-1)$
RB	32	$0-(2^{32}-1)$
OB	32	$0-(2^{32}-1)$
IL	32	Variable

M.11.7.4.2 EP field format for the CIEJab colourspace

If the value of EnumCS is 19, specifying that the layer is encoded in the CIEJab colourspace, then the format of the EP field shall be as follows (see Figure M.21):

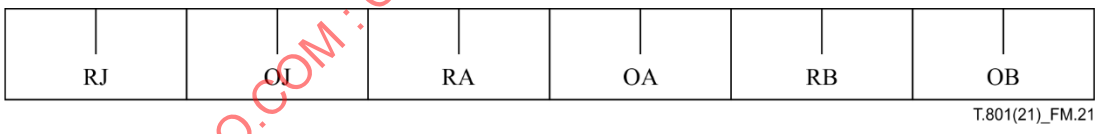

Figure M.21 – Organization of the contents of the EP field for the CIEJab (EnumCS = 19)

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:

$$
\begin{aligned}
N_J &= \frac{MAX_J}{RJ} \times J + \frac{2^{\lceil \log_2 MAX_J \rceil}}{2^{(BPC_J \ AND \ 0x7F)+1}} \times OJ \\
N_a &= \frac{MAX_a}{RA} \times a + \frac{2^{\lceil \log_2 MAX_a \rceil}}{2^{(BPC_a \ AND \ 0x7F)+1}} \times OA \\
N_b &= \frac{MAX_b}{RB} \times b + \frac{2^{\lceil \log_2 MAX_b \rceil}}{2^{(BPC_b \ AND \ 0x7F)+1}} \times OB
\end{aligned}
\qquad \text{(M-19)}
$$

where MAX_J, MAX_a and MAX_b are the nominal maximum sample values according to Table M.24bis 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 clause M.11.7.2.

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.30bis. 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.30bis – 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 the definition of the $Ssiz^i$ parameter in Rec. ITU-T 800 | ISO/IEC 15444-1.

The format of the contents of the EP field for CIEJab (EnumCS = 19) is given in Table M.31.

Table M.31 – Format of the contents of the EP field for CIEJab (EnumCS = 19)

Field name	Size (bits)	Value
RJ	32	$0-(2^{32}-1)$
OJ	32	$0-(2^{32}-1)$
RA	32	$0-(2^{32}-1)$
OA	32	$0-(2^{32}-1)$
RB	32	$0-(2^{32}-1)$
OB	32	$0-(2^{32}-1)$

M.11.7.5 Channel Definition box

The binary format of the Channel Definition box is identical to that defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, clause I.5.3.6. However, in a JPX file that is not readable by a JP2 reader, or in a codestream in a JPX file that will not be

read by a JP2 reader, any channel may be associated with any colour or type and multiple channels may be associated with the same colour.

EXAMPLE – Multiple channels of the same colour in a Bayer pattern can be described using the same Typ^i and $Asoc^i$ value pair, but different component registration position, as carried in the optional CRG marker segment.

The following additional value of the $Asoc^i$ field are normatively defined (see Table M.32):

Table M.32 – Colours indicated by the $Asoc^i$ field

Class of colourspace	Colour indicated by the following value of the $Asoc^i$ field			
	1	**2**	**3**	**4**
RGB	R	G	B	
Greyscale	Y			
XYZ	X	Y	Z	
Lab	L	a	b	
Luv	L	u	v	
YC_bC_r	Y	C_b	C_r	
Yxy	Y	x	y	
HSV	H	S	V	
HLS	H	L	S	
CMYK	C	M	Y	K
CMY	C	M	Y	
Jab	J	a	b	
n colour colourspaces	1	2	3	4

M.11.7.6 Opacity box

The Opacity box provides a minimal-overhead mechanism for specifying opacity through a chroma-key or specifying that a particular compositing layer contains only colour channels followed by a single opacity channel. If a Compositing Layer Header box contains an Opacity box, then it shall not contain a Channel Definition box. Compositing layers that require a channel definition more complex than can be defined using an Opacity box shall use a Channel Definition box. Each Compositing Layer Header box shall contain zero or one Opacity boxes, and Opacity boxes shall be found in no other locations in the file.

Chroma-keyed opacity is a form of paletization and as such images using chroma-keyed opacity obey similar rules to full palettized images with respect to lossy compression. In either case, differences between the original image and the decompressed images reflect errors in a space that does not directly map to visual perception, and thus should not be coded or decompressed in a lossy mode. However, for chroma-key values, in contrast to a fully palettized component, only the samples of the image that are of the chroma-key value have to be encoded and decoded losslessly. Joint lossless encoding of the chroma-keyed region and lossy coding of the remaining image region can be achieved using a ROI within the codestream.

NOTE – The Opacity box is intended for use with lossless coding.

The type of the Opacity box shall be 'opct' (0x6F70 6374). The contents of this box shall be as follows (see Figure M.22):

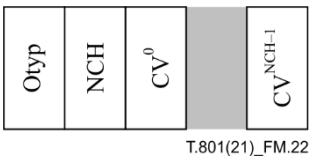

Figure M.22 – Organization of the contents of an Opacity box

Otyp: Opacity type. This field specifies the type of opacity used by this compositing layer. This field is encoded as a 1-byte unsigned integer. Legal values of the Otyp field are as follows:

Table M.33 – Otyp field values

Value	Meaning
0	The last channel in this compositing layer is an opacity channel and all other channels are colour channels where the channel association is equal to the channel number +1. For example, a four-channel compositing layer would contain 3 colour channels (with associations 1, 2 and 3 respectively) followed by an opacity channel. If the value of Otyp is 0, then the NCH, PR and CV^i fields shall not be found.
1	The last channel in this compositing layer is a premultiplied opacity channel and all other channels are colour channels where the channel association is equal to the channel number +1. For example, a four-channel compositing layer would contain 3 colour channels (with associations 1, 2 and 3 respectively) followed by a premultiplied opacity channel. If the value of Otyp is 0, then the NCH, PR and CV^i fields shall not be found.
2	This compositing layer specifies that samples of a particular colour shall be considered fully transparent (chroma-key). The chroma-key colour is specified by the NCH, PR and CV^i fields.
	All other values reserved.

NCH: Number of channels. This field specifies the number of channels used to specify the chroma-key colour. This value shall be equal to the number of channels in the compositing layer. This field is specified as a 1-byte unsigned integer.

CV^i: Chroma-key value. This field specifies the value of channel *i* for the chroma-key colour. Samples that match the chroma-key value for all channels shall be considered fully transparent. The size of this field is specified by the bit depth of the corresponding channel. If the value is not a multiple of 8, then each CV^i value shall be padded to a multiple of 8 bits with bits equal to the sign bit and the actual value shall be stored in the low-order bits of the padded value. For example, if the depth of a channel is a signed 10-bit value, then the CV^i value shall be stored in the low 10 bits of a 16-bit field and the high-order 6 bits shall be all equal to the sign bit of the value in this CV^i field.

The format of the contents of the Opacity box is given in Table M.34.

Table M.34 – Format of the contents of the Opacity box

Field name	Size (bits)	Value
Otyp	8	0-2
NCH	8 0	0-255; if Otyp ≠ 2 Not applicable; if Otyp = 2
CV^i	Variable 0	Variable; if Otyp ≠ 2 Not applicable; if Otyp = 2

M.11.7.7 Codestream Registration box

When combining multiple codestreams to create a single compositing layer, it is important that the reference grids of those codestreams be properly registered to ensure the registration of the individual samples from the multiple components. This box specifies how those codestreams shall be registered when rendering the layer. A Compositing Layer Header box shall contain zero or one Codestream Registration boxes, and Codestream Registration boxes shall be found in no other locations in the file; a Codestream Registration box shall not be placed into the JP2 Header box to specify a default registration. If any Compositing Layer Header box contains a Codestream Registration box, then every Compositing Layer Header box shall contain a Codestream Registration box. If this Compositing Layer Header box does not contain a Codestream Registration box, then the compositing layer shall be represented by one and only one codestream.

If codestream registration is not specified for a particular compositing layer, then the codestreams in that compositing layer shall be aligned by directly aligning their reference grids at both (0,0) and (1,1).

If a Codestream Registration box exists, then the default display resolution (specified within a Resolution box with the same Compositing Layer Header box) applies to the compositing layer registration grid.

This registration is specified with respect to an independent compositing layer registration grid.

The type of the Codestream Registration box shall be 'creg' (0x6372 6567). The contents of this box shall be as follows (see Figure M.23):

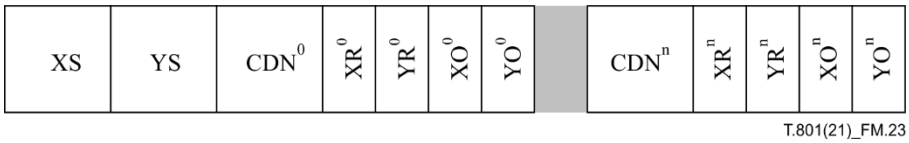

Figure M.23 – Organization of the contents of a Codestream Registration box

The fields of the codestream registration box may be interpreted in the context of the following equations defining the scaling factors RX^i and RY^i of the ith codestream in x and y:

$RX^i = XR^i / XS$

$RY^i = YR^i / YS$

and the offsets OX^i and OY^i of the ith codestream in x and y:

$OX^i = XO^i / XS$

$OY^i = YO^i / YS$

on the compositing layer registration grid. The overall area of the registration grid is defined to be the intersection of the areas covered by the scaled and offset component codestreams grown out as required to the nearest whole integer registration grid point. The requirement that $XO^i < XS$ and $YO^i < YS$ ensures that this intersected area always includes the origin (0,0) of the compositing layer registration grid.

If a Codestream Registration box exists, then the default display resolution (specified within a Resolution box with the same Compositing Layer Header box) applies to the compositing layer after all components have been rescaled and registered.

XS: Horizontal grid size. This field defines the number of horizontal fractional grid points from (0, 0) to (1, 0) on the compositing layer registration grid. These fractional grid points are used to measure the distance between the reference grids of the individual codestreams. This field is encoded as a 2-byte unsigned integer.

YS: Vertical grid size. This field defines the number of vertical fractional grid points from (0, 0) to (0, 1) on the compositing layer registration grid. These fractional grid points are used to measure the distance between the reference grids of the individual codestreams. This field is encoded as a 2-byte unsigned integer.

CDN^i: Codestream number. This field specifies the number of the codestream for this registration value.

XR^i: Horizontal resolution. This field specifies the horizontal distance between points on the reference grid of the codestream specified by the CDN^i parameter, measured in the number of fractional points on the compositing layer registration grid. This field effectively specifies the horizontal scaling needed to match the codestream's reference grid with the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

YR^i: Vertical resolution. This field specifies the vertical distance between points on the reference grid of the codestream specified by the CDN^i parameter, measured in the number of fractional points on the compositing layer registration grid. This field effectively specifies the vertical scaling needed to match the codestream's reference grid with the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

XO^i: Horizontal offset. This field specifies the horizontal distance (to the right) from the origin of the compositing layer registration grid to the centre of the top left point on the reference grid of the codestream specified by the CDN^i parameter. This field is encoded as a 1-byte unsigned integer. Its value shall be strictly less than the value of XS.

YO^i: Vertical offset. This field specifies the vertical distance (downward) from the origin of the compositing layer registration grid to the centre of the top left point on the reference grid of the codestream specified by the CDN^i parameter. This field is encoded as a 1-byte unsigned integer. Its value shall be strictly less than the value of YS.

Table M.35 – Format of the contents of the Codestream Registration box

Field name	Size (bits)	Value
XS	16	0-65 535
YS	16	0-65 535
CDN^i	16	0-65 535
XR^i	8	0-255
YR^i	8	0-255
XO^i	8	0-255
YO^i	8	0-255

M.11.7.8 Pixel Format box

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 Rec. ITU-T 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 Rec. ITU-T 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 1 – 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 (see Figure M.23bis):

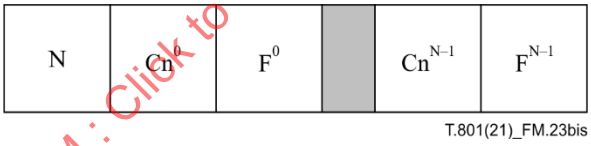

Figure M.23bis – Layout of the Pixel Format Box

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.35bis lists allowed values for this field and its encoding.

Table M.35bis – Allowed values for the pixel format

Value of 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 $V = \text{mantissa} \cdot 2^{\text{exponent}-136}$ The association of the combined sample value is defined by the 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 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 ISO/IEC/IEEE 60559.
all other values	Reserved for future use.

NOTE 2 – 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 ISO/IEC/IEEE 60559 encodings as much as possible, even though ISO/IEC/IEEE 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 Table M.35ter 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.35ter – Common floating point formats (informative)

Value	Meaning
0100 0000 0001 0111	ISO/IEC/IEEE 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 shall have a bit precision of 32 bits.
0100 0000 0011 0100	ISO/IEC/IEEE 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	ISO/IEC/IEEE 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 3 – If for example the intent is to encode non-negative ISO/IEC/IEEE 60559 single precision numbers with the least significant 8 bits of the mantissa stripped off, the BPC^i value of a component using this encoding would be 23 as 24 bits in total are used. The corresponding F^i would then be `0100 0000 0000 1111` as 15 mantissa bits remain in the truncated format.

M.11.8 Contiguous Codestream box

In a JPX file, the Contiguous Codestream box contains an entire codestream as defined by the codestream syntax. However, unlike the JP2 file format, the codestreams contained within a JPX file are not restricted to codestreams defined by Annex A of Rec. ITU-T T.800 | ISO/IEC 15444-1. Codestreams contained within a JPX file may also use extensions to the codestream syntax defined in Annex A of this Recommendation | International Standard.

Contiguous Codestream boxes shall be found only at the top level of the file; they shall not be found within a superbox.

M.11.9 Media Data box

The Media Data box contains fragments of the JPEG 2000 codestream or other media data, such as MPEG-4 audio data. In any case, there shall be other boxes in the file that specify the meaning of the data within the Media Data box. Applications should not access Media Data boxes directly, but instead use the fragment table to determine what parts of which Media Data boxes represent a valid JPEG 2000 codestream or other media stream.

The type of a Media Data box shall be 'mdat' (0x6D64 6174). The contents of a Media Data box in general are not defined by this Recommendation | International Standard.

M.11.10 Composition box (superbox)

The Composition box specifies how the individual composition layers are combined to create the rendered result. It contains a set of global options, followed by a sequence of one or more sets of rendering instructions (each contained within an Instruction Set box). Each individual instruction is associated with a composition layer in the file and defines how that composition layer shall be rendered: its location, scaling, composite operation, etc. A reader that supports composition and animation shall display the file containing the Composition box by executing the sequence of instructions defined within the Composition box. Details on the composition and animation model are specified in clause M.5.3. A JPX file shall contain zero or one Composition boxes. If present, that box shall be found at the top level of the JPX file; it shall not be found within a superbox.

The type of the Composition box shall be 'comp' (0x636F 6D70) and it shall have the following contents (see Figure M.24):

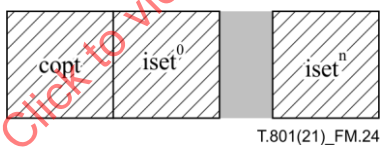

Figure M.24 – Organization of the contents of a Composition box

copt: Composition Options box. This box specifies parameters that apply to the composition or animation as a whole. It is defined in clause M.11.10.1.

iset^i: Instruction Set box. This box contains a set of instructions for how to combine the multiple composition layers in the file. The entire set of Instruction Set boxes specify the entire composition or animation, and are processed in the order they are found within the Composition box. The Composition Instruction box is defined in clause M.11.10.2.

The format of the contents of the Composition box is given in Table M.36.

Table M.36 – Format of the contents of the Composition box

Parameter	Size (bits)	Value
copt	Variable	Variable
$iset^i$	Variable	Variable

M.11.10.1 Composition Options box

The Composition Options box specifies parameters that apply to the composition or animation as a whole. The Composition Options box shall be the first box in the Composition box and a Composition Options box shall not be found in any other location in the file.

The type of the Composition Options box shall be 'copt' (0x636F 7074) and contents of the box shall have the following format (see Figure M.25):

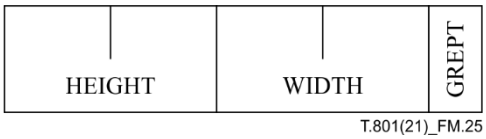

Figure M.25 – Organization of the contents of a Composition Options box

HEIGHT: Rendered result height. This field specifies the height, in samples, of the final rendered result. The resolution of this value is optionally defined in the Default Display Resolution box in the JP2 Header box. This field is encoded as a 4-byte unsigned integer.

WIDTH: Rendered result width. This field specifies the width, in samples, of the final rendered result. The resolution of this value is optionally defined in the Default Display Resolution box in the JP2 Header box. This field is encoded as a 4-byte unsigned integer.

GREPT: Global Repetition. This field specifies the number of times to fully repeat the display instructions, after executing the display instructions the first time. For a GREPT value of zero this means the entire set of instructions in the comp box is executed once. A value of 255 indicates that the reader should repeat the entire set of instructions indefinitely. Prior to each execution of the instruction set, the display area shall be restored to its original state and all instructions' composition layer association reset. Each loop execution should be visually equivalent to redisplaying the composition from scratch. This field is encoded as a 1-byte unsigned integer.

The format of the contents of the Composition Options box is given in Table M.37.

Table M.37 – Format of the contents of the Composition Options box

Parameter	Size (bits)	Value
HEIGHT	32	$1\text{-}2^{32}–1$
WIDTH	32	$1\text{-}2^{32}–1$
GREPT	8	0-255

M.11.10.2 Instruction Set box

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.

The type of the Instruction Set box shall be 'inst' (0x696E 7374) and contents of the box shall have the following format (see Figure M.26):

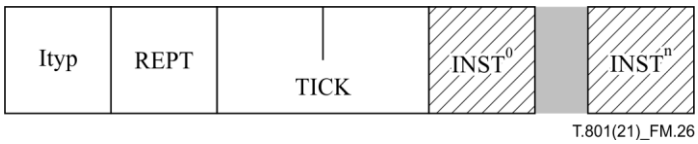

Figure M.26 – Organization of the contents of an Instruction Set box

Ityp: Instruction type. This field specifies the type of this instruction, and thus which instruction parameters shall be found within this Composition Instruction box. This field is encoded as a 16-bit flag. The meaning of each bit in the flag is as follows:

Table M.38 – 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	All other values reserved.

REPT: Repetition. This field specifies the number of times to repeat this particular set of instructions after executing the instruction set the first time. The instructions are always executed at least once (REPT is zero) and the instructions may apply to different compositing layers on each repetition as determined by the Next-use field of the instructions. This field is encoded as a 2-byte big endian unsigned integer. A value of 65 535 indicates to repeat the instruction indefinitely.

TICK: Duration of timer tick. This field specifies the duration of a timer tick (used by the LIFE instruction parameter) in milliseconds. This field is encoded as a 4-byte big endian unsigned integer. If the Ityp field specifies that the LIFE instruction parameter is not used, then this field shall be set to 0, and shall be ignored by readers.

INST^{i}: Instruction. This field specifies a series of instruction parameters for a single instruction. The format of this field is specified in clause M.11.10.2.1.

The format of the contents of the Instruction Set box is given in Table M.39.

Table M.39 – Format of the contents of the Instruction Set box

Parameter	Size (bits)	Value
Ityp	16	0-65 535
REPT	16	0-65 535
TICK	32	0-(2^{32}–1)
INST^{i}	Variable	Variable

M.11.10.2.1 Instruction parameter

Figure M.27 shows the contents of each individual INST field (a single compositing instruction) within an Instruction Set box:

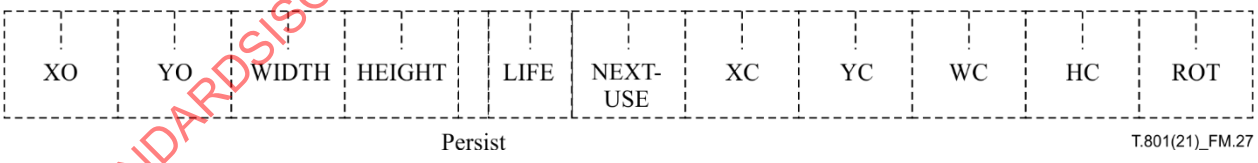

Figure M.27 – Organization of the contents of an INST field within an Instruction Set box

XO: Horizontal offset. This field specifies the horizontal location at which the top left corner of the compositing layer being acted on by this instruction shall be placed in the render area, in samples. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, a default value of zero shall be used.

YO: Vertical offset. This field specifies the vertical location at which the top left corner of the compositing layer being acted on by this instruction shall be placed in the render area, in samples. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, a default value of zero shall be used.

WIDTH: Width of the current compositing layer. This field specifies the width on the render area, in display samples, into which to scale and render the compositing layer being acted on by this instruction. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, the width of the compositing layer shall be used.

HEIGHT: Height of the current compositing layer. This field specifies the height on the render area, in display samples, into which to scale and render the compositing layer being acted on by this instruction. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, the height of the compositing layer shall be used.

PERSIST: Persistence. This field specifies whether the samples rendered to the display as a result of the execution of the current instruction shall persist on the display background or if the display background shall be reset to the state before the execution of this instruction, before the execution of the next instruction. This field is encoded as a 1-bit Boolean field. A value of 1 indicates true, that the current compositing layer shall persist. If this field is not present, the persistence shall be set to true.

LIFE: Duration of this instruction. This field specifies the number of timer ticks that should ideally occur between completing the execution of the current instruction and completing execution of the next instruction. A value of zero indicates that the current instruction and the next instruction shall be executed within the same display update; this allows a single frame from the animation to be composed of updates to multiple compositing layers. A value of $2^{31}-1$ indicates an indefinite delay or pause for user interaction. This field is encoded as a 31-bit big endian unsigned integer. If this field is not present, the life of the instruction shall be set to 0.

NEXT-USE: Number of instructions before reuse. This field specifies the number of instructions that shall be executed before reusing the current compositing layer. This field allows readers to simply optimize their caching strategy. A value of zero implies that the current image shall not be reused for any ensuing instructions, notwithstanding the execution of a global loop as a result of a non-zero value of the GREPT parameter in the Composition Options box. A value of one (1) implies that the current compositing layer will be used with the next instruction and so on. The compositing layer passed on for reuse in this manner shall be the original compositing layer, prior to any cropping or scaling indicated by the current instruction. If this field is not present, the number of instructions shall be set to zero, indicating that the current compositing layer shall not be reused. This field is encoded as a 4-byte big endian unsigned integer.

XC: Horizontal crop offset. This field specifies the horizontal distance in samples to the left edge of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the horizontal crop offset shall be set to 0. This field is encoded as a 4-byte big endian unsigned integer.

YC: Vertical crop offset. This field specifies the vertical distance in samples to the top edge of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the vertical crop offset shall be set to 0. This field is encoded as a 4-byte big endian unsigned integer.

WC: Cropped width. This field specifies the horizontal size in samples of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the cropped width shall be set to the width of the current compositing layer. This field is encoded as a 4-byte big endian unsigned integer.

HC: Cropped height. This field specifies the vertical size in samples of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the cropped height shall be set to the height of the current compositing layer. This field is encoded as a 4-byte big endian unsigned integer.

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.40bis.

References to Ityp within the individual instruction parameters in Table M.40 refers to the Ityp field within the Instruction Set box that contains this Instruction.

Table M.40 – Format of the contents of the INST^{i} parameter in the Instruction Set box

Parameter	Size (bits)	Value
XO	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xxx1 Not applicable otherwise
YO	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xxx1 Not applicable otherwise
WIDTH	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xx1x Not applicable otherwise
HEIGHT	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xx1x Not applicable otherwise
PERSIST	1 0	0, 1; if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
LIFE	31 0	0-(2^{31}–1); if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
NEXT-USE	32	0-(2^{31}–1); if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
XC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
YC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
WC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
HC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
ROT	32 0	0-31, see Table M.47; if ITyp contains xxxx xxxx x1xx xxxx. Not applicable otherwise

Table M.40bis – 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

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.

M.11.11 Association box (superbox)

The Association box allows data in the file to be associated with other data in the file. The Association box is a superbox, containing a sequence of two or more boxes. It creates independent semantic associations between the boxes it contains or the entities represented by those boxes. In particular, associations are created between the first box (or entities represented by it) (referred to as BF) and each of the other boxes (or represented entities) (referred to as B^{i}) in the sequence. In the case where there are more than one B^{i} boxes, it can be thought of as creating semantic clusters around the BF box. There is no explicit association between the B^{i} boxes.

For example, the association box may be used to associate a label with an entity (image, image set, metadata document, etc.) by placing a Label box in the Association box as BF and the other appropriate boxes as the B^i boxes. It may also be used to associate several items of metadata with the same image or image set by placing a Number List box as BF, followed by the metadata boxes as the B^i boxes. In addition, it may be used recursively to create different levels of association, for example to associate some metadata with a Region of Interest (ROI) and then to associate that ROI and its metadata with an image or image set. These examples are illustrated in Figures M.28 to M.31.

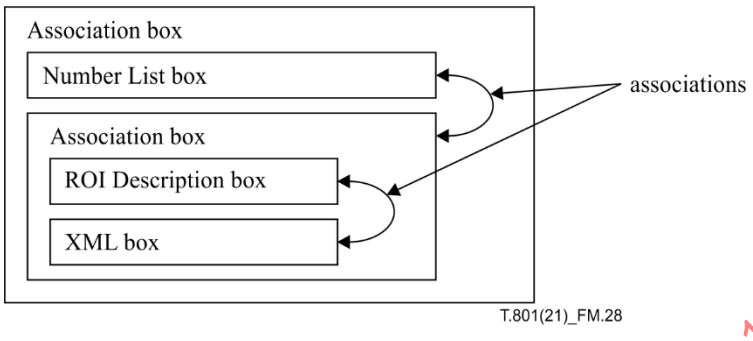

Figure M.28 – Example of ROI specific metadata associated with one or more images

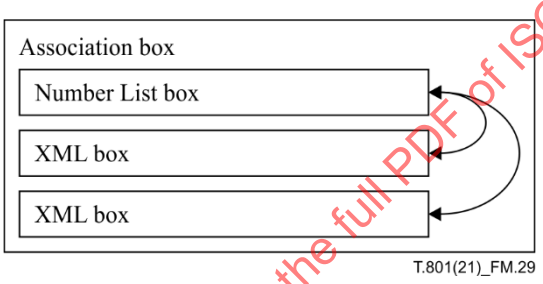

Figure M.29 – Example of Multiple XML documents associated with one or more images

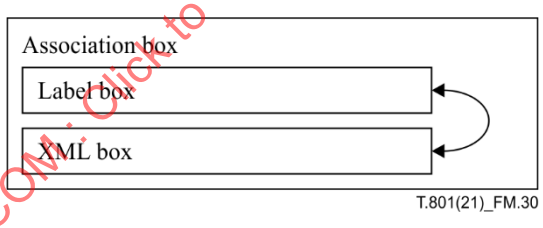

Figure M.30 – Example of a Labelled XML document

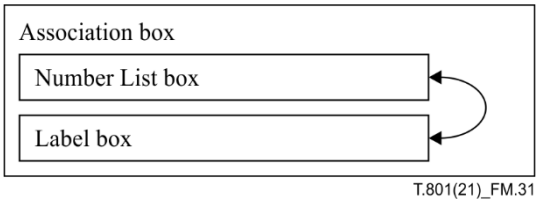

Figure M.31 – Example of a labelled image

The Association box is optional, and there may be multiple Association boxes in the file. An Association box may be found anywhere in the file except before the Reader Requirements box.

The type of an Association box shall be 'asoc' (0x6173 6F63). The contents of the Association box are defined as follows (see Figure M.32):

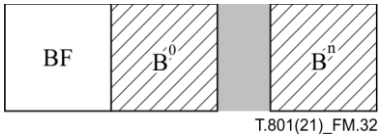

Figure M.32 – Organization of the contents of an Association box

BF: First box. This is the box to which all other boxes within this Association box are associated.

B^i: Box to be associated. This may be any box other than those that are restricted to occurring at particular locations within the file. This box shall be associated with the box BF.

The format of the contents of the Association box is given in Table M.41.

Table M.41 – Format of the contents of the Association box

Parameter	Size (bits)	Value
BF	Variable	Variable
B^i	Variable	Variable

M.11.12 Number List box

The Number List box contains a list of numbers designating entities in the file. Within an Association box, a Number List box stands for the listed entities.

The type of a Number List box shall be 'nlst' (0x6E6C 7374). The contents of the Number List Box shall be as follows (see Figure M.33):

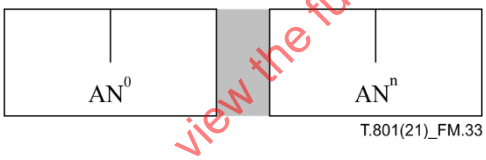

Figure M.33 – Organization of the contents of a Number List box

AN^i: Associate Number. This field specifies the number of an entity with which the data contained within the same Association box is associated. This value is stored as a 4-byte big endian unsigned integer, where the high order byte specifies the type of entity with which the data is associated, and the three low order bytes specify the number of that entity. Legal values of this field are given in Table M.42.

Table M.42 – AN^i field values

Value	Meaning
0x0000 0000	The rendered result.
0x01XX XXXX	The low three order bytes (of value *i*) specify Codestream *i* in the JPX file.
0x02XX XXXX	The lower three order bytes (of value *i*) specify Compositing Layer *i* in the JPX file.
	All other values reserved.

The format of the contents of the Number List box is given in Table M.43.

Table M.43 – Format of the contents of the Number List box

Parameter	Size (bits)	Value
AN^i	32	$0-(2^{32}-1)$

M.11.13 Label box

The Label box contains a textual label that may be associated with an entity or entities in the file by inclusion of the Label box within an Association box, a Codestream Header box, or a Compositing Layer Header box.

The type of a Label box shall be 'lbl\040' (0x6C62 6C20). The contents of the Label box are as follows (see Figure M.34):

Figure M.34 – Organization of the contents of a Label box

S: Label string. A textual label associated with an entity. This value is stored as ISO/IEC 10646 characters in the UTF-8 encoding. Characters in the ranges U+0000 to U+001F inclusive and U+007F to U+009F inclusive, as well as the specific characters '/', ';', '?', ':' and '#', are not permitted in the label string. Label strings are not null-terminated or padded in any other way; every character that is present is significant.

M.11.14 Binary Filter box

The Binary Filter box allows portions of the file to be further compressed or encoded (i.e., encrypted). For example, if the file contains a significant amount of metadata in XML, it can be losslessly compressed to drastically reduce the file size. This box contains an indicator specifying how the data was transformed, as well as the transformed data. Once the data is transformed through the reverse operation (i.e., decrypted or decompressed), the resulting data shall be a sequence of boxes, where the first byte is the first byte of the first box header, and the last byte is the last byte of the last box. The Binary Filter box is optional, and there may be multiple Binary Filter boxes in the file. A Binary Filter box may be found anywhere in the file except before the Reader Requirements box.

A conforming decoder is not required to process the data within a Binary Filter box. Thus, a Binary Filter box shall not contain boxes for which interpretation is required for reader conformance.

The type of a Binary Filter box shall be 'bfil' (0x6266 696C). The contents of the Binary Filter box are defined as follows (see Figure M.35):

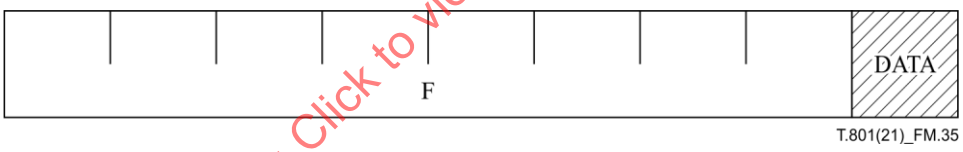

Figure M.35 – Organization of the contents of a Binary Filter box

F: Filter type. This field specifies how the data was transformed before storage. This value is encoded as a UUID. Standard defined values are given in Table M.44.

Table M.44 – Legal Filter types

Value	Meaning
EC340B04-74C5-11D4-A729-879EA3548F0E	**Compressed with GZIP.** The contents of the DATA field have been compressed using the DEFLATE algorithm (as specified in RFC 1951). The compressed data is stored in the binary structure defined by the GZIP file format, as specified in RFC 1952.
EC340B04-74C5-11D4-A729-879EA3548F0F	**Encrypted using DES.** The contents of the DATA field has been encrypted using DES as defined in ISO 10126-2. NOTE: ISO 10126-2 is withdrawn.
	All other values reserved.

If a conforming reader does not recognize the particular UUID, then the reader shall ignore this Binary Filter box.

DATA: Transformed data. This field contains previously transformed data. Once the reverse transformation has been applied (as specified by F), the result shall be a sequence of boxes. The contents of the data field may include information needed to perform the reverse filter, in addition to the filtered data. It is fully up to the definition of the F field to define the binary structure and format of the DATA field.

The format of the contents of the Binary Filter box is given in Table M.45.

Table M.45 – Format of the contents of the Binary Filter box

Parameter	Size (bits)	Value
F	128	Variable
DATA	Variable	Variable

M.11.15 Desired Reproductions box (superbox)

The Desired Reproductions box specifies a set of transformations that shall be applied to the image to guarantee a specific desired reproduction on a set of different output devices, respectively. For example, consider an image that contains real-world blue colours. This image is intended to be printed in a catalogue, and thus the printed image shall match the actual colour of the original physical object when seen by a human viewer. However, the CMYK printing process does not reproduce the same range of blue colours as are viewable by the human visual system. In this instance, the catalog artist shall determine how to best convert the blue colour in the image to a printed blue colour to minimize differences between the physical object from the printed reproduction.

A JPX reader is not required to process the image through the specified transformations.

This box contains a set of separate desired reproductions. There shall be only one Desired Reproductions box within the file, which may be found anywhere within the file.

The type of the Desired Reproductions box is 'drep' (0x6472 6570). This box is a superbox, and the contents of the box shall be as follows (see Figure M.36):

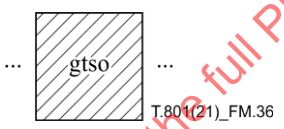

Figure M.36 – Organization of the contents of the Desired Reproductions box

gtso: Graphics Technology Standard Output box. This box specifies the desired output colour and tone reproduction for the rendered result when printed under commercial printing conditions. The format and definition of this box is specified in clause M.11.15.1

Other boxes may be found within the Desired Reproductions box. Readers shall ignore any boxes that they do not understand.

The Desired Reproduction box is optional for conforming files.

M.11.15.1 Graphics Technology Standard Output box

A Graphics Technology Standard Output box specifies the desired reproduction of the rendered result for commercial printing and proofing systems. The box contains an Output ICC profile specifying the desired conversion of the image from the Profile Connection Space (PCS) to the desired device specific output colourspace. There shall be only zero or one Graphics Technology Standard Output box within the file. If present, this box shall be found within the Desired Reproductions box.

The type of a Graphics Technology Standard Output box is 'gtso' (0x6774 736F). The contents of the box shall be as follows (see Figure M.37):

Figure M.37 – Organization of the contents of the Graphics Technology Standard Output box

OUTP: This field shall be a valid Output ICC profile as defined by the ICC Profile format specification ICC-1. Version information is embedded within the profile itself. Applications that only support specific versions of the ICC Profile Format Specifications can extract the version number from bytes 8-11 of the profile (bytes 8–11 of the contents of the Output ICC Profile box).

The format of the contents of the Graphics Technology Standard Output box is given in Table M.46.

Table M.46 – Format of the contents of the Graphics Technology Standard Output box

Parameter	Size (bits)	Value
OUTP	Variable	Variable

M.11.16 ROI Description box

An ROI Description box contains information about parts of an image that might be useful in certain applications such as random access. The ROI description box can also be used together with the association box to associate metadata to parts of the image. The ROIs described in this box are not necessarily coded as ROIs within the codestream; it also allows an application or user to signal the importance of certain parts of an image even if these parts are not emphasized by the RGN or ARN marker segment in the codestream. There can be multiple ROI Description boxes within the file. However, a ROI Description box shall be found only at the top level of the file, or within the JP2 Header box, a Codestream Header box or an Association box. If the ROI Description box is found within a Codestream Header box, then the ROIs described in that ROI Description box pertain to the particular codestream described by that Codestream Header box. If the ROI Description box is found within the JP2 Header box, then the ROI description box specifies default ROI information for all codestreams. If the ROI Description box is found at the top level of the file, then it specifies ROI information for the rendered result; the ROIs described at a top-level box are not directly associated with coded ROIs within any codestream.

The type of the ROI Description box shall be 'roid' (0x726F 6964). The contents of this box shall be as follows (see Figure M.38):

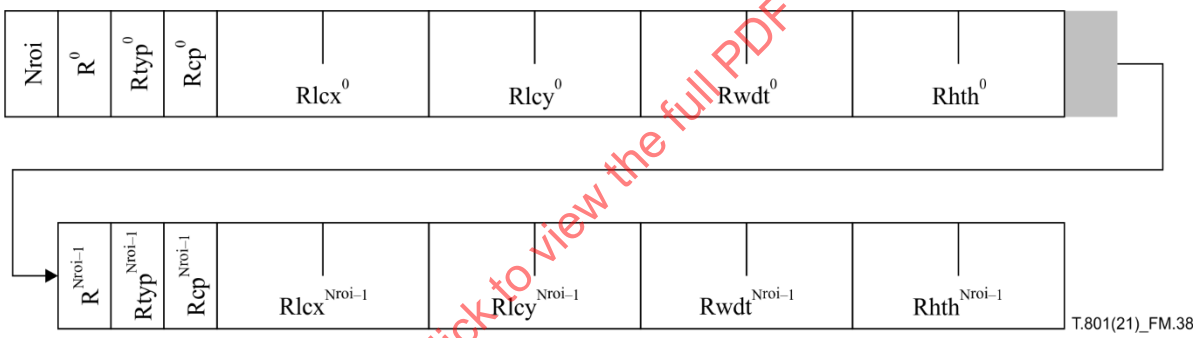

Figure M.38 – Organization of the contents of the ROI Description box

Nroi: Number of Regions of Interest. Encoded as an 8-bit integer.

R^i: Region of Interest present in codestream. Encoded as an 8-bit integer. Legal values of the R^i field are given in Table M.47.

Table M.47 – Legal R^i values

Value	Meaning
0	Codestream does not contain a static region of interest at this location.
1	Codestream contains a static region of interest at this location.
	All other values reserved.

$Rtyp^i$: 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:

Table M.48 – 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.

Rcp^i: Region of Interest coding priority. This value describes the coding priority of the Region of Interest. The value 0 means low coding priority and 255 means maximum coding priority. This value is encoded as a 1-byte unsigned integer. In transcoding applications, bits should be allocated with respect to the coding priority of each ROI.

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 immediately preceding region of interest. This value is stored as a 4-byte big endian unsigned integer.

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 immediately preceding region of interest. This value is stored as a 4-byte big endian unsigned integer.

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.

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.

The format of the contents of the ROI Description box is given in Table M.49.

Table M.49 – 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 ≠ 3 1 if Ryp^i = 3
Rhth^i	32	1 – (2^{32}–1) if Rtyp^i ≠ 3 1 if Ryp^i = 3

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:

$$Rlcx^i \le x < Rlcx^i + Rwdt^i \text{ and } Rlcy^i \le y < Rlcy^i + Rhth^i.$$

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.49, 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:

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

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

These values shall satisfy the following constraints:

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

The quadrilateral 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:

$$V1y = Y[0];\ V2y = Y[1];\ V3y = Y[2];\ \text{and } V4y = Y[3].$$

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.49, 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.49, 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.49; 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.49bis identifies the three possible connectivity cases and the associated values of D.

Table M.49bis – 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 1 – 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:

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

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

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 $\alpha \cdot (Y_C\text{-}y)$. Location (x,y) belongs to the oriented ellipse if and only if it satisfies: $(x - X_C - \alpha(Y_C\text{-}y))^2 / W_I^2 + ((y - Y_C)^2 / H_O^2 \leq 1$.

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 $\beta \cdot (x\text{-}X_C)$. Location (x,y) belongs to the oriented ellipse if and only if it satisfies: $(y - Y_C - \beta\,(x - X_C))^2 / H_I^2 + ((x - X_C)^2 / W_O^2 \leq 1$.

The values of $Rlcx^i$ and $Rlcy^i$ 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 $Rlcx^i$ and $Rlcy^i$ satisfy the following compatibility constraint:

$$\gamma W_O - ½ \leq (Rlcx^i - X_C) \leq \gamma W_O + ½;\ \text{and}\ \gamma H_O - ½ \leq (Y_C - Rlcy^i) \leq \gamma H_O + ½.$$

NOTE 2 – The compatibility constraint serves to ensure compatibility between the horizontal and vertical skewing procedures.

M.11.17 Digital Signature box

This box contains a checksum or digital signature that may be used to verify data contained within the file. This digital signature is used to protect a very specific byte stream within the file. Any change to that byte stream will invalidate the digital signature. For example, if a compressed codestream is signed, and then later modified by adding error resilience markers, the digital signature will indicate that the byte stream has been modified.

The type of a Digital Signature box shall be 'chck' (0x6368 636B). The Digital Signature box is optional and may occur anywhere in the file except before the Reader Requirements box. There may be more than one Digital Signature box in the file. The contents of a Digital Signature box shall be as follows (see Figure M.39):

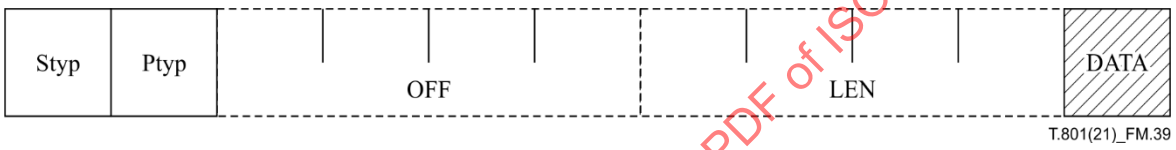

Figure M.39 – Organization of the contents of a Digital Signature box

Styp: Signature type. This field specifies the type of digital signature contained within this Digital Signature box. This field is encoded as a 1-byte unsigned integer. Legal values of the Styp field are as shown in Table M.50.

Table M.50 – Legal Styp values

Value	Meaning
0	A checksum, generated by applying the MD5 algorithm, as defined in IETF RFC 1321, to the source data, is stored in the DATA field of this Digital Signature box as a sequence of bytes in the order specified by IETF RFC 1321.
1	The checksum is generated by applying the SHA-1 algorithm, as defined in ANSI X9.30.2, to the source data. The resulting signature is stored in the DATA field of this Digital Signature box as a 20-byte big endian unsigned integer. NOTE – The SHA-1 algorithm is considered cryptographically weak and may not be suitable for some purposes; it may, however, be suitable for use with the Recommendation | International Standard under some circumstances.
2	The digital signature is generated by applying the DSA algorithm, as defined in FIPS 186-4, to the source data. The signature consists of two unsigned integers, r and s. The DATA field shall be 40 bytes long. The first 20 bytes shall be the value r, encoded as a 20-byte big endian unsigned integer. The second 20 bytes shall be the value s, encoded as a 20-byte big endian unsigned integer.
3	The digital signature stored in the DATA field of this Digital Signature box is generated by applying the RSA signature algorithm PKCS #1 Version 1.5, as defined in FIPS 186-4, with the MD5 message digest algorithm, as defined in IETF RFC 1321, to the source data.
4	The digital signature stored in the DATA field of this Digital Signature box is generated by applying the RSA signature algorithm PKCS #1 Version 1.5, as defined in FIPS 186-4, with the SHA-1 message digest algorithm, as defined in ANSI X9.30.2, to the source data. See note for the Styp value 1 above.
5	A ContentInfo value of the Cryptographic Message Syntax is stored in the DATA field of this Digital Signature box. Its 'content' field shall contain either a DigestedData value or a SignedData value, and in either case shall use the 'external signatures' mechanism described in section 5.2 of RFC 2630 to apply the chosen digest or signature algorithm to the source data.
	All other values reserved.

If key management is not an issue for a particular application (for example, if a checksum is being sent, or if the recipient already knows the public key with which to verify a signature), and if the CMS method (Styp = 5) is being used, it may help simple readers to include in the file an additional Digital

Signature box using one of the other methods (Styp < 5) on the same source data. Readers without CMS support would still be able to process the additional box.

Determination of any required public key is outside the scope of this Recommendation | International Standard.

Ptyp: Source pointer type. This field indicates how the source data range that is signed by this Digital Signature box is specified. This field is encoded as a 1-byte unsigned integer. Legal values of the Ptyp field are as given in Table M.51.

Table M.51 – Legal Ptyp values

Value	Meaning
0	The source data that is signed by this Digital Signature box shall be all bytes of the file, starting with the first byte, up to the byte immediately preceding the box header for this Digital Signature box. If the source data is specified using a Ptyp of 0, then this Digital Signature box shall not be in any superbox in the file; it shall be at the top level of the file.
1	The source data that is signed by this Digital Signature box shall be a range of bytes, starting with the byte at the location specified by the OFF field. The length of this range is specified by the LEN field. If the source data is specified using a Ptyp of 1, then OFF shall point to the start of a box header and the source range shall include only complete boxes; the Digital Signature box shall be at the same level in the box hierarchy as the box pointed to by the OFF field.
	All other values reserved.

OFF: Source data offset. This field specifies the offset in bytes to the start of the source data range that is signed by this Digital Signature box. This offset is relative to the first byte of the file. This field is encoded as an 8-byte big endian unsigned integer. If the value of Ptyp is 1, then this field shall not exist.

LEN: Source data length. If non-zero, this field specifies the length in bytes of the source data range that is signed by this Digital Signature box. A value of zero specifies that the end of the source data range is the last byte of the file. This field is encoded as an 8-byte big endian unsigned integer.

DATA: Signature data. This field contains the digital signature produced from the source data range. The format of this data is as specified by the Styp field.

The Format of the contents of the Digital Signature box is given in Table M.52.

Table M.52 – Format of the contents of the Digital Signature box

Parameter	Size (bits)	Value
Styp	1	0
Ptyp	1	0-1
OFF	64 0	0-(2^{64}–1); if Ptyp = 1 not applicable; if Ptyp = 0
LEN	64 0	0-(2^{64}–1); if Ptyp = 1 not applicable; if Ptyp = 0
DATA	variable	variable

M.11.18 XML box

The JP2 file format defines the XML box to contain a well-formed XML document as defined by W3C Recommendation: Extensible Markup Language (XML) 1.0. In a JPX file, this box is extended to allow JPX readers to better make use of the XML data. The format of the XML box is unchanged. However, a JPX reader should take the following actions to parse the XML document, using the definitions provided by W3C Recommendations: XML Schema Part 1; and XML Schema Part 2.

– If the reader finds the "xsi:schemaLocation" attribute in the root element, then the structure of this XML document or instance data is defined by a schema. This attribute specifies the physical location of the schema document.

– If the reader finds the "!DOCTYPE" line in the header of the XML document, then the structure of the XML document or instance data is defined by a Document Type Definition (DTD) document. This line

specifies which DTD is used by this XML document or instance data, as well as the root element name and the location of the DTD.

– If the XML schema or DTD document referred to by the XML document contained within the XML box contains a comment field of the form "<!--HUMAN_SCHEMA_DTD_LOCATION: *LOC* -->", then a reader may retrieve a human readable document from the URL specified by *LOC*. This document shall contain a human readable description of the schema or DTD. This document will support application developers and users of the metadata.

M.11.19 MPEG-7 Binary box

This box contains metadata in MPEG-7 binary format (BiM) as specified in ISO/IEC 23001-1.

The type of an MPEG-7 Binary box shall be 'mp7b' (0x6D70 3762). It may be found anywhere in the file after the Reader Requirements Box. The contents of the MPEG-7 Binary box are as follows (see Figure M.40):

Figure M.40 – Organization of the contents of a MPEG-7 Binary box

DATA: MPEG-7 BiM Stream.

M.11.20 Free box

The Free box specifies a section of the file that is not currently used and may be overwritten when editing the file. Readers shall ignore all Free boxes. A Free box may be found anywhere in the file except before the Reader Requirements box.

The type of a Free box shall be 'free' (0x6672 6565). As a free box contains meaningless data, the contents of a Free box are undefined.

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 shall also contain a Compositions box, at least one Codestream Header box and at least one Compositing Layer Header box. Moreover, all such boxes shall 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 (see Figure M.41):

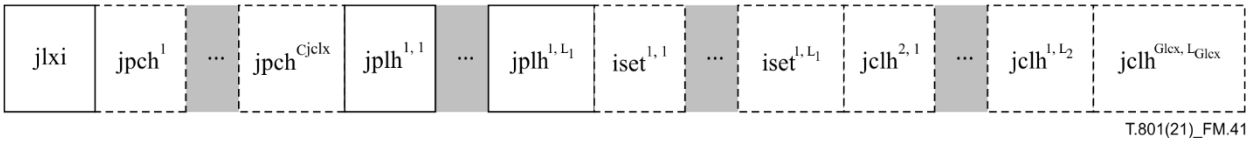

Figure M.41 – Organization of the contents of a Compositing Layer Extensions box

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 shall 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 have 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

$$Ctotal = Cprev + Mjclx \cdot Cjclx,$$

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, then 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 clause 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 rth replica of Codestream Header box $jpch^{c}$, starting from r=0, as

$$Cprev + r \cdot Cjclx + c.$$

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 rth replica of Compositing Layer header box $jplh^{g,j}$ as

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

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 $iset^{g,1}$ to $iset^{g,Ig}$ to the compositing layers formed from the Mjclx replicas of Compositing Layer Header boxes $jclh^{g,1}$ to $jclh^{g,Lg}$.

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:

$0 \le c < Ctop$, or else

$Cprev \le c < Cprev+Cjclx$,

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

$Cmap(r,c) = c$, if $c < Ctop$, else

$Cmap(r,c) = Cprev-Ctop + r \cdot Cjclx + c$, if $c \ge Ctop$

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:

$Lmap(r,n) = n$, if $n < Ltop$, else

$Lmap(r,n) = Lprev-Ltop + r \cdot Ljclx + n$, if $n \ge Ltop$

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.

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 (see Figure M.42):

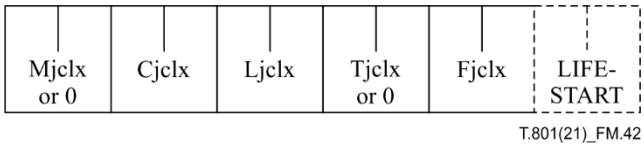

Figure M.42 – Organization of the Compositing Layer Extensions Info box

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 clause 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.

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 shall 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 1 – 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 (see Figure M.43):

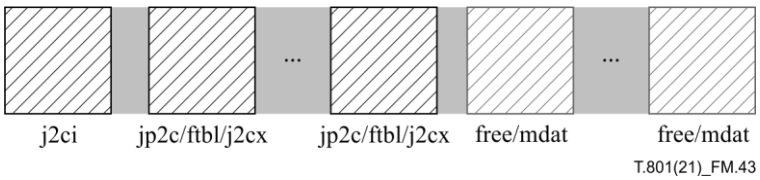

Figure M.43 – Organization of the contents of a Multiple Codestream box

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 2 – 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 3 – 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.53 – Format of the contents of the Multiple Codestream box

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

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 (see Figure M.44):

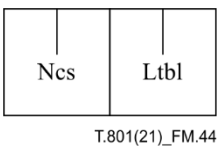

Figure M.44 – Organization of the contents of a Multiple Codestream Offsets box

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 $Ltbl = R \cdot 2^{26} + L$.

Table M.54 – Format of the contents of the Multiple Codestream Info box

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

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 (see Figure M.45):

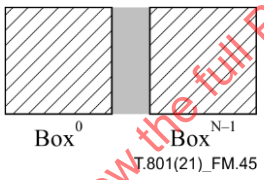

Figure M.45 – Organization of the contents of a Grouping box

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 include (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 JPIP server, using the tools and mechanisms defined in Rec. ITU-T T.808 | ISO/IEC 15444-9.

M.11.26 Decomposed XML box

The Decomposed XML box specifies front-matter from an XML document that is decomposed through a hierarchy of Association boxes. When placed as the first box inside an association box, the Decomposed XML box allows for subsequent XML elements to be further decomposed using Association boxes, as shown in clause M.11.11.

The Decomposed XML box shall only be found as the first box within an association box, and that parent association box shall contain only the Decomposed XML box, other association boxes (with XML Header boxes as their first box) and XML boxes.

The type of the Decomposed XML box shall be 'dxml' (0x786D 6C64) and it shall have the following contents (see Figure M.46):

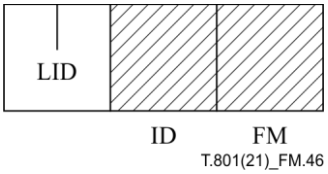

Figure M.46 – Organization of the contents of a decomposed XML box

LID: Length of the ID field, specified as a 2-byte unsigned integer

ID: A text string specifying an ID for this XML document. The length of this field is the LID field, in bytes.

FM: A text string specifying the front-matter of the XML document. This string shall contain the "<?xml>" tag and preferably all other complete XML elements describing the type of data in the document, such as namespace and schema tags, as defined in W3C Recommendations: Namespaces in XML; XML Schema Part 1; and XML Schema Part 2.

M.11.27 XML Header box

The XML Header box specifies an ID and the opening tag for a single XML element. When placed as the first box inside an association box, the XML Header box allows for a well-formed XML document to be decomposed into a hierarchical structure using boxes; The content of the XML element specified in the XML Header box are taken as boxes 2–N within the Association box. The XML Header box shall be found only as the first box inside an Association box, and that Association box shall only contain the XML Header box, other Association boxes (with XML Header boxes as their first box) and XML boxes.

The type of the XML Header box shall be 'hxml' (0x786D 6C68) and it shall have the following contents (see Figure M.47):

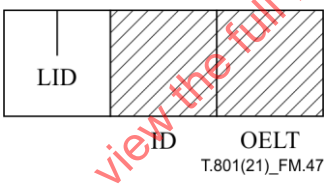

Figure M.47 – Organization of the contents of a XML header box

LID: Length of the ID field, specified as a 2-byte unsigned integer

ID: A text string specifying an ID for this XML element. The length of this field is the LID field, in bytes

OELT: A text string specifying the opening tag, including left and right angle brackets ('<' and '>') and all attributes for this XML element

M.12 Dealing with unknown boxes

A conforming JPX file may contain boxes not known to applications based solely on this Recommendation | International Standard. If a conforming reader finds a box that it does not understand, it shall skip and ignore that box.

M.13 Using the JPX file format in conjunction with other multi-media standards (informative)

While the JPX file format provides a powerful architecture for storing still-images, there are many applications in which still-images are stored in conjunction with other multi-media types. For example, many digital still cameras allow the user to capture an audio annotation to describe a particular photograph.

This integration with other multimedia types is facilitated by the use of the Box structure for encapsulating data within the JP2 and JPX file format. The box structure itself has the same binary definition as a QuickTime atom or an MPEG-4 atom. As such, a file can be created using both JPX boxes and Quicktime or MPEG-4 atoms. Provided all offsets within the file are correct with respect to the data location from the beginning of the file (and take into account the presence of all boxes and atoms), a dual-mode file can be created.

For example, it is very easy to create a file that contains both a still photograph and an audio annotation. The boxes required to store the still photograph file can be combined with the atoms required to store a MPEG-4 audio file into a single file, as the MPEG-4, JPX and JP2 formats are flexible with respect to the location of many important boxes or

atoms. A file writer would only need to be concerned that the offsets within the boxes and atoms are all determined such that they point to the location of the data in the combined file.

A reader that supports only the JPX file format would treat the file as a photograph. A reader that supports only the MPEG-4 audio standard would treat the file as an audio file. A new reader that supports both standards could then provide advanced features by combining photographic capabilities with audio capabilities.

M.14 Decomposing an XML document into multiple boxes

Using the Decomposed XML box, the XML Header box, and the Association box, a single XML document can be decomposed into multiple boxes; this may be necessary to break a large document into smaller chunks that can be retrieved separately through an interactive protocol.

While file writers are given some flexibility as to how a document is broken up, there is one important restriction. If a specific element is to be decomposed using an XML Header box and an Association box, then all ancestors of that element, up to the root of the original XML document shall also be decomposed.

Within the structure rooted at the Association box containing the Decomposed XML box, the file shall contain only Decomposed XML boxes, XML Header boxes, XML boxes, and other Association boxes. Other boxes shall not be found within the hierarchy.

NOTE – While it is conforming to include a decomposed XML document in an isolated JPX file, this capability is intended to provide a mechanism to decompose the contents of an XML box into a sequence of box in situations where only portions of the XML box are to be returned or transmitted, for example by a JPEG 2000 image server operating in the context of Rec. ITU-T T.808 | ISO/IEC 15444-9 (JPIP). In these cases, the server might transcode a standard XML box into a sequence of boxes upon loading the original file which would facilitate selective delivery of the portions of the XML content of interest by a JPIP server, using the placeholder and stream equivalence tools and mechanisms defined in Rec. ITU-T T.808 | ISO/IEC 15444-9.

Informative Example: Consider the following well-formed XML document:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:po="http://www.example.com/graphic"
      targetNamespace="http://www.example.com/graphic"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">
<shape id="box" z="2">
      <rect x="20" y="19" w="40" h="28" />
</shape>
<smallshape id="spot" z="0">
      <point x="40" y="22">
</smallshape>
<shape id="pinhole" z="1">
      <circle x="40" y="22" r="10" />
</shape>
```

This XML document might be decomposed as indicated in Figure M.48.

'dxml'

ID: Camera graphic

FM:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:po="http://www.example.com/graphic"
      targetNamespace="http://www.example.com/graphic"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">
```

'asoc'

'hxml':

ID: Lense

OELT:

```
<shape id="pinhole" z="1">
```

'xml ':

```
<smallshape id="spot" z="0">
<point x="40" y="22">
</smallshape>
```

'asoc'

'hxml'

ID: Camera body

OELT:

```
<shape id="box" z="2">
```

'xml':

```
<circle x="40" y="22" r="10" />
```

Figure M.48 – Example Box layout for a Decomposed XML box

Annex N

JPX file format extended metadata definition and syntax

(This annex does not form an integral part of this Recommendation | International Standard.)

This annex defines a comprehensive set of metadata elements that may be embedded in a JPX file within XML boxes. Use of, and conformance to, this form of metadata is optional for JPX readers.

This annex is not actively maintained and should not be used for new applications.

N.1 Introduction to extended metadata

Metadata is additional information that is associated with the primary data (the image). In the context of this Recommendation | International Standard, it is additional data linked with the image data beyond the pixels which define the image. Metadata, to be most valuable for the owner(s) and user(s) of an image, needs to be consistently maintained throughout the image lifecycle. In today's environment of image editing applications, rapid transmission via the Internet, and high quality photographic printers, the lifecycle of a digital image may be very long as well as complex.

Image metadata is a building block for digital imaging that may be used within the wide spectrum of the imaging workflow. This annex defines a standard set of image metadata based on a generic concept that may be further divided into conceptual metadata groups. Each of these groups describes a unique aspect of the image. By partitioning metadata into discrete groups, users may extend a particular block without affecting the entire architecture thereby ensuring semantic interoperability while allowing others to add value to the metadata and image data itself.

N.2 Additional references for extended metadata

- ASTM E1708-95: *Standard Practice for Electronic Interchange of Color and Appearance Data*, 1995.
- IETF RFC 2426: *vCard MIME Directory Profile*, September 1998.
- ISO 12232:1998, *Photography – Electronic still-picture cameras – Determination of ISO speed.*
- ISO 12233:2000, *Photography – Electronic still-picture cameras – Resolution measurements.*
- ISO 14524:1999, *Photography – Electronic still-picture cameras – Methods for measuring opto-electronic conversion functions (OECFs).*
- JEIDA: *Digital Still Camera File Format Standard (Exif). Version 2.1*, June 1998.
- WIPO. Berne Convention for the Protection of Literary and Artistic Works. Paris Act of 24 July 1971, amended 28 September 1979.
- WIPO. World Intellectual Property Organization Copyright Treaty, 1996.

N.3 Scope of metadata definitions

This annex consists of five logical groups of metadata as well as common definitions of dataypes that are referred to by other metadata definitions. While each group is logically partitioned, they may be linked to each other to form additional semantics.

N.3.1 Image Creation metadata

The Image Creation metadata defines the "how" metadata that specifies the source of which the image was created. For example, the camera and lens information and capture condition are useful technical information for professional and serious amateur photographers as well as advanced imaging applications.

N.3.2 Content Description metadata

The Content Description metadata defines the descriptive information of "who," "what," "when," and "where" aspect of the image. Often this metadata takes the form of extensive words, phrases, or sentences to describe a particular event or location that the image illustrates. Typically, this metadata consists of text that the user enters, either when the images are taken or scanned or later in the process during manipulation or use of the images.

N.3.3 History metadata

The History is used to provide partial information about how the image got to the present state. For example, history may include certain processing steps that have been applied to an image. Another example of a history would be the image creation events including digital capture, exposure of negative or reversal films, creation of prints, transmissive scans of negatives or positive film, or reflective scans of prints. All of this metadata is important for some applications. To permit flexibility in construction of the image history metadata, two alternate representations of the history are permitted. In the first, the history metadata is embedded in the image metadata. In the second, the previous versions of the image, represented as a URL/URI, are included in the history metadata as pointers to the location of the actual history. The history metadata for a composite image (i.e., created from two or more previous images) may also be represented through a hierarchical metadata structure. While this Specification does not define the "how" or "how much" part of the processing aspect, it does enable logging of certain processing steps applied to an image as hints for future use.

N.3.4 Intellectual Property Rights metadata

The Intellectual Property Rights (IPR) metadata defines metadata to either protect the rights of the owner of the image or provide further information to request permission to use it. It is important for developers and users to understand the implications of intellectual property and copyright information on digital images to properly protect the rights of the owner of the image data.

N.3.5 Fundamental metadata types and elements

The Fundamental metadata types define common datatypes that may be used within each metadata groups. Those include an address type or a persona type which is a collection of other primitive datatypes. The Fundamental metadata elements define elements that are commonly referenced within other metadata groups. These include a definition for language specification and a timestamp.

N.3.6 Image Identifier metadata

Image Identifier metadata is used to uniquely identify the image.

N.4 Metadata syntax

As defined in ITU-T T.800 | ISO/IEC 15444-1 Annex I, the JP2 file format allows XML format metadata to be contained within the box structure. Metadata defined by this annex shall be well-formed XML as defined by XML 1.0. The XML shall conform to all the normative requirements of N.6, not just those expressed in the DTD and the XML Schema. The default character encoding shall be UTF-8 unless otherwise specified in the XML document.

N.4.1 Metadata schema definition language

This Recommendation | International Standard uses the XML Schema syntax as defined by XML Schema Part 1 and XML Schema Part 2 to describe the elements of the metadata.

N.4.2 Namespace

XML namespace is a collection of names, identified by a Universal Resource Identifier (URI), that allows XML documents of different sources to use elements with the same names, to be merged within a single document with no confusion. Considering JPX metadata, either incorporating other metadata for extensibility or being used in other applications, it is important to define a XML namespace for JPX elements and attributes. To specify the JPX XML namespace the following URI is defined.

```
xmlns:xsd="http://www.jpeg.org/jpx"
```

The following namespace are used for XML and XML Schema defined elements, attributes and values:

```
xmlns:xml="http://www.w3.org/XML/1998/namespace/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

N.4.3 Document type definition information

An XML Document type definition (DTD) for this Recommendation | International Standard is defined by the DTD specified in clause N.8.

The Formal Public Identifier (FPI) for this DTD shall be:

```
PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN"
```

This FPI shall be used on the DOCTYPE declaration within a XML document referencing the DTD defined by this Recommendation | International Standard.

The following URL references the DTD for this Recommendation | International Standard:

"http://www.jpeg.org/metadata/15444-2.dtd"

In metadata defined by this annex, a DOCTYPE declaration shall be present prior to the root element of the XML document. The name in the DOCTYPE declaration shall be set to the root element name for the defined boxes in clause N.5. The system identifier may be modified appropriately to reference the DTD expressed in clause N.8.

N.4.4 XML Schema information

An XML Schema for this Recommendation | International Standard is defined by the XML Schema specified in clause N.9.

The following URL references the XML Schema for this Recommendation | International Standard:

"http://www.jpeg.org/metadata/15444-2.xsd"

Where an XML Schema location is used in metadata defined by this annex, the root element shall contain an xsi:schemaLocation attribute listing the jp namespace as specified in clause N.4.2 and the appropriate URL reference of the XML Schema file expressed in clause N.9.

N.5 Defined boxes

The following boxes are defined as part of JPX file format extended metadata. All boxes defined in this annex are optional unless otherwise stated. If a JPX reader supports a given box, it shall understand all the elements within the box.

N.5.1 Image Creation metadata box

The Image Creation metadata box defines metadata that are related to the creation of a digital image. The scope of this box is applicable to metadata elements that are relevant to the creation of the digital image data, i.e., camera and scanner device information and its capture condition as well as the software or firmware to create such image. It defines the "how" metadata that specifies the origin of the image.

The type of the Image Creation box shall be 'xml\040' (0x786D 6C20) as defined in ITU-T T.800 | ISO/IEC 15444-1, clause I.7.1. The contents of this box shall be as follows (see Figure N.1):

Figure N.1 – Organization of the contents of Image Creation box

ICre: Image Creation metadata field. This field shall be well-formed XML as defined by XML 1.0.

Table N.1 – Format of the contents of the Image Creation box

Field name	Size (bits)	Value
ICre	Variable	This field contains an XML document as defined in clause N.4, with the root element IMAGE_CREATION, containing metadata defined in clause N.6.1.

N.5.2 Content Description metadata box

This box comprises the content description of an image. The content description has two main purposes:

– Firstly: It can be used to classify the image. Images placed in a database need to be extracted from that database. For any image to be useful (happy snaps saved in the file system of a personal computer through to an extensive professional photo library), this is required. This classification may be used to search for images.

– Secondly: Once an image is retrieved, some data which describes the image but is not useful when searching may be included. For example – "Bob is the guy asleep on the lounge" is not all that useful when searching, but is useful when describing the content.

The metadata listed in this box contains data for both of the above cases.

The type of the Content Description box shall be 'xml\040' (0x786D 6C20) as defined in ITU-T T.800 | ISO/IEC 15444-1, clause I.7.1. The contents of this box shall be as follows (see Figure N.2):

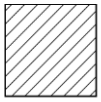

Figure N.2 – Organization of the contents of Content Description box

CDes: Content Description field. This field shall be well-formed XML as defined by XML 1.0.

Table N.2 – Format of the contents of the Content Description box

Field name	Size (bits)	Value
CDes	Variable	This field contains an XML document as defined in clause N.4, with the root element CONTENT_DESCRIPTION, containing metadata defined in clause N.6.2.

N.5.3 History box

This box contains the history of metadata of an image. The History metadata is used to provide partial information about how the picture got to the present state. This data is only approximate because:

– some of the data is collapsed, thus providing only a summary;

– some of the data may not have been properly entered because applications used were not able to update the history metadata.

The History box contains a summary of basic image editing operations that have already been applied to the image and previous version(s) of the image metadata. The History metadata is not designed to be used to reverse (undo) image editing operations.

The type of the History box shall be 'xml\040' (0x786D 6C20) as defined in ITU-T T.800 | ISO/IEC 15444-1, clause I.7.1. The contents of this box shall be as follows (see Figure N.3):

Figure N.3 – Organization of the contents of History box

Hist: History field. This field shall be well-formed XML as defined by XML 1.0.

Table N.3 – Format of the contents of the History box

Field name	Size (bits)	Value
MHist	Variable	This field contains an XML document as defined in clause N.4, with the root element HISTORY, containing metadata defined in clause N.6.3.

N.5.4 Intellectual Property Rights box

This box contains Intellectual property rights (IPR) related information associated with the image such as moral rights, copyrights as well as exploitation information.

The type of the Intellectual property rights box shall be 'jp2i' (0x6A70 3269) as defined in ITU-T T.800 | ISO/IEC 15444-1, clause I.6. The contents of this box shall be as follows (see Figure N.4):

Figure N.4 – Organization of the contents of Intellectual Property Rights box

IPR: Intellectual Property Rights field. This field shall be well-formed XML as defined by XML 1.0.

Table N.4 – Format of the contents of the Intellectual Property Rights box

Field name	Size (bits)	Value
IPR	Variable	This field contains an XML document as defined in clause N.4, with the root element IPR, containing metadata defined in clause N.6.4.

N.5.5 Image Identifier box

This box contains the image identifier metadata of an image. The Image Identifier metadata is used to uniquely identify the image.

The type of the Image Identifier box shall be 'xml\040' (0x786D 6C20) as defined in ITU-T T.800 | ISO/IEC 15444-1, clause I.7.1. The contents of this box shall be as follows (see Figure N.5):

Figure N.5 – Organization of the contents of Image Identifier box

ImgId: Image Identifier field. This field shall be well-formed XML as defined by XML 1.0.

Table N.5 – Format of the contents of the Image Identifier box

Field name	Size (bits)	Value
ImgId	Variable	This field contains an XML document as defined in clause N.4, with the root element IMAGE_ID, containing metadata defined in clause N.6.5.

N.6 Metadata definitions

This clause specifies the metadata element syntax and semantics defined as part of JPX file format extended metadata. Each of the following metadata elements is based on the XML format as defined in XML 1.0. A JPX reader shall ignore metadata elements it does not understand.

N.6.1 Image Creation metadata

This element specifies information that are relevant to the creation of the image file. This element may contain the sub-elements listed below. See also Figure N.6.

```
<xsd:element name="IMAGE_CREATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="jp:GENERAL_CREATION_INFO" minOccurs="0"/>
      <xsd:element ref="jp:CAMERA_CAPTURE" minOccurs="0"/>
      <xsd:element ref="jp:SCANNER_CAPTURE" minOccurs="0"/>
      <xsd:element ref="jp:SOFTWARE_CREATION" minOccurs="0"/>
      <xsd:element ref="jp:CAPTURED_ITEM" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
```

Figure N.6 – Schema of the Image Creation metadata

GENERAL_CREATION_INFO: General creation information. This element specifies generic information on how the image was created. The syntax of this element is specified in clause N.6.1.1.

CAMERA_CAPTURE: This element specifies a camera capture metadata of a scene. The syntax of this element is specified in clause N.6.1.2.

SCANNER_CAPTURE: This element specifies scanner capture metadata that may be used for various scanners such as flatbed and film scanners. The syntax of this element is specified in clause N.6.1.8.

SOFTWARE_CREATION: This element specifies software information that created the original digital image. The syntax of this element is specified in clause N.6.1.10.

CAPTURED_ITEM: This element contains description of the item that was digitally captured. The syntax of this element is specified in clause N.6.1.11.

N.6.1.1 General Creation Information metadata

This element specifies general information on how the image was created. Applications may choose to skip further parsing based on the values stored here. For example, if the application is only interested in digital camera metadata, it can skip additional parsing based on the Image source value. This element may contain the sub-elements listed below. See also Figure N.7.

```
  <xsd:element name="GENERAL_CREATION_INFO">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CREATION_TIME" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="IMAGE_SOURCE" type="jp:tLangString" minOccurs="0"/>
        <xsd:element name="SCENE_TYPE" type="jp:tLangString" minOccurs="0"/>
        <xsd:element name="IMAGE_CREATOR" type="jp:tPerson" minOccurs="0"/>
        <xsd:element name="OPERATOR_ORG" type="jp:tOrganization"
minOccurs="0"/>
        <xsd:element name="OPERATOR_ID" type="jp:tLangString" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.7 – Schema of the General Creation Information metadata

CREATION_TIME: This element specifies the date and time the image was created. This element should be stored when the creation process started. (E.g., it may be a 8 minute exposure.) This element should never be changed after it is written in the image creation device.

IMAGE_SOURCE: This element specifies the device source of the digital file, such as a film scanner, reflection print scanner, or digital camera. Table N.6 lists suggested values for this element.

Table N.6 – Image Source values

Value	Meaning
Digital Camera	Image create by a digital camera
Film Scanner	Image create by a film scanner
Reflection Print Scanner	Image create by a reflection print scanner (commonly referred to as flat bed)
Still From Video	Image create by from video
Computer Graphics	Image digitally created on computers

SCENE_TYPE: This element specifies the type of scene that was captured. It differentiates "original scenes" (direct capture of real-world scenes) from "second generation scenes" (images captured from pre-existing hardcopy images). It provides further differentiation for scenes that are digitally composed. Table N.7 lists suggested values for this element.

Table N.7 – Scene type values

Value	Meaning
Original Scene	Direct capture of real-world scenes
Second Generation Scene	Images captured from pre-existing hardcopy images such a photograph
Digital Scene Generation	Graphic arts or images digitally composed

IMAGE_CREATOR: This element specifies the name of the image creator. The image creator may be, for example, the photographer who captured the original picture on film, the illustrator or graphic artist who conducted the image-creation process, etc. See Person type (clause N.7.1.13) for the format of this element.

OPERATOR_ORG: Operator organization. This element specifies the name of the service bureau, photofinisher, or organization where the image capture process (photographed, scanned or created by software) is conducted. See Organization type (clause N.7.1.14) for the format of this element.

OPERATOR_ID: This element specifies a name or ID for the person conducting the capture process.

N.6.1.2 Camera Capture metadata

This element specifies a camera capture of a scene. This element may contain camera and lens information, device characterization and camera capture settings. See also Figure N.8.

```
  <xsd:element name="CAMERA_CAPTURE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element       name="CAMERA_INFO"       type="jp:tProductDetails"
minOccurs="0"/>
        <xsd:element      name="SOFTWARE_INFO"      type="jp:tProductDetails"
minOccurs="0"/>
        <xsd:element        name="LENS_INFO"        type="jp:tProductDetails"
minOccurs="0"/>
        <xsd:element ref="jp:DEVICE_CHARACTER" minOccurs="0"/>
        <xsd:element ref="jp:CAMERA_SETTINGS" minOccurs="0"/>
        <xsd:element name="ACCESSORY" type="jp:tProductDetails" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.8 – Schema of the Camera Capture metadata

CAMERA_INFO: Camera information. This element specifies information of the camera that captured the image. See Product Details type (clause N.7.1.21) for the format of this element.

SOFTWARE_INFO: Software information. This element specifies information about the software or firmware used to capture the image. See Product Details type (clause N.7.1.21) for the format of this element.

LENS_INFO: Lens information. This element specifies information about the lens that captured the image. See Product Details type (clause N.7.1.21) for the format of this element.

DEVICE_CHARACTER: Device characterization. This element specifies the technical characterization of the digital capture device. The syntax of this element is specified in clause N.6.1.3.

CAMERA_SETTINGS: Camera capture settings. This element specifies the camera settings used when the image was captured. The syntax of this element is specified in clause N.6.1.7.

ACCESSORY: This element specifies the information of the accessories used with the camera to capture the image. Professional and amateur photographers may want to keep track of a variety of miscellaneous technical information, such as the use of extension tubes, bellows, close-up lenses, and other specialized accessories. See Product Details type (clause N.7.1.21) for the format of this element.

N.6.1.3 Device Characterization metadata

This element specifies the technical characterization of the digital capture device. This element may contain the sub-elements listed below. See also Figure N.9.

```
<xsd:element name="DEVICE_CHARACTER">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="SENSOR_TECHNOLOGY" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="One-Chip Color Area"/>
            <xsd:enumeration value="Two-Chip Color Area"/>
            <xsd:enumeration value="Three-Chip Color Area"/>
            <xsd:enumeration value="Color Sequential Area"/>
            <xsd:enumeration value="Trilinear"/>
            <xsd:enumeration value="Color Sequential Linear Sensor"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element       name="FOCAL_PLANE_RES"       type="jp:tDoubleSize"
minOccurs="0"/>
      <xsd:element       name="SPECTRAL_SENSITIVITY"       type="xsd:string"
minOccurs="0"/>
      <xsd:element    name="ISO_SATURATION"    type="jp:tNonNegativeDouble"
minOccurs="0"/>
      <xsd:element       name="ISO_NOISE"       type="jp:tNonNegativeDouble"
minOccurs="0"/>
      <xsd:element ref="jp:SPATIAL_FREQ_RESPONSE" minOccurs="0"/>
      <xsd:element ref="jp:CFA_PATTERN" minOccurs="0"/>
      <xsd:element ref="jp:OECF" minOccurs="0"/>
      <xsd:element     name="MIN_F_NUMBER"     type="jp:tNonNegativeDouble"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.9 – Schema of the Device Characterization metadata

SENSOR_TECHNOLOGY: This element specifies either the type of image sensor or the sensing method used in the camera or image-capturing device. Table N.8 lists suggested values for this element.

Table N.8 – Sensor technology values

Value	Meaning
One-Chip Colour Area	An one-chip colour area sensor technology used.
Two-Chip Colour Area	A two-chip colour area sensor technology used.
Three-Chip Colour Area	A three-chip colour area sensor technology used.
Colour Sequential Area	A colour sequential area sensor technology used.
Trilinear	An trilinear sensor technology used.
Colour Sequential Linear Sensor	A colour sequential linear sensor technology used.

FOCAL_PLANE_RES: Focal plane resolution. This element specifies the number of pixels per meter in the X (width) and Y (height) directions for the main image. The resolution stored is the resolution of the image generated rather than the width and height of the image sensor.

SPECTRAL_SENSITIVITY: This element specifies the spectral sensitivity of each channel of the camera used to capture the image. It is useful for certain scientific applications. The contents of this element is compatible with ASMT E1708-95 and expected to be defined by another standard. If the Spectral Sensitivity data contains a "<" or "&" characters, then all of the occurrences of "<" shall be substituted with "<" and "&" shall be substituted with "&".

ISO_SATURATION: ISO saturation speed rating. This element specifies the ISO saturation speed rating classification as defined in ISO 12232.

ISO_NOISE: ISO noise speed rating. This element specifies the ISO noise-based speed rating classification as defined in ISO 12232.

SPATIAL_FREQ_RESPONSE: Spatial frequency response. This element specifies the Spatial Frequency Response (SFR) of the image capturing device. The syntax of this element is specified in clause N.6.1.4.

CFA_PATTERN: Colour filter array pattern. This element specifies the colour filter array (CFA) pattern of the image sensor used to capture a single-sensor colour image. The syntax of this element is specified in clause N.6.1.5.

OECF: Opto-electronic conversion function. This element specifies the Opto-Electronic Conversion Function (OECF). The OECF is the relationship between the optical input and the image file code value outputs of an electronic camera. The property allows OECF values defined in ISO 14524 to be stored as a table of values. The syntax of this element is specified in clause N.6.1.6.

MIN_F_NUMBER: Minimum F-number. This element specifies the minimum lens f-number of the camera or image capturing device.

NOTE – ISO 12232 :1998 and ISO 14524 :1999, which were in-force at the time of initial publication of this Recommendation | International Standard, have been superseded by more recent editions.

N.6.1.4 Spatial Frequency Response metadata

This specifies the Spatial Frequency Response (SFR) of the image capturing device. The device measured SFR data, described in ISO 12233, can be stored as a table of spatial frequencies, horizontal SFR values, vertical SFR values, and diagonal SFR values. See also Figure N.10.

```
<xsd:element name="SPATIAL_FREQ_RESPONSE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="SPATIAL_FREQ_VAL" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="SPATIAL_FREQ" type="jp:tNonNegativeDouble"/>
            <xsd:element name="HORIZ_SFR" type="jp:tNonNegativeDouble"/>
            <xsd:element name="VERT_SFR" type="jp:tNonNegativeDouble"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.10 – Schema of the Spatial Frequency Response metadata

SPATIAL_FREQ_VAL: Spatial frequency value. This element specifies the list of SFR values.

SPATIAL_FREQ: Spatial frequency value in line widths per picture height units.

HORIZ_SFR: Horizontal SFR value.

VERT_SFR: Vertical SFR value.

NOTE – ISO 12233:2000, which was in-force at the time of initial publication of this Recommendation | International Standard, has been superseded by more recent editions.

N.6.1.5 Colour Filter Array Pattern metadata

This element encodes the actual colour filter array (CFA) geometric pattern of the image sensor used to capture a single-sensor colour image. It is not relevant for all sensing methods. The data contains the minimum number of rows and columns of filter colour values that uniquely specify the colour filter array. See also Figure N.11.

```
<xsd:element name="CFA_PATTERN">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="COLOR_ROW" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="COLOR" maxOccurs="unbounded">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="Red"/>
                  <xsd:enumeration value="Green"/>
                  <xsd:enumeration value="Blue"/>
                  <xsd:enumeration value="Cyan"/>
                  <xsd:enumeration value="Magenta"/>
                  <xsd:enumeration value="Yellow"/>
                  <xsd:enumeration value="White"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.11 – Schema of the Colour Filter Array Pattern metadata

COLOR_ROW: This element specifies the list of colour values of the CFA pattern.

COLOR: CFA pattern values. The values shall be either Red, Green, Blue, Cyan, Magenta, Yellow, or White.

N.6.1.6 Opto-electronic Conversion Function metadata

This element specifies the Opto-Electronic Conversion Function (OECF). The OECF is the relationship between the optical input and the image file code value outputs of an electronic camera. The property allows OECF values defined in ISO 14524 to be stored as a table of values. See also Figure N.12.

```
<xsd:element name="OECF">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="LOG_VAL" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="LOG_EXPOSURE" type="xsd:double"/>
              <xsd:element name="OUTPUT_LEVEL" type="jp:tNonNegativeDouble"
maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Figure N.12 – Schema of the Opto-electronic Conversion Function metadata

LOG_VAL: This element specifies the list of OECF values.

LOG_EXPOSURE: Optical input log exposure value.

OUTPUT_LEVEL: Image file code value output value.

N.6.1.7 Camera Capture Settings metadata

This element specifies the camera settings used when the image was captured. New generations of digital and film cameras make it possible to capture more information about the conditions under which a picture was taken. This may include information about the lens aperture and exposure time, whether a flash was used, which lens was used, etc. This technical information is useful to professional and serious amateur photographers. In addition, some of these properties are useful to image database applications for populating values useful to advanced imaging applications and algorithms as well as image analysis and retrieval. This element may contain the sub-elements listed below. See also Figure N.13.

```
  <xsd:element name="CAMERA_SETTINGS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice minOccurs="0">
          <xsd:element name="EXP_TIME" type="jp:tNonNegativeDouble"/>
          <xsd:element name="R_EXP_TIME" type="jp:tRational"/>
        </xsd:choice>
        <xsd:element       name="F_NUMBER"       type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="EXP_PROGRAM" type="jp:tLangString" minOccurs="0"/>
        <xsd:element name="BRIGHTNESS" type="xsd:double" minOccurs="0"/>
        <xsd:element name="EXPOSURE_BIAS" type="xsd:double" minOccurs="0"/>
        <xsd:element    name="SUBJECT_DISTANCE"    type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element        name="METERING_MODE"        type="jp:tLangString"
minOccurs="0"/>
        <xsd:element       name="SCENE_ILLUMINANT"       type="jp:tLangString"
minOccurs="0"/>
        <xsd:element      name="COLOR_TEMP"      type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element     name="FOCAL_LENGTH"     type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="FLASH" type="xsd:boolean" minOccurs="0"/>
        <xsd:element     name="FLASH_ENERGY"     type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="FLASH_RETURN" type="xsd:boolean" minOccurs="0"/>
        <xsd:element name="BACK_LIGHT" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Front Light"/>
              <xsd:enumeration value="Back Light 1"/>
              <xsd:enumeration value="Back Light 2"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element        name="SUBJECT_POSITION"        type="jp:tPosition"
minOccurs="0"/>
        <xsd:element name="EXPOSURE_INDEX" type="xsd:double" minOccurs="0"/>
        <xsd:element name="AUTO_FOCUS" minOccurs="0">
          <xsd:simpleType>
```

```
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Auto Focus Used"/>
              <xsd:enumeration value="Auto Focus Interrupted"/>
              <xsd:enumeration value="Near Focused"/>
              <xsd:enumeration value="Soft Focused"/>
              <xsd:enumeration value="Manual"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element            name="SPECIAL_EFFECT"            minOccurs="0"
maxOccurs="unbounded">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Colored"/>
              <xsd:enumeration value="Diffusion"/>
              <xsd:enumeration value="Multi-Image"/>
              <xsd:enumeration value="Polarizing"/>
              <xsd:enumeration value="Split-Field"/>
              <xsd:enumeration value="Star"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element        name="CAMERA_LOCATION"        type="jp:tLocation"
minOccurs="0"/>
        <xsd:element name="ORIENTATION" type="jp:tDirection" minOccurs="0"/>
        <xsd:element name="PAR" type="jp:tRational" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.13 – Schema of the Camera Capture Settings metadata

EXP_TIME: Exposure time. This element specifies the exposure time used when the image was captured. The value of this element is stored in seconds.

R_EXP_TIME: Rational exposure time. This element specifies the exposure time used when the image was captured. The value of this element is stored in rational values in seconds.

F_NUMBER: F-Number. This element specifies the lens f-number (ratio of lens aperture to focal length) used when the image was captured.

EXP_PROGRAM: Exposure program. This element specifies the class of exposure program that the camera used at the time the image was captured. Table N.9 lists suggested values for this element.

Table N.9 – Exposure program values

Value	Meaning
Manual	The exposure setting set manually by the photographer.
Program Normal	A general purpose auto-exposure program.
Aperture Priority	The user selected the aperture, and the camera selected the shutter speed for proper exposure.
Shutter Priority	The user selected the shutter speed, and the camera selected the aperture for proper exposure.
Program Creative	The exposure setting is biased toward greater depth of element.
Program Action	The exposure setting is biased toward faster shutter speed.
Portrait Mode	The exposure setting is intended for close-up photos with the background out of focus.
Landscape Mode	The exposure setting is intended for landscapes with the background in good focus.

BRIGHTNESS: Brightness value. This element specifies the Brightness Value (Bv) measured when the image was captured, using APEX units. The expected maximum value is approximately 13.00 corresponding to a picture taken of a snow scene on a sunny day, and the expected minimum value is approximately –3.00 corresponding to a night scene. If the value supplied by the capture device represents a range of values rather than a single value, the minimum and maximum value may be specified.

EXPOSURE_BIAS: Exposure bias value. This element specifies the actual exposure bias (the amount of over-or underexposure relative to a normal exposure, as determined by the camera's exposure system) used when capturing the image, using APEX units. The value is the number of exposure values (stops). For example, –1.00 indicates 1 eV (1 stop) underexposure, or half the normal exposure.

SUBJECT_DISTANCE: This element specifies the distance between the front nodal plane of the lens and the subject on which the camera was focusing. The camera may have focused on a subject within the scene that may not have been the primary subject. The subject distance may be specified by a single number if the exact value is known. Alternatively, a range of values indicating the minimum and maximum distance of the subject may be set. The value of this element is in meters.

METERING_MODE: This element specifies the metering mode (the camera's method of spatially weighting the scene luminance values to determine the sensor exposure) used when capturing the image. Table N.10 lists suggested values for this element.

Table N.10 – Metering mode values

Value	Meaning
Average	Average mode used.
Center Weighted Average	Center weighted average mode used.
Spot	Spot mode used.
MultiSpot	MultiSpot mode used.
Pattern	Pattern mode used.
Partial	Partial mode used.

SCENE_ILLUMINANT: This element specifies the light source (scene illuminant) that was present when the image was captured. Table N.11 lists suggested values for this element.

Table N.11 – Scene illuminant values

Value	Meaning
Daylight	Daylight illuminant used.
Fluorescent Light	Fluorescent light used.
Tungsten Lamp	Tungsten lamp used.
Flash	Flash used.
Standard Illuminant A	Standard illuminant A used.
Standard Illuminant B	Standard illuminant B used.
Standard Illuminant C	Standard illuminant C used.
D55 Illuminant	D55 illuminant used.
D65 Illuminant	D65 illuminant used.
D75 Illuminant	D75 illuminant used.

COLOR_TEMP: Colour temperature. This element specifies the actual colour temperature value of the scene illuminant stored in units of Kelvin.

FOCAL_LENGTH: This element specifies the lens focal length used to capture the image. The focal length may be specified by using a single number, for a fixed focal length lens or a zoom lens, if the zoom position is known. The value of this element is stored in meters.

FLASH: This element specifies whether flash was used at image capture.

FLASH_ENERGY: This element specifies the amount of flash energy that was used. The measurement units are Beam Candle Power Seconds (BCPS).

FLASH_RETURN: This element specifies whether the camera judged that the flash was not effective at the time of exposure.

BACK_LIGHT: This element specifies the camera's evaluation of the lighting conditions at the time of exposure. Table N.12 lists BACK_LIGHT values used for lighting situations.

Table N.12 – Back light values

Value	Meaning
Front Light	The subject is illuminated from the front side.
Back Light 1	The brightness value difference between the subject centre and the surrounding area is greater than one full step (APEX). The frame is exposed for the subject centre.
Back Light 2	The brightness value difference between the subject centre and the surrounding area is greater than one full step (APEX). The frame is exposed for the surrounding area.

SUBJECT_POSITION: This element specifies the approximate position of the subject in the scene. See Position Type for the format of this element.

EXPOSURE_INDEX: This element specifies the exposure index setting the camera selected.

AUTO_FOCUS: This element specifies the status of the focus of the capture device at the time of capture. Table N.13 lists values used for auto focus status.

Table N.13 – Auto focus values

Value	Meaning
Auto Focus Used	The camera successfully focused on the subject.
Auto Focus Interrupted	The image was captured before the camera had successfully focused on the subject.
Near Focused	The camera deliberately focused at a distance closer than the subject to allow for the super-imposition of a focused foreground subject.
Soft Focused	The camera deliberately did not focus exactly at the subject distance to create a softer image (commonly used for portraits).
Manual	The camera was focused manually.

SPECIAL_EFFECT: Special Effects. This element specifies the types of special effects filters used. It contains a list of filter elements, where the order of the elements in the array indicates the stacking order of the filters. The first value in the array is the filter closest to the original scene. This element specifies the special effect filter used. Legal values are Coloured, Diffusion, Multi-Image, Polarizing, Split-Field, Star.

CAMERA_LOCATION: This element specifies the location of the camera when the picture was taken. See Location Type for the format of this element.

ORIENTATION: This element specifies the orientation of the camera when the picture was taken. See Direction Type for the format of this element.

PAR: Print aspect ratio. This element specifies the print aspect ratio specified by the user when the picture was taken.

N.6.1.8 Scanner Capture metadata

This element specifies scanner capture metadata that may be used for various scanners such as flatbed and film scanners. It optionally contains scanner information, device characterization and scanner capture settings. This element may contain the sub-elements listed below. See also Figure N.14.

```
  <xsd:element name="SCANNER_CAPTURE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element      name="SCANNER_INFO"       type="jp:tProductDetails"
minOccurs="0"/>
        <xsd:element      name="SOFTWARE_INFO"      type="jp:tProductDetails"
minOccurs="0"/>
        <xsd:element ref="jp:SCANNER_SETTINGS" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.14 – Schema of the Scanner Capture metadata

SCANNER_INFO: Scanner information. This element specifies information about a particular scanner that was used to digitize an image item. It is recommended that applications are able to create a unique value of the scanner by combining all elements. See Product Details type (clause N.7.1.21) for the format of this element.

SOFTWARE_INFO: Software information. This element specifies information about the software or firmware used to capture the image. See Product Details type (clause N.7.1.21) for the format of this element.

SCANNER_SETTINGS: This element specifies the scanner settings used when the image was scanned. The syntax of this element is specified in clause N.6.1.9.

N.6.1.9 Scanner Settings metadata

This element specifies the scanner settings used when the image was scanned. This element may contain the sub-elements listed below. See also Figure N.15.

```
<xsd:element name="SCANNER_SETTINGS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="PIXEL_SIZE" type="jp:tNonNegativeDouble"
minOccurs="0"/>
      <xsd:element name="PHYSICAL_SCAN_RES" type="jp:tDoubleSize"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.15 – Schema of the Scanner Settings metadata

PIXEL_SIZE: This element specifies the pixel size, in meters, of the scanner.

PHYSICAL_SCAN_RES: Physical scan resolution. These elements specify the physical scanning resolution of the device (not the interpolated resolution of the final output data) in the X (width) and Y (height) directions. The value of these elements are in meters.

N.6.1.10 Software Creation metadata

This element specifies software creation information (e.g., the application name) that created the original image. See also Figure N.16.

```
<xsd:element name="SOFTWARE_CREATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="SOFTWARE_INFO"  type="jp:tProductDetails"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.16 – Schema of the Software Creation metadata

SOFTWARE_INFO: Software information. This element specifies information about the software that created the original image. See Product Details type (clause N.7.1.21) for the format of this element.

N.6.1.11 Captured Item metadata

This element specifies capture item metadata. It optionally contains reflection print or film. This element may contain the sub-elements listed below. See also Figure N.17.

```
<xsd:element name="CAPTURED_ITEM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice>
        <xsd:element ref="jp:REFLECTION_PRINT" minOccurs="0"/>
        <xsd:element ref="jp:FILM" minOccurs="0"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.17 – Schema of the Captured Item metadata

REFLECTION_PRINT: This element specifies information about a reflection print that was digitally captured. The syntax of this element is specified in clause N.6.1.12.

FILM: This element specifies information about the film. The syntax of this element is specified in clause N.6.1.13.

N.6.1.12 Reflection Print metadata

This element specifies information about a reflection print that was digitally captured. This element may contain the sub-elements listed below. See also Figure N.18.

```
<xsd:element name="REFLECTION_PRINT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="DOCUMENT_SIZE" type="jp:tDoubleSize"
minOccurs="0"/>
      <xsd:element name="MEDIUM" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Continuous Tone Image"/>
            <xsd:enumeration value="Halftone Image"/>
            <xsd:enumeration value="Line Art"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="RP_TYPE" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="B/W Print"/>
            <xsd:enumeration value="Color Print"/>
            <xsd:enumeration value="B/W Document"/>
            <xsd:enumeration value="Color Document"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.18 – Schema of the Reflection Print metadata

DOCUMENT_SIZE: This element specifies the lengths of the X (width) and Y (height) dimension of the original photograph or document, respectively. The values of these elements are given in meters.

MEDIUM: This element specifies the medium of the original photograph, document, or artifact. Legal values include Continuous Tone Image, Halftone Image, and Line Art.

RP_TYPE: Reflection print type. This element specifies the type of the original document or photographic print. Legal values include B/W Print, Colour Print, B/W Document, and Colour Document.

N.6.1.13 Film metadata

This element specifies information on the film that was digitized. This element may contain the sub-elements listed below. See also Figure N.19.

```
<xsd:element name="FILM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="BRAND" type="jp:tProductDetails" minOccurs="0"/>
      <xsd:element name="CATEGORY" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Negative B/W"/>
            <xsd:enumeration value="Negative Color"/>
            <xsd:enumeration value="Reversal B/W"/>
            <xsd:enumeration value="Reversal Color"/>
            <xsd:enumeration value="Chromagenic"/>
            <xsd:enumeration value="Internegative B/W"/>
            <xsd:enumeration value="Internegative Color"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="FILM_SIZE" type="jp:tDoubleSize" minOccurs="0"/>
      <xsd:element name="ROLL_ID" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="FRAME_ID" type="xsd:positiveInteger"
minOccurs="0"/>
      <xsd:element name="FILM_SPEED" type="xsd:positiveInteger"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.19 – Schema of the Film metadata

BRAND: This element specifies the name of the film manufacturer. See Product Details type (clause N.7.1.21) for the format of this element.

CATEGORY: This element specifies the category of film used. Legal values include Negative B/W, Negative Colour, Reversal B/W, Reversal Colour, Chromagenic, Internegative B/W, and Internegative Colour. The category Chromagenic refers to B/W negative film that is developed with a C41 process (i.e., colour negative chemistry).

FILM_SIZE: This element specifies the size of the X and Y dimension of the film used, and the unit is in meters.

ROLL_ID: This element specifies the roll number or ID of the film. For some film, this number is encoded on the film cartridge as a bar code.

FRAME_ID: This element specifies the frame number or ID of the frame digitized from the roll of film.

FILM_SPEED: This element specifies the film speed of the film. This element is measured in the arithmetic scale specified in ISO 12232.

N.6.2 Content Description metadata

The Content Description metadata describes the content of the information captured in the image. Those are semantic information typically requiring user input. The value of such information increases as time passes. This element may contain the sub-elements listed below. See also Figure N.20.

```
<xsd:element name="CONTENT_DESCRIPTION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="GROUP_CAPTION" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="CAPTION" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="CAPTURE_TIME" type="jp:tDateTime" minOccurs="0"/>
      <xsd:element name="LOCATION" type="jp:tLocation" minOccurs="0"/>
      <xsd:element ref="jp:PERSON" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:THING" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:ORGANIZATION" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:EVENT" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:AUDIO" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:DICTIONARY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.20 – Schema of the Content Description metadata

GROUP_CAPTION: This element specifies the subject or purpose of the image. It may be additionally used to provide any other type of information related to the image.

CAPTION: This element specifies the subject or purpose of the image. It may be additionally used to provide any other type of information related to the image.

CAPTURE_TIME: This element specifies the time and date the image was initially generated. This may be different to the capture device date where the capture device is a scanner that scans the image at a different time to when it was initially captured. See DateTime type (clause N.7.1.8) for the format of this element.

LOCATION: The element describes the location of the image. This location is the physical location of the image (e.g., address, GPS coordinate), not the position of an object within the image. See Location type (clause N.7.1.15) for the format of this element.

PERSON: Person Description. This element specifies a person within an image. The syntax of this element is specified in clause N.6.2.1.

THING: Thing Description. This element specifies the names of tangible things depicted in the image. The syntax of this element is specified in clause N.6.2.2.

ORGANIZATION: Organization Description. This element specifies an organization within an image. The syntax of this element is specified in clause N.6.2.3.

EVENT: Event description. This element specifies events depicted in the image. The syntax of this element is specified in clause N.6.2.4.

AUDIO: This element specifies audio streams associated with an image. The syntax of this element is specified in clause N.6.2.7.

PROPERTY: This element specifies information used to describe an image or an object within an image. The syntax of this element is specified in clause N.6.2.8.

DICTIONARY: This element specifies a dictionary of a property. The syntax of this element is specified in clause N.6.2.9.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in this group. See Comment element (clause N.7.3.1) for the format of this element.

N.6.2.1 Person Description metadata

This element specifies a person within an image. See Person type (clause N.7.1.13) for the format of this element. This element may contain the sub-elements listed below. See also Figure N.21.

```
<xsd:element name="PERSON">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jp:tPerson">
        <xsd:sequence>
          <xsd:element name="POSITION" type="jp:tPosition" minOccurs="0"/>
          <xsd:element name="LOCATION" type="jp:tLocation" minOccurs="0"/>
          <xsd:element ref="jp:PROPERTY" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Figure N.21 – Schema of the Person Description metadata

POSITION: This element specifies the position of the person within the image. See Position type (clause N.7.1.17) for the format of this element.

LOCATION: This element specifies the physical location of the person. This element does not specify the relative position of the person. See Location type (clause N.7.1.15) for the format of this element.

PROPERTY: This element specifies additional information describing the person. See Property metadata (clause N.6.2.8) for the format of this element.

N.6.2.2 Thing Description metadata

This element specifies the names and/or properties of tangible things depicted in the image (for example, Washington Monument) or of abstract regions. This element may contain the sub-elements listed below. See also Figure N.22.

```
<xsd:element name="THING">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NAME" type="jp:tLangString" minOccurs="0"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
      <xsd:element name="POSITION" type="jp:tPosition" minOccurs="0"/>
      <xsd:element name="LOCATION" type="jp:tLocation" minOccurs="0"/>
      <xsd:element ref="jp:PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="jp:THING" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:string"/>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.22 – Schema of the Thing Description metadata

NAME: This element specifies the name of the Thing.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the Thing. See Comment element (clause N.7.3.1) for more information on this element.

POSITION: This element specifies the position of the thing within the image. See Position type (clause N.7.1.17) for the format of this element.

LOCATION: This element specifies the physical location of the thing. This element does not specify the relative position of the thing within the image. See Location type (clause N.7.1.15) for the format of this element.

PROPERTY: The Thing also contains multiple Propertys. These Propertys describe the thing. See Property metadata (clause N.6.2.8) for the format of this element.

THING: Sub-thing description. The Thing element may contain zero or more Thing elements, with the interpretation that these are sub-things of the containing thing.

ID: This element is the identifier attribute for the Thing.

N.6.2.3 Organization Description metadata

This element specifies an organization depicted within an image. This description can also be used to describe the entire image. See Organization type (clause N.7.1.14) for the format of this element. This element may contain the sub-elements listed below. See also Figure N.23.

```
  <xsd:element name="ORGANIZATION">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="jp:tOrganization">
          <xsd:sequence>
            <xsd:element name="POSITION" type="jp:tPosition" minOccurs="0"/>
            <xsd:element name="LOCATION" type="jp:tLocation" minOccurs="0"/>
            <xsd:element           ref="jp:PROPERTY"           minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

Figure N.23 – Schema of the Organization Description metadata

POSITION: This element specifies the position of the organization within the image. See Position type (clause N.7.1.17) for the format of this element.

LOCATION: This element specifies the physical location of the organization. This element does not specify the relative position of the organization. See Location type (clause N.7.1.15) for the format of this element.

PROPERTY: This element specifies additional information describing the organization. See Property metadata (clause N.6.2.8) for the format of this element.

N.6.2.4 Event Description metadata

This element specifies a description of the event depicted in the image. An Event is the most likely reason why an image is captured. This element may contain the sub-elements listed below unless otherwise stated. See also Figure N.24.

```
  <xsd:element name="EVENT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="EVENT_TYPE" type="jp:tLangString"/>
        <xsd:element name="DESCRIPTION" type="jp:tLangString" minOccurs="0"/>
        <xsd:element name="LOCATION" type="jp:tLocation" minOccurs="0"/>
        <xsd:element name="EVENT_TIME" type="jp:tDateTime" minOccurs="0"/>
        <xsd:element name="DURATION" type="xsd:duration" minOccurs="0"/>
        <xsd:element ref="jp:COMMENT" minOccurs="0"/>
        <xsd:element            ref="jp:PARTICIPANT"            minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element          ref="jp:EVENT_RELATION"           minOccurs="0"
maxOccurs="unbounded"/>
        <!-- Sub-events -->
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="jp:EVENT"/>
          <xsd:element name="EVENT_REF" type="xsd:string"/>
        </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="ID" type="xsd:string"/>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.24 – Schema of the Event Description metadata

EVENT_TYPE: Event type. If there is an Event or Sub-event element, the Event type element shall exist. The Event type element may occur only once in a node level of an Event tree or Sub-event branch.

DESCRIPTION: This element specifies a description of the event. This element is used to describe an event in text human readable format.

LOCATION: This element identifies the physical location of the Event and not the position within the image. See Location type (clause N.7.1.15) for more information on this element.

EVENT_TIME: Event date and time. This element specifies the start time of the event. See DateTime type (clause N.7.1.8) for the format of this element.

DURATION: This element specifies the duration of the Event.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the Event. See Comment element (clause N.7.3.1) for more information on this element.

PARTICIPANT: This element specifies the participants of the event. A participant may be a Person, an Organization or a Thing. The syntax of this element is specified in clause N.6.2.5.

EVENT_RELATION: Event relationships. This element specifies relationships to other events. The syntax of this element is specified in clause N.6.2.6.

Sub-events. The Event element may contain one or more Sub-event elements of the encompassing event. A Sub-event element may contain Sub-events. The sub-event element may be either contained within the event element, or referenced:

EVENT: Sub-event description.

EVENT_REF: Event reference. A reference to the sub-event. This element is a link to one of the other Event elements.

ID: This element specifies the unique identifier for the Event.

N.6.2.5 Participant metadata

This element specifies the participants in the event. A participant may be a Person, an Organization or a Thing. See also Figure N.25.

```
<xsd:element name="PARTICIPANT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ROLE" type="jp:tLangString" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:choice>
        <xsd:element name="OBJECT_REF" type="xsd:string"/>
        <xsd:element ref="jp:PERSON"/>
        <xsd:element ref="jp:THING"/>
        <xsd:element ref="jp:ORGANIZATION"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.25 – Schema of the Participant metadata

ROLE: This element specifies the role of the participant within the event.

OBJECT_REF: Object reference. This element is a reference to a participant. This element is a link to one of the Person, Organization or Thing elements within the Content Description metadata.

PERSON: This element specifies a Person who is a participant of an event yet not depicted within the image. See Person description metadata (see clause N.6.2.1) for the format of this element.

THING: This element specifies a Thing that is a participant of an event yet not depicted within the image. See Thing description metadata (see clause N.6.2.2) for the format of this element.

ORGANIZATION: This element specifies the Organization that is a participant of an event yet not depicted within the image. See Organization description metadata (see clause N.6.2.3) for the format of this element.

N.6.2.6 Event Relationship metadata

This element specifies relationships to other events. These are used for relationships between events that are not directly sub-events of each other. An example of a relationship might be a link to a previous event of the same type. See also Figure N.26.

```
<xsd:element name="EVENT_RELATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="RELATION" type="jp:tLangString" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="EVENT_REF" type="xsd:string"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.26 – Schema of the Event Relationship metadata

RELATION: This element specifies a description of the relationship(s) to the other event(s).

EVENT_REF: Event reference. This element is a reference to related events. This element is a link to one of the other Event elements within the Event description metadata.

N.6.2.7 Audio metadata

This element specifies audio metadata associated with an image. Image metadata can contain zero or more audio streams. Each audio stream can contain a comment element describing the audio. A single comment should also be able to describe more than one audio stream. See Figure N.27.

```
<xsd:element name="AUDIO">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="AUDIO_STREAM" type="xsd:anyURI"/>
      <xsd:element name="AUDIO_FORMAT" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="MIME_TYPE" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DESCRIPTION" type="jp:tLangString" minOccurs="0"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.27 – Schema of the Audio metadata

AUDIO_STREAM: This element specifies an URI reference to an audio stream. The format of the stream is not defined.

AUDIO_FORMAT: Audio Stream Format. This element specifies the name of the audio stream format. For example, AIFF, MIDI, MP3 and WAV.

MIME_TYPE: This element specifies the Internet media type of the audio file.

DESCRIPTION: This element specifies a description of the audio stream.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the Audio. See Comment element (clause N.7.3.1) for more information on this element.

N.6.2.8 Property metadata

This element specifies a description of an image or an object within an image. This element shall contain a name and may optionally contain a value and sub-property elements. A Property is either a single word or a small phrase and an optional value. The property is a non-exact language-specific definition of the image or part of the image. This element may contain the sub-elements listed below. See Figure N.28.

```
<xsd:element name="PROPERTY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NAME" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="VALUE" type="jp:tLangString" minOccurs="0"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
      <xsd:element ref="jp:PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="DICT_REF" type="xsd:string"/>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.28 – Schema of the Property metadata

NAME: This element specifies the name of the Property.

VALUE: This element specifies the property value. A Property that contains a value cannot contain sub-property elements.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the Property. See Comment element (clause N.7.3.1) for more information on this element.

PROPERTY: Sub-property. This element specifies sub-Properties of the encompassing Property. A property that contains sub-properties cannot contain a value.

DICT_REF: Dictionary reference. This element specifies a reference to a Dictionary (see clause N.6.2.9).

N.6.2.9 Dictionary Definition metadata

This element specifies the name of a dictionary. A Property may be defined using a specific dictionary. The advantage of this is that there is a single definition for each Property metadata, and that two different Property metadata annotations are not used to define the same thing. See Figure N.28.

To give an example, a dictionary may define the word "Vehicle" to be used to describe a car, vehicle, truck, automobile, etc. A second example is the use of the word "Date." Date may be used to specify the fruit of the palm "date" and not the definition of date as a day. This element may contain the sub-elements listed below.

```
<xsd:element name="DICTIONARY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="DICT_NAME" type="jp:tLangString" minOccurs="0"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="DICT_ID" type="xsd:string"/>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.29 – Schema of the Dictionary Definition metadata

DICT_NAME: Dictionary name. This element specifies the name of the dictionary.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the Dictionary. See Comment element (clause N.7.3.1) for more information on this element.

DICT_ID: Dictionary ID. This element specifies the unique identifier for the dictionary.

N.6.3 History metadata

The History element contains a summary of basic image editing operations that have already been applied to the image and previous version(s) of the image metadata. The History metadata is not designed to be used to reverse (undo) image editing operations. This element may contain the sub-elements listed below. See Figure N.30.

```
<xsd:element name="HISTORY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="jp:PROCESSING_SUMMARY" minOccurs="0"/>
      <xsd:element ref="jp:IMAGE_PROCESSING_HINTS" minOccurs="0"/>
      <xsd:element ref="jp:METADATA"/>
    <xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.30 – Schema of the History metadata

PROCESSING_SUMMARY: This element specifies a list of operations previously applied to an image during the course of its workflow. The syntax of this element is specified in clause N.6.3.1.

IMAGE_PROCESSING_HINTS: This element specifies a list of the operations previously performed when editing an image. The syntax of this element is specified in clause N.6.3.2.

METADATA: Previous metadata. This element specifies a previous version of the metadata that may include metadata about portions of an image that was deleted (e.g., cropped). The syntax of this element is specified in clause N.6.3.3.

N.6.3.1 Processing Summary metadata

This element specifies a list of the operations performed over the life of the image, listing the operations performed and not the ordering or the number of times each operation is performed.

The processing summary defined below should be considered potential and in all likelihood partial information. That is because the presence of a particular hint, such as "Image Cropped," indicates that the image has been cropped. However, absence of a "Image Cropped" hint is no assurance that the image has never been cropped. This element may contain the sub-elements listed below. See Figure N.31.

```
<xsd:element name="PROCESSING_SUMMARY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IMG_CREATED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_CROPPED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_TRANSFORMED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_GTC_ADJ" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_STC_ADJ" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_SPATIAL_ADJ" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_EXT_EDITED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_RETOUCHED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_COMPOSITED" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="IMG_METADATA" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.31 – Schema of the Processing Summary metadata

IMG_CREATED: Digital image created. The presence of this element indicates that the image was created by a metadata-aware application or process. Where a number of operations are performed in the creation of an image (such as removing borders), then these operations should be summarized using Digital Image Created operation and not listed independently. This element is especially useful to show truncation of image metadata. Where this element is not present, the full history of the metadata is known to be incomplete. Presence of this element does not show that the metadata history is complete though.

IMG_CROPPED: Image cropped. The presence of this element indicates that an image editing application, program, or system has cropped the image.

IMG_TRANSFORMED: Image Transformed. The presence of this element indicates that an image has been transformed.

IMG_GTC_ADJ: Global Tone/Colour Adjustment. The presence of this element indicates that a contrast or density adjustment has been applied to the image, or that the image colouring has been adjusted.

IMG_STC_ADJ: Selective Tone/Colour Adjustment. The presence of this element indicates that a contrast or density adjustment has been applied to a selected region of the image.

IMG_SPATIAL_ADJ: Global Spatial Adjustment. The presence of this element indicates that the image has been sharpened, or compressed, or blurred, or re-sampled.

IMG_EXT_EDITED: Pixels Extensively Edited. The presence of this element indicates the image has been edited extensively – enough to change the captured scene content.

IMG_RETOUCHED: Image Retouched. The presence of this element indicates the image pixels have been edited to remove scratches or red-eye, or other minor image blemishes.

IMG_COMPOSITED: Image Composited. The presence of this element indicates the image has been created by compositing an image with another image, or a background, graphic, or text.

IMG_METADATA: Metadata Adjusted. The presence of this element indicates the image metadata has been modified.

N.6.3.2 Image Processing Hints metadata

This element specifies a list of the operations performed when editing an image. They differ from the Processing Summary in that the hints list all the operations in order and the operations may be listed more than once (if the operation was used more than once). The Processing Summary metadata lists all the operations performed during the life of an image while the Image Processing Hints metadata stores the most current set of operations in greater detail. The complete list of operations (and their order) can be generated by combining all Image Processing Hints metadata within a Metadata History tree.

The Image Processing Hints element contains the same elements as the Processing Summary metadata. See Processing Summary (clause N.6.3.1) for the definition of each element. Each sub-element may appear more than once within each field and each element may contain a textual description of the operation. The Image Processing Hints metadata defined below should be considered potentially partial information. That is because the presence of a particular hint, such as "Image Cropped," indicates that the image has been cropped and other metadata may have been omitted at the same time. However, absence of an "Image Cropped" hint is no assurance that the image has never been cropped. See Figure N.32.

```
<xsd:element name="IMAGE_PROCESSING_HINTS">
  <xsd:complexType>
    <xsd:sequence>
    <xsd:element name="MODIFIER" type="jp:tProductDetails"minOccurs="0"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="IMG_CREATED" type="jp:tLangString"/>
        <xsd:element name="IMG_CROPPED" type="jp:tLangString"/>
        <xsd:element name="IMG_TRANSFORMED" type="jp:tLangString"/>
        <xsd:element name="IMG_GTC_ADJ" type="jp:tLangString"/>
        <xsd:element name="IMG_STC_ADJ" type="jp:tLangString"/>
        <xsd:element name="IMG_SPATIAL_ADJ" type="jp:tLangString"/>
        <xsd:element name="IMG_EXT_EDITED" type="jp:tLangString"/>
        <xsd:element name="IMG_RETOUCHED" type="jp:tLangString"/>
        <xsd:element name="IMG_COMPOSITED" type="jp:tLangString"/>
        <xsd:element name="IMG_METADATA" type="jp:tLangString"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.32 – Schema of the Image Processing Hints metadata

MODIFIER: This element specifies the application (most probably software), that performed the operations listed in Processing Summary (clause N.6.3.1). See Product Details type (clause N.7.1.21) for the format of this element.

N.6.3.3 Previous metadata

This element contains a previous version of the metadata (including previous History metadata). The format of this element is defined along with the History metadata (Figure N.30).

Each time a new image is created as a result of editing an image or combining several images, some of the metadata from the previous image(s) may be moved to or referenced by the image history metadata. The contributing image(s) Image Creation, Content Description, History and IPR metadata may be recorded in a Previous metadata element. Careful consideration shall be made with regards to this previous metadata, particularly previous IPR metadata. See Figure N.33.

```
<xsd:element name="METADATA">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="jp:BASIC_IMAGE_PARAM" minOccurs="0"/>
      <xsd:element ref="jp:IMAGE_CREATION" minOccurs="0"/>
      <xsd:element ref="jp:CONTENT_DESCRIPTION" minOccurs="0"/>
      <xsd:element ref="jp:HISTORY" minOccurs="0"/>
      <xsd:element ref="jp:IPR" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.33 – Schema of the Previous metadata

BASIC_IMAGE_PARAM: This element specifies references to previous versions of the image. The syntax of this element is specified in clause N.6.3.4.

IMAGE_CREATION: This element specifies the image creation information. The syntax of this element is specified in clause N.6.1.

CONTENT_DESCRIPTION: This element specifies the content description information. The syntax of this element is specified in clause N.6.2.

HISTORY: This element specifies previous history metadata. The syntax of this element is specified in clause N.6.3.

IPR: This element specifies image intellectual property. The syntax of this element is specified in clause N.6.4.

N.6.3.4 Image Referencing metadata

This element specifies information for referencing previous versions of the image. This element may contain the sub-elements listed below. See Figure N.34.

```
<xsd:element name="BASIC_IMAGE_PARAM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="BASIC_IMAGE_INFO" minOccurs="0"/>
        <xsd:complexType>
          </xsd:sequence>
            <xsd:element name="FILE_FORMAT " minOccurs="0"/>
              <xsd:complexType>
                </xsd:sequence>
                  <xsd:element      name="FILE_NAME"      type="xsd:anyURI"
minOccurs="0"/>
                  <xsd:element     name="FORMAT_TYPE"     type="xsd:string"
minOccurs="0"/>
                  <xsd:element      name="NIME_TYPE"      type="xsd:string"
minOccurs="0"/>
                  <xsd:element       name="VERSION"       type="xsd:string"
minOccurs="0"/>
                </xsd:sequence>
              <xsd:complexType>
            </xsd:element>
            <xsd:element ref="jp:IMAGE_ID" minOccurs="0"/
          </xsd:sequence>
          <xsd:attribute ref="jp:TIMESTAMP"/>
          <xsd:attribute ref="xml:lang"/>
        <xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  <xsd:complexType>
</xsd:element>
```

Figure N.34 – Schema of the Image Reference metadata

FILE_NAME: This field specifies the name of an image file.

FORMAT_TYPE: File Format Type. This field specifies the file format of the image.

MIME_TYPE: This field specifies the Internet media type of the image file.

VERSION: This field specifies the version of the file format.

IMAGE_ID: This element specifies the image identifier. The syntax of this element is specified in clause N.6.5.

N.6.4 Intellectual Property Rights metadata

This element specifies Intellectual Property Rights (IPR) related information associated with the image such as moral rights, copyrights as well as exploitation information.

Moral rights are those rights attached to the creation process; therefore, moral rights persistently pertain to the author or creator of the art work, whereas copyrights can be repeatedly transferred to different owners, under exploitation conditions which are also part of the IPR and exploitation metadata. Additional information such as conditions of use, names, content

description, dates, as well as IPR-related administrative tasks, identification (e.g., a unique inventory number) and contact point for exploitation are also considered important metadata.

Use and interpretation of this information is beyond the scope of this Recommendation | International Standard. Nothing in this Recommendation | International Standard should be taken to imply or to waive legal obligations or restrictions that may apply within any particular jurisdiction.

NOTE – Implementors should take into account the Berne Convention for the Protection of Literary and Artistic Works, World Intellectual Property Organization Copyright Treaty and other WIPO publications, if appropriate.

This element may contain the sub-elements listed below. See Figure N.35.

```
<xsd:element name="IPR">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="jp:IPR_NAMES" minOccurs="0"/>
      <xsd:element ref="jp:IPR_DESCRIPTION" minOccurs="0"/>
      <xsd:element ref="jp:IPR_DATES" minOccurs="0"/>
      <xsd:element ref="jp:IPR_EXPLOITATION" minOccurs="0"/>
      <xsd:element ref="jp:IPR_IDENTIFICATION" minOccurs="0"/>
      <xsd:element ref="jp:IPR_CONTACT_POINT" minOccurs="0"/>
      <xsd:element name="IPR_HISTORY" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="jp:IPR" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.35 – Schema of the Intellectual Property Rights metadata

IPR_NAMES: This element specifies names related to the represented image. The syntax of this element is specified in clause N.6.4.1.

IPR_DESCRIPTION: This element specifies the description of the content such as the title and caption. The syntax of this element is specified in clause N.6.4.2.

IPR_DATES: This element specifies the IPR-related date information. The syntax of this element is specified in clause N.6.4.3.

IPR_EXPLOITATION: This element specifies exploitation information such as type of protection, use restriction and obligations to exploit an image. The syntax of this element is specified in clause N.6.4.4.

IPR_IDENTIFICATION: This element specifies an identifier of an image that is a link to a place where additional information is kept. The syntax of this element is specified in clause N.6.4.6.

IPR_CONTACT_POINT: This element specifies the contact point of the right holder. The syntax of this element is specified in clause N.6.4.9.

IPR_HISTORY: This element contains previous IPR metadata. The content is specified in clause N.6.4.10.

N.6.4.1 IPR Names metadata

This element specifies names related to the represented image. These names include different categories, such as the creator, photographer, and producer, all who claim rights. People appearing within the image amy also be named, as there are restrictions on publishing the image of a person who has not consented to publication that varies from country to country. "Who," "what," and "where" (i.e., the subject of the image) can also be names in the title of the image.

A name may be either a Person, an Organization, or a reference to a name or a person. See Person type (clause N.7.1.13) and Organization type (clause N.7.1.14), respectively for the format of this element. This element may contain the sub-elements listed below. See Figure N.36.

```
<xsd:element name="IPR_NAMES">
  <xsd:complexType>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="jp:IPR_PERSON"/>
      <xsd:element ref="jp:IPR_ORG"/>
      <xsd:element ref="jp:IPR_NAME_REF"/>
    </xsd:choice>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="IPR_PERSON">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jp:tPerson">
        <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="IPR_ORG">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jp:tOrganization">
        <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="IPR_NAME_REF">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
</xsd:element>
```

Figure N.36 – Schema of the IPR Names metadata

IPR_PERSON: Person. This element specifies the person description. See Person type (clause N.7.1.13) for the format of this element.

IPR_ORG: Organization. This element specifies the organization description. See Organization type (clause N.7.1.14) for the format of this element.

IPR_NAME_REF: Name reference. This element specifies a reference to a person or organization within the IPR metadata.

DESCRIPTION: This element is the description of the name. Table N.14 lists suggested values for this element which have the following meaning.

Table N.14 – Name description values

Value	Meaning
Original Work Author	This value specifies that the element is the name of the author who created the original work that is represented in the image (e.g., painter sculptor, architect, etc.), when the image is not a creation itself. By contrast, a photograph of a sunset will be considered as a creation of the photographer. An original work author may be "anonymous."
Image Creator	This value specifies that the element is the name of the image creator. The image creator may be, for example, the photographer who captured the original picture on film, the illustrator or graphic artist who conducted the image-creation process, etc.
Right Holder	This value specifies that the element is the name of the intellectual property right holder of the image. The right holder may be the author of the image, a stock photo agency, or vendor. He is the one to sell the license to anyone willing to exploit the image, such as a publisher who will also sell the result or an end user in a pay-per-view process. The right holder has acquired the rights from the creator or previous right holder in a transaction which usually has been registered officially.
Represented Individuals	This value specifies that the element is the name of an individual shown in the image. This may be used as a description of the image or because privacy rights may require that individuals depicted grant consent to publish their image. In such an example, this descriptive element may result in restriction of use for the image, as well as describing the image contents.

N.6.4.2 IPR Description metadata

This element specifies the description of the content. It may be desirable to have a complementary explanation about the content of the image in order to exploit the content. For instance, a technical description of the content may help users in understanding and, therefore, valuing the content of an image (e.g., circumstances under which the image was taken). The format is vendor specific. This element may contain the sub-elements listed below. See Figure N.37.

```
<xsd:element name="IPR_DESCRIPTION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_TITLE" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_LEGEND" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_CAPTION" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="COPYRIGHT" type="jp:tLangString" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.37 – Schema of the IPR Description metadata

IPR_TITLE: Title of image. This element specifies the title of the image. It is a string that may be used, for instance, as a caption when printing. When the author creates the title, he may add meaning to the image. However, titles are not necessarily significant of IPR. This is determined on a case-by-case basis.

IPR_LEGEND: Legend. This element specifies the legend, a caption added to the picture, e.g., at the back of a photograph, written by the photographer to later classify the photos. It is generally a more detailed or technical description of what appears in the image. This element may answer the question, "why?" An example is saying, "image taken at dawn to test a 135 mm. zoom on stand."

IPR_CAPTION: Caption. This element specifies the caption of the image. This element addresses the text which has been added as complementary information to assist in understanding the image's content (e.g., second draft by Durer for a study on a Biblical scene). The caption often has a tutorial motivation.

COPYRIGHT: Copyright. This element specifies the copyright notice of the image. Usually this element defines the right holder who wants to be identified, saying e.g., "copyright agency XYZ." This is an indication that the property of the image is well defined and that the contact point is the designated agency.

N.6.4.3 IPR Dates metadata

This element specifies the IPR-related date information. There are a variety of valid DateTime formats. For example, a date may be an exact year, possibly with month and day, sometimes with hour, minute, second and thousandth (i.e., ISO timestamp, which is always GMT time). However, date may also be less delimiting. For example, the date may be "first half of the fifteenth century," "late middle-age," "early Roman," etc.

Professional applications may prefer an exact date, whereas specifying a year ± 5 years may satisfy users of early century photographs.

This element may contain the sub-elements listed below. See Figure N.38.

```
<xsd:element name="IPR_DATES">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_DATE" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="jp:tDateTime">
              <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.38 – Schema of the IPR Dates metadata

IPR_DATE: The date element contains a date of arbitrary precision. See DateTime type (clause N.7.1.8) for the format of this element. The comment element defined in the DataTime type may be used for describing more information on the element.

DESCRIPTION: This element is the description of the date. The precision of IPR Dates may vary in accuracy depending on the age of the operation or item and other information known at the time of the metadata generation. Table N.15 lists suggested values for this element which have the following meaning.

Table N.15 – Date description values

Value	Meaning
Original Work Creation	This value specifies that the element is the date that the original work was created. All types of dates may appear here, as stated above.
Picture Taken	This value specifies that the element is the date that the picture was taken. Some digital cameras insert this information automatically.
Scanned	This value specifies that the element is the date that the image was scanned.
Processed	This value specifies that the element is the date that the image was processed.
Modified	This value specifies that the element is the date when any kind of modification was made to the original work. This element will store the most recent modification date. Although it is valid to have more than one modification date in this section, it would be more common that the entire IPR is updated during the modification, and the previous modifications moved to the IPR history. The processing tool may generate this date automatically.
Last Modified	This value specifies the last date the image was modified. This date should be easily found, because there may be either an automatic process putting this element and replacing the previous "last modification" as a "history element" or a manual process where the operator has to do the same operation by hand.

N.6.4.4 IPR Exploitation metadata

This element specifies metadata to identify IPR protection mechanisms, specific restrictions imposed by the right holder or obligations resulting from the use of the image, and the IPR management system in use for this IPR metadata. This element may contain the sub-elements listed below. See Figure N.39.

```
  <xsd:element name="IPR_EXPLOITATION">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element       name="IPR_PROTECTION"        type="jp:tLangString"
minOccurs="0"/>
        <xsd:element     name="IPR_USE_RESTRICTION"     type="jp:tLangString"
minOccurs="0"/>
        <xsd:element       name="IPR_OBLIGATION"        type="jp:tLangString"
minOccurs="0"/>
        <xsd:element ref="jp:IPR_MGMT_SYS" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.39 – Schema of the IPR Exploitation metadata

IPR_PROTECTION: This element either indicates that there is a watermark, that the image is registered, or that the image is protected by some other means. A value of zero specifies that the image is not protected and contains no watermark. Values between 1 and 255 are reserved. Other values may exist. If this element is not present, then the watermark content (or its presence) is undefined.

IPR_USE_RESTRICTION: This element specifies the use restrictions of an image. Use restrictions may apply to an image that is not allowed outside the factory for industrial applications, or for which exclusive rights of copy have been delegated to a unique agency, or for which prior authorization of represented people is mandatory before publishing. Other restrictions may exist.

IPR_OBLIGATION: This element specifies the obligations of exploiting an image. Obligation may concern any mandatory condition for exploiting the content of a file. For example, the copyright information may be required to be written on the side of any printout for photographs; other obligations may concern the need to get allowance from persons represented on a picture if the picture is published. Obligations may vary with time. For example, it may be forbidden to publish a photograph before a given date, etc.

IPR_MGMT_SYS: IPR management system. This element specifies what management system is used. The syntax of this element is specified in clause N.6.4.5.

N.6.4.5 IPR Management System metadata

IPR Management Systems such as IPMP (Intellectual Property Management & Protection) or ECMS (Electronic Copyright Management System) use these elements to determine where information is kept regarding the management system. An example use of these elements is to track the usage of an image. During transfer, an agency determines the owner of the image from the management systems elements. It already knows the consumer, and uses this information to charge the user and credit the owner the amount as determined by the management system. This information is commonly stored on a server describing the IPR of the image, and depending upon whether IPR licensing is mandatory or recommended, there shall be a link to where all information about it is kept. This element may contain the sub-elements listed below. See Figure N.40.

```
  <xsd:element name="IPR_MGMT_SYS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="IPR_MGMT_TYPE" type="xsd:string" minOccurs="0"/>
        <xsd:element name="IPR_MGMT_SYS_ID" type="xsd:string" minOccurs="0"/>
        <xsd:element      name="IPR_MGMT_SYS_LOCATION"      type="xsd:anyURI"
minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute ref="jp:TIMESTAMP"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
  </xsd:element>
```

Figure N.40 – Schema of the IPR Management Systems metadata

IPR_MGMT_TYPE: The type of IPR Management System being used.

IPR_MGMT_ID: Information of an ID.

IPR_MGMT_LOCATION: Information of the location. E.g., URL.

N.6.4.6 IPR Identification metadata

This element specifies a link to a place (e.g., secured database or other storage place) where critical information is kept. The identifier identifies a content; therefore, if an image is cropped, modified or made into a new image, then the image shall be registered again, and a new identifier shall be acquired, because there are now two objects instead of merely one. However, the parent image shall appear in the metadata set of the child. This element may contain the sub-elements listed below. See Figure N.41.

```
<xsd:element name="IPR_IDENTIFICATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="jp:IPR_IDENTIFIER" minOccurs="0"/>
      <xsd:element ref="jp:LICENCE_PLATE" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.41 – Schema of the IPR Identification metadata

IPR_IDENTIFIER: Generic IPR identifier. This element contains a generic purpose IPR identifier. The syntax of this element is specified in clause N.6.4.7.

LICENCE_PLATE: This element specifies License plate of the content. The syntax of this element is specified in clause N.6.4.8.

N.6.4.7 Generic IPR Identifier metadata

This element specifies a generic IPR identifier. This element may contain the sub-elements listed below. See Figure N.42.

```
<xsd:element name="IPR_IDENTIFIER">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_ID_MODE" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_ID" type="jp:tLangString" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.42 – Schema of the IPR Identifier metadata

IPR_ID_MODE: This element specifies the identification mode.

IPR_ID: This element specifies the identification. The Mode element describes the content of this element.

N.6.4.8 License Plate metadata

This element specifies the license plate of the original image, defined in ISO/IEC 10918-3. The combination of the elements in the license plate contains a globally unique identifying sequence of numbers. This element may contain the sub-elements listed below. See Figure N.43.

```
<xsd:element name="LICENCE_PLATE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="LP_COUNTRY" type="xsd:string" minOccurs="0"/>
      <xsd:element name="LP_REG_AUT" type="xsd:string" minOccurs="0"/>
      <xsd:element name="LP_REG_NUM" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DELIVERY_DATE" type="xsd:dateTime" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.43 – Schema of the License Plate metadata

LP_COUNTRY: This element specifies the country of registration. The element contains the country code (3-digit number) for the license plate as defined in ISO 3166-1.

LP_REG_AUT: This element specifies the registration authority number for the license plate.

LP_REG_NUM: This element specifies the registration number for the license plate.

LP_DELIVERY_DATE: This element specifies when the license plate was delivered to the registrant by the registration authority.

NOTE – A previous edition of this Recommendation | International Standard referenced the 1997 edition of ISO 3166-1. This Recommendation | International Standard now makes an undated reference to ISO 3166-1.

N.6.4.9 IPR Contact Point metadata

This element specifies the contact point of the right holder. It includes a way to contact the current right holder in order to acquire the rights under the form of a licence. Such information may be a postal address, URL or any phone or fax number that is a non-ambiguous link to the right holder.

A contact point may be either a Person, an Organization, or a reference to a name or a person. This element may contain the sub-elements listed below. See Figure N.44.

```
<xsd:element name="IPR_CONTACT_POINT">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="jp:IPR_PERSON"/>
      <xsd:element ref="jp:IPR_ORG"/>
      <xsd:element ref="jp:IPR_NAME_REF"/>
    </xsd:choice>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.44 – Schema of the IPR Contact Point metadata

IPR_PERSON: This element specifies the person description. The syntax of this element is specified in clause N.6.4.1.

IPR_ORG: Organization. This element specifies the organization description. The syntax of this element is specified in IPR Names (see clause N.6.4.1).

IPR_NAME_REF: Name Reference. This element specifies a reference to a person or organization within the IPR metadata. This element is a link to one of the Person or Organization elements within the IPR Names metadata (see clause N.6.4.1).

DESCRIPTION: This element is the description of the contact point that is an additional value for the person or organization in Table N.14. The value listed in Table N.16 is added and has the following meaning.

Table N.16 – Additional name description values

Value	Meaning
Collection	This value is a link to a collector, museum, group, institution, etc. The contact point may be a link to a name specified in IPR Names.

N.6.4.10 IPR History metadata

This element specifies previous IPR metadata. The format of this element is defined along with the Intellectual Property Rights metadata (Figure N.35).

Each time the IPR information of an image is changed, some of the IPR metadata defined through clause N.6.4.1 and clause N.6.4.9 may be moved to this IPR History metadata element. The IPR History metadata stores all IPR metadata-related modifications.

N.6.5 Image Identifier metadata

This element specifies an image identifier that uniquely identifies the image. The format may be globally unique (e.g., UUID), vendor or application dependent. This element may contain the sub-elements listed below. See Figure N.45.

```
<xsd:element                                                    name="IMAGE_ID">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element     name="UID"      type="xsd:string"      minOccurs="0"/>
      <xsd:element     name="ID_TYPE"   type="xsd:anyURI"     minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure N.45 – Schema of the Image Identifier metadata

UID: Unique Identifier. This element specifies the unique identifier of an image. The ID_TYPE element specifies the format of the field.

ID_TYPE: Unique Identifier Type. This element specifies the type of the UID element as a URI.

N.7 Fundamental type and element definitions

XML Schema Part 2 defines many built-in and derived datatypes; however, they are not sufficient to specify various metadata elements defined in this Recommendation | International Standard. This clause defines the common types and elements that are referenced within other metadata boxes. The types and elements defined are intended only to be used or referred to in other schemas, and have no intrinsic significance.

N.7.1 Defined types

N.7.1.1 Non-negative double type

This type is used for double numbers greater than or equal to zero. See Figure N.46.

```
<xsd:simpleType name="tNonNegativeDouble">
  <xsd:restriction base="xsd:double">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure N.46 – Schema of the non-negative double type

N.7.1.2 Rational type

This type is used to define rational numbers. It contains an enumerator and denominator in a single string. See Figure N.47.

```
<xsd:simpleType name="tRational">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(\-|\+)?[0-9]+/[0-9]+"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure N.47 – Schema of the rational type

N.7.1.3 String including language attribute type

This type is used to when an element requires a string and a language attribute definition. The content of this element is intended to store human readable data. See Figure N.48.

```
<xsd:complexType name="tLangString">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute ref="xml:lang"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Figure N.48 – Schema of the string including language attribute type

N.7.1.4 Degree type

This type specifies a direction in degrees and fractions of degrees. The exact meaning of the values is dependent on usage. See Figure N.49.

```
<xsd:simpleType name="tDegree">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="-180"/>
    <xsd:maxInclusive value="180"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure N.49 – Schema of the degree type

N.7.1.5 Half degree type

This type specifies a direction in degrees and fractions of degrees. The exact meaning of the values is dependent on usage. This type defines a smaller range than Degree Type (see clause N.7.1.4). See Figure N.50.

```
<xsd:simpleType name="tHalfDegree">
 <xsd:restriction base="xsd:double">
  <xsd:minExclusive value="-90"/>
  <xsd:maxInclusive value="90"/>
 </xsd:restriction>
</xsd:simpleType>
```

Figure N.50 – Schema of the half degree type

N.7.1.6 Double size type

This type specifies a size in double coordinates. See Figure N.51.

```
<xsd:complexType name="tDoubleSize">
  <xsd:sequence>
    <xsd:element name="WIDTH" type="jp:tNonNegativeDouble"/>
    <xsd:element name="HEIGHT" type="jp:tNonNegativeDouble"/>
  </xsd:sequence>
 </xsd:complexType>
```

Figure N.51 – Schema of the double size type

N.7.1.7 Integer size type

This type specifies a size in integer coordinates (e.g., pixels). See Figure N.52.

```
<xsd:complexType name="tIntSize">
  <xsd:sequence>
    <xsd:element name="WIDTH" type="xsd:positiveInteger"/>
    <xsd:element name="HEIGHT" type="xsd:positiveInteger"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure N.52 – Schema of the integer size type

N.7.1.8 DateTime type

This type specifies a partial or exact date. A date can include either a specific day (e.g., 26 January 2000), or a broader definition such as "Winter." A date may or may not include a time. This type may contain the sub-elements listed below. See Figure N.53.

```
<xsd:complexType name="tDateTime">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element name="EXACT" type="xsd:dateTime"/>
      <xsd:element name="DATE" type="xsd:date"/>
      <xsd:sequence>
        <xsd:element name="MONTH" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:positiveInteger">
              <xsd:minInclusive value="1"/>
              <xsd:maxInclusive value="12"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="YEAR" type="xsd:gYear" minOccurs="0"/>
        <xsd:element name="CENTURY" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer"/>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:choice>
    <xsd:element name="WEEK_DAY" type="xsd:string" minOccurs="0"/>
    <xsd:element name="SEASON" type="xsd:string" minOccurs="0"/>
    <xsd:element ref="jp:COMMENT" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.53 – Schema of the DateTime type

EXACT: This element contains an exact date and a time.

DATE: This element contains a date (excluding the time of day).

MONTH: This element contains a month of the year. An integer value is used rather than a string to be consistent with the other elements contained in the DateTime type. The value for January shall correspond to 1 and December to 12.

YEAR: This element contains a calendar year. Positive values used for AD and negative values for BC. The year zero is not valid.

CENTURY: This element contains the century that an event occurred. For example, the twentieth century is stored as "19." The century zero is not valid.

WEEK_DAY: This element is a text description of the day. Examples include "Monday" and "Friday."

SEASON: This element is a text description of a season. Examples include, "Spring," "Summer," "Autumn," and "Winter."

COMMENT: See Comment element (clause N.7.3.1) for more information on this element. Examples include "Easter Sunday," "Morning," "Just after lunch."

N.7.1.9 Address type

This type specifies the address of an object or location. For example, it may be used to describe the address an image was captured, or the address of the intellectual property owner of an image. This type may contain the sub-elements listed below. See Figure N.54.

```
<xsd:complexType name="tAddress">
  <xsd:sequence>
    <xsd:element name="ADDR_NAME" type="jp:tLangString" minOccurs="0"/>
    <xsd:element name="ADDR_COMP" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="jp:tLangString">
            <xsd:attribute name="TYPE" type="xsd:string"/>
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:choice minOccurs="0">
      <xsd:element name="ZIPCODE" type="xsd:string"/>
      <xsd:element name="POSTCODE" type="xsd:string"/>
    </xsd:choice>
    <xsd:element name="COUNTRY" type="jp:tLangString" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="TYPE" type="xsd:string"/>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.54 – Schema of the Address type

ADDR_NAME: Address name. It is a descriptive element for the address.

ADDR_COMP: Address component. Multiple elements are used to specify the complete address. The order of the address elements specifies the full address. A full address shall be generated by concatenating the separate address elements. For example, if the type is a "state", this element contains the name of the state. Where the type is a "street," this element contains the name of the street. ISO 3166-2 lists country subdivision codes. These codes may be used in this element, when the element is being used to specify a country subdivision.

TYPE: This is the name of this part of the address. Examples include "street" or "state." ISO 3166-2 specifies country subdivisions and the types of these divisions. These subdivision types may be used to specify the address type. Suggested values and their corresponding meanings are listed in Table N.17. Multiple values shall not be specified within a single element.

NOTE 1 – A previous edition of this Recommendation | International Standard referenced the 1998 edition of ISO 3166-2. This Recommendation | International Standard now makes an undated reference to ISO 3166-2.

Table N.17 – Address component type values

Value	Meaning
Unit	The unit number of the address to identify a house or a house name relative to a street.
Room	The room number within a building or an apartment.
Street	The street address in a postal address. Examples are street name, avenue and house number.
Postbox	The post office box number
City	The locality of a.geographic area.
State	The name of a geographical subdivision. Other terms such as "Province", "Prefecture", "County" may be used instead.

ZIPCODE/POSTCODE: This element specifies the postcode (or zip code) of the address. This element is not limited in length. The element has the title "Postcode" or "Zip code." An address cannot contain both a postcode and a zip code.

COUNTRY: This element specifies the country of the address. The element can either contain the country code as defined in ISO 3166-1 or a string identifying the country. The ISO 3166-1 country code is preferred.

TYPE: This element specifies the type of the whole address. The address type would include whether the address is a home address or a business address. Suggested type values are listed in Table N.18. Multiple type values may be specified delimited with a comma (",").

NOTE 2 – A previous edition of this Recommendation | International Standard referenced the 1997 edition of ISO 3166-1. This Recommendation | International Standard now makes an undated reference to ISO 3166-1.

Table N.18 – Address type values

Value	Meaning
Domestic	The domestic delivery address.
International	The international delivery address.
Postal	The postal delivery address.
Home	The delivery address for a residence.
Work	The delivery address for a place of work.

N.7.1.10 Phone number type

This type specifies a phone number. This type may contain the sub-elements listed below. See Figure N.55.

```
<xsd:complexType name="tPhone">
  <xsd:sequence>
    <xsd:element name="COUNTRY_CODE" type="xsd:string" minOccurs="0"/>
    <xsd:element name="AREA" type="xsd:string" minOccurs="0"/>
    <xsd:element name="LOCAL" type="xsd:string" minOccurs="0"/>
    <xsd:element name="EXTENSION" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="TYPE" type="xsd:string"/>
  <xsd:attribute ref="jp:TIMESTAMP"/>
</xsd:complexType>
```

Figure N.55 – Schema of the Phone number type

COUNTRY_CODE: This element contains the country code part of a phone number. This phone code does not include any prefix such as "00" used to dial international numbers, but instead just the international country code. This element also does not include a leading "+."

AREA: This element contains the local area code part of a phone number. This area code does not include leading zeros (or other digits) used to dial an interstate number from within a country. It appears as it would be appended directly to a country code.

LOCAL: This element contains the local phone number.

EXTENSION: This element contains the extension part of the phone number.

TYPE: This element defines the type of the phone number. The phone number type would include whether the phone number is a home phone number or a business phone number. Suggested type values are listed in Table N.19. Multiple type values may be specified delimited with a comma (",").

Table N.19 – Phone number type values

Value	Meaning
Home	Phone number associated with a residence.
Message	Phone number that has voice message support.
Work	Phone number associated with a place of work.
Voice	Phone number indicating a voice telephone.
Cell	Cellular telephone number.
Video	Video conference telephone number.
BBS	Bulletin board system telephone number.
Modem	A modem connected telephone number.
Car	A car-phone telephone number.
ISDN	ISDN service telephone number.
PCS	Personal communication service telephone number.

N.7.1.11 Email address type

This type specifies an email address. This type may contain the sub-elements listed below. See Figure N.56.

```
<xsd:complexType name="tEmail">
  <xsd:simpleContent>
    <xsd:extension base="jp:tLangString">
      <xsd:attribute name="TYPE" type="xsd:string"/>
      <xsd:attribute ref="jp:TIMESTAMP"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Figure N.56 – Schema of the Email address type

TYPE: This element contains the type of the email address.

N.7.1.12 Web address type

This type specifies a web page address. This type may contain the sub-elements listed below. See Figure N.57.

```
<xsd:complexType name="tWeb">
  <xsd:simpleContent>
    <xsd:extension base="jp:tLangString">
      <xsd:attribute name="TYPE" type="xsd:string"/>
      <xsd:attribute ref="jp:TIMESTAMP"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Figure N.57 – Schema of the Web address type

TYPE: This element contains the type of the web page.

N.7.1.13 Person type

This type specifies a person. The sub-elements are compatible with the vCard description defined in RFC 2426. This type may contain the sub-elements listed below. See Figure N.58.

```
  <xsd:complexType name="tPerson">
    <xsd:sequence>
      <xsd:element name="NAME_TITLE" type="jp:tLangString" minOccurs="0"/>
      <xsd:element name="PERSON_NAME" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="NAME_COMP" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                    <xsd:attribute name="TYPE" use="optional" default="Given">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:enumeration value="Prefix"/>
                          <xsd:enumeration value="Given"/>
                          <xsd:enumeration value="Family"/>
                          <xsd:enumeration value="Suffix"/>
                          <xsd:enumeration value="Maiden"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:extension>
                </xsd:simpleContent>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute ref="jp:TIMESTAMP"/>
          <xsd:attribute ref="xml:lang"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="NICK_NAME" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="JOB_TITLE" type="xsd:string" minOccurs="0"/>
      <xsd:choice minOccurs="0">
        <xsd:element name="PERSON_ORG" type="jp:tOrganization"/>
        <xsd:element name="ORG_REF" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ADDRESS" type="jp:tAddress" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="PHONE" type="jp:tPhone" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="EMAIL" type="jp:tEmail" minOccurs="0"
maxOccurs="unbounded"/>
```

```
      <xsd:element       name="WEB"       type="jp:tWeb"       minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="BIRTH_DATE" type="xsd:date" minOccurs="0"/>
      <xsd:element name="AGE" type="xsd:duration" minOccurs="0"/>
      <xsd:element ref="jp:COMMENT" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:string"/>
    <xsd:attribute ref="jp:TIMESTAMP"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
```

Figure N.58 – Schema of the Person type

NAME_TITLE: The element contains the person's title.

PERSON_NAME: This element specifies a framework to describe a person's name. A person's name is composed of multiple name components (e.g., given name(s) and family name(s)). The order of the name component elements specifies the full name of the person. For example, in languages where the family name is usually placed before the given name, then they would appear in this order in the file.

NAME_COMP: Name component. This element contains a single portion (word) of the name of a person. A name component element may contain a single initial rather than a complete word. To specify the full name of a person, multiple name component elements are used. This element contains a type as specified below.

TYPE: Name component type. This element defines the type of the Name Component element. This element would include whether the name component is a Suffix, Prefix, Given or Family name. Suggested values and their corresponding meanings are listed in Table N.20. Multiple values shall not be specified within a single type filed.

Table N.20 – Name component type values

Value	Meaning
Prefix	A personal title. (e.g., Dr., Sir)
Given	A name construct that is normally given to an individual by the parent or is chosen by the individual. This is the default value of the name component type.
Family	A name component that is normally inherited by their parent or assumed by marriage.
Suffix	A generation qualifier (e.g., Jr., III), decorations and awards. (e.g., Q.C., Ph. D)
Maiden	A name component of a woman's family name before getting married.

NICK_NAME: This element specifies a nick name of the person. E.g., "Jimmy."

JOB_TITLE: This element specifies the person's job title.

ORGANIZATION: This element specifies the organization for which a person is a member of. The organization element may be either contained within the person element, or referenced.

ORG_REF: Organization reference. A reference to the organization. This element is a link to one of the Organization elements within the metadata.

ADDRESS: This element specifies address information for the person. For example, it can contain a home address or a work address. It does not necessarily contain the address depicted within the image, but instead information about the person. See Address type (clause N.7.1.9) for the format of this element.

PHONE: Phone number. This element specifies phone number information for the person. See Phone number type (clause N.7.1.10) for the format of this element.

EMAIL: Email address. This element specifies an email address for a person. See Email address type (clause N.7.1.11) for the format of this element.

WEB: Web page. This element contains a web page for a person. See Web address type (clause N.7.1.12) for the format of this element.

BIRTH_DATE: Date of birth. This element specifies the birth date of the person. This element shall specify an exact date. For non-specific information the Comment element shall be used.

AGE: This element contains the age of a person.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the person type. See Comment element (clause N.7.3.1) for more information on this element.

ID: This element specifies the unique identifier for the person.

N.7.1.14 Organization type

This type specifies an organization. The sub-elements are compatible with the vCard description defined in RFC 2426. This type may contain the sub-elements listed below. See Figure N.59.

```
<xsd:complexType name="tOrganization">
  <xsd:sequence>
    <xsd:element name="ORG_NAME" type="jp:tLangString" minOccurs="0"/>
    <xsd:element name="ADDRESS" type="jp:tAddress" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="PHONE" type="jp:tPhone" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="EMAIL" type="jp:tEmail" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="WEB" type="jp:tWeb" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="LOGO_FILE" type="xsd:anyURI" minOccurs="0"/>
    <xsd:element name="LOGO_FORMAT" type="xsd:string" minOccurs="0"/>
    <xsd:element name="MIME_TYPE" type="xsd:string" minOccurs="0"/>
    <xsd:element ref="jp:COMMENT" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:string"/>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.59 – Schema of the Organization type

ORG_NAME: Organization name. This element specifies the name of the organization.

ADDRESS: This element specifies address information for the organization. It does not necessarily contain the address depicted within the image, but instead information about the organization. See Address type (clause N.7.1.9) for the format of this element.

PHONE: Phone number. This element specifies phone number information. See Phone number type (clause N.7.1.10) for the format of this element.

EMAIL: Email address. This element specifies an email address for an Organization. See Email address type (clause N.7.1.11) for the format of this element.

WEB: Web page. This element specifies a web page for an Organization. See Web address type (clause N.7.1.12) for the format of this element.

LOGO_FILE: This element specifies a reference to a logo file of the organization.

LOGO_FILE_FORMAT: This element specifies the name of the logo file format. For example, EPS, JP2 and TIFF.

MIME_TYPE: This element specifies the Internet media type of the logo file.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the organization type. See Comment element (clause N.7.3.1) for more information on this element.

ID: This element specifies the unique identifier for the organization.

N.7.1.15 Location type

This type specifies the physical location of an object or a scene. For example, it may be used to describe an object within an image, or the location of a camera at the time of capture. The Location is the physical location, whereas the Position is the position of an object relative to the image. See Figure N.60.

```
<xsd:complexType name="tLocation">
  <xsd:sequence>
    <xsd:element ref="jp:COORD_LOC" minOccurs="0"/>
    <xsd:element name="ADDRESS" type="jp:tAddress" minOccurs="0"/>
    <xsd:element ref="jp:GPS" minOccurs="0"/>
    <xsd:element ref="jp:COMMENT" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.60 – Schema of the Location type

COORD_LOC: Coordinate location. This element specifies the exact longitude, latitude and altitude of an object. The syntax of this element is specified in clause N.7.1.15.1.

ADDRESS: This element specifies the location of an object using an address. See Address type (clause N.7.1.9) for the format of this element.

GPS: Global Positioning System. This element specifies location information received from a GPS receiver. The syntax of this element is specified in clause N.7.1.15.2.

COMMENT: This element specifies the location of an object that cannot be described using the other location elements. For example, "Under the table." See Comment element (clause N.7.3.1) for the format of this element.

N.7.1.15.1 Coordinate location

This element specifies the terrestrial location (altitude/longitude/latitude) of an object. It may be used to describe the content of an image along with the location of a camera.

While the coordinate location may have come from a GPS (and a GPS block may or may not be present in the metadata), the values in the coordinate location may have come for some other means. For this reason, the location information is a more general system for storing the location than the GPS system. The location information and the raw GPS data are stored in different formats.

GPS is one of a number of methods that may be used to determine a location. If the GPS information is filled in, it is expected that the coordinate location is also specified. A reader shall only look in a single place to determine the coordinate location (this element).

The meridian through Greenwich (Great Britain) is defined with the value longitude $l = 0$. The longitude l of a point P on the surface is the angle between the planes through its meridian and the Greenwich meridian. The longitude is counted from Greenwich up to $l = \pm 180°$ in east(+) and west(–) directions.

The latitude j of a point P is the angle between a line normal to its parallel and the equatorial plane ($j = 0$). On a sphere this normal line will be the connecting line between its center and the point P. On the elliptical earth this line will only pass the center if P is situated at the equator. The latitude is counted from the equator up to $j = \pm 90°$ in north (+) and south (–) directions. See Figure N.61.

```
<xsd:element name="COORD_LOC">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="LONGITUDE" type="jp:tDegree" minOccurs="0"/>
      <xsd:element name="LATITUDE" type="jp:tHalfDegree" minOccurs="0"/>
      <xsd:element name="ALTITUDE" type="xsd:double" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jp:TIMESTAMP"/>
  </xsd:complexType>
</xsd:element>
```

Figure N.61 – Schema of the Coordinate location element

LONGITUDE: This element specifies the longitude, represented in double degrees and fractions of degrees. E.g., "138,700," "–122,450."

LATITUDE: This element specifies the latitude, represented in double degrees and fractions of degrees. E.g., "35,383," "37,767."

ALTITUDE: This element would contain the distance in meters. Zero is sea level, positive is above, and negative is below.

N.7.1.15.2 Raw GPS Information

The information in these elements is expected to be imported from a GPS system and is compatible with NMEA-0138. For this reason, the elements are not consistent with other metadata elements. For example, a distance on the GPS elements may be stored in miles, while all other metadata distances are stored in meters. These elements are compatible with Exif version 2.1.

If information for latitude, longitude and altitude are present in the raw GPS information, the matching elements in the Coordinate location shall be filled in.

This element may contain the sub-elements listed below. See Figures N.62 to N.64.

```
  <xsd:element name="GPS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="GPS_LAT_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="N"/>
              <xsd:enumeration value="S"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="GPS_LATITUDE" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="D" type="xsd:nonNegativeInteger"/>
              <xsd:element name="M" type="xsd:nonNegativeInteger"/>
              <xsd:element        name="S"        type="jp:tNonNegativeDouble"
minOccurs="0"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="GPS_LONG_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="E"/>
              <xsd:enumeration value="W"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="GPS_LONGITUDE" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="D" type="xsd:nonNegativeInteger"/>
              <xsd:element name="M" type="xsd:nonNegativeInteger"/>
              <xsd:element        name="S"        type="jp:tNonNegativeDouble"
minOccurs="0"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element     name="GPS_ALTITUDE"     type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="GPS_TIME" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="GPS_SATELLITES" type="xsd:string" minOccurs="0"/>
```

```
        <xsd:element name="GPS_STATUS" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="A"/>
              <xsd:enumeration value="V"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="GPS_MEASURE_MODE" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:positiveInteger">
              <xsd:minExclusive value="2"/>
              <xsd:maxInclusive value="3"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element        name="GPS_DOP"        type="jp:tNonNegativeDouble"
minOccurs="0"/>
```

Figure N.62 – Schema of the Raw GPS Information element

```
        <xsd:element name="GPS_SPEED_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="K"/>
              <xsd:enumeration value="N"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element       name="GPS_SPEED"       type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="GPS_TRACK_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="T"/>
              <xsd:enumeration value="M"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element       name="GPS_TRACK"       type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="GPS_IMAGE_DIR_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="T"/>
              <xsd:enumeration value="M"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element     name="GPS_IMAGE_DIR"     type="jp:tNonNegativeDouble"
minOccurs="0"/>
        <xsd:element name="GPS_MAP_DATUM" type="xsd:string" minOccurs="0"/>
        <xsd:element name="GPS_DEST_LAT_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="N"/>
              <xsd:enumeration value="S"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="GPS_DEST_LATITUDE" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="D" type="xsd:nonNegativeInteger"/>
              <xsd:element name="M" type="xsd:nonNegativeInteger"/>
```

```
              <xsd:element        name="S"        type="jp:tNonNegativeDouble"
minOccurs="0"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="GPS_DEST_LONG_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="E"/>
              <xsd:enumeration value="W"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="GPS_DEST_LONGITUDE" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="D" type="xsd:nonNegativeInteger"/>
              <xsd:element name="M" type="xsd:nonNegativeInteger"/>
              <xsd:element        name="S"        type="jp:tNonNegativeDouble"
minOccurs="0"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="GPS_DEST_BEARING_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="T"/>
              <xsd:enumeration value="M"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element    name="GPS_DEST_BEARING"    type="jp:tNonNegativeDouble"
minOccurs="0"/>
```

Figure N.63 – Schema of the Raw GPS Information element *(continued)*

```
        <xsd:element name="GPS_DEST_DISTANCE_REF" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="K"/>
              <xsd:enumeration value="N"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element   name="GPS_DEST_DISTANCE"   type="jp:tNonNegativeDouble"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Figure N.64 – Schema of the Raw GPS Information element *(concluded)*

GPS_LAT_REF: GPS Latitude Reference. This element specifies whether the GPS Latitude is North or South. Table N.21 lists legal values of this element.

Table N.21 – Latitude reference values

Value	Meaning
N	North Latitude
S	South Latitude

GPS_LATITUDE: GPS Latitude. This element contains the latitude of the GPS receiver. Table N.22 lists legal values of this element.

Table N.22 – Latitude values

Value	Meaning
D	The number of degrees of latitude.
M	The number of minutes of latitude.
S	The number of seconds of latitude.

GPS_LONG_REF: GPS Longitude Reference. This element specifies whether the GPS Longitude is East or West. Table N.23 lists legal values of this element.

Table N.23 – Longitude reference values

Value	Meaning
E	East longitude
W	West longitude

GPS_LONGITUDE: GPS Longitude. This element contains the longitude of the GPS receiver. Table N.24 lists legal values of this element.

Table N.24 – Longitude values

Value	Meaning
D	The number of degrees of longitude.
M	The number of minutes of longitude.
S	The number of seconds of longitude.

GPS_ALTITUDE: GPS Altitude. This element contains the altitude of the GPS receiver. The altitude reading is given in meters relative to sea level (geoid).

GPS_TIME: GPS Time. This element contains the time of the GPS location was determined. This element is in Greenwich Mean Time. This is not necessarily the camera capture time.

GPS_SATELLITES: GPS Satellites. This element contains information about the satellites used to determine the camera position. This element can be used to describe the number of satellites, their ID number, angle of elevation, azimuth, SNR and other information. The format is not specified.

GPS_STATUS: GPS Status. This element contains information on the GPS receiver at time of image capture. Table N.25 lists legal values of this element.

Table N.25 – GPS Status values

Value	Meaning
A	Measurement in progress.
V	Measurement is interrupted.

GPS_MEASURE_MODE: GPS Measure Mode. This element contains information on the measurement mode used to determine the GPS location. Table N.26 lists legal values of this element.

Table N.26 – GPS Measure mode values

Value	Meaning
2	2 dimensional measurement.
3	3 dimensional measurement.

GPS_DOP: GPS Data Degree of Precision (DOP). This element contains a value indicating the GPS DOP. An HDOP (horizontal degree of precision) value is written during a two-dimensional measurement, and a PDOP (3D degree of precision) value is written during a three-dimensional measurement.

GPS_SPEED_REF: GPS Speed Reference. This element contains the units of measure for the GPS Speed element. Table N.27 lists legal values of this element.

Table N.27 – GPS Speed reference unit values

Value	Meaning
K	Kilometres per hour
N	Knots

GPS_SPEED: GPS Speed. This element contains a value indicating the speed of the GPS receiver. The value units are defined by the GPS Speed Reference.

GPS_TRACK_REF: GPS Track Reference. This element contains the reference for the GPS Track element. Table N.28 lists legal values of this element.

Table N.28 – Direction reference values

Value	Meaning
T	True north
M	Magnetic north

GPS_TRACK: GPS Track. This element contains the value in degrees indicating the direction of the GPS receiver movement. 0 indicates North and 90 indicate East.

GPS_IMAGE_DIR_REF: GPS Image Direction Reference. This element contains the reference for the GPS Image Direction element. Table N.28 lists legal values of this element.

GPS_IMAGE_DIR: GPS Image Direction. This element contains the value in degrees indicating the direction the camera is facing at the time of taking the picture. 0 indicates North and 90 indicate East.

GPS_MAP_DATUM: GPS Map Datum. This element specifies the geodetic survey data used by the GPS receiver. For example, if the survey data is restricted to Japan, the value of this tag is "TOKYO" or "WSG-84."

GPS_DEST_LAT_REF: GPS Destination Latitude Reference. This element specifies whether the GPS Destination Latitude is North or South. Table N.21 lists legal values of this element.

GPS_DEST_LATITUDE: GPS Destination Latitude. This element contains the destination latitude of the GPS receiver. Table N.22 lists legal values of this element.

GPS_DEST_LONG_REF: GPS Destination Longitude Reference. This element specifies whether the GPS Destination Longitude is East or West. Table N.23 lists legal values of this element.

GPS_DEST_LONGITUDE: GPS Destination Longitude. This element contains the destination longitude of the GPS receiver. Table N.24 lists legal values of this element.

GPS_DEST_BEARING_REF: GPS Destination Bearing Reference. This element contains the reference for the GPS Destination Bearing element. Table N.28 lists legal values of this element.

GPS_DEST_BEARING: GPS Destination Bearing. This element contains the value in degrees indicating the direction of the destination from the GPS receiver. 0 indicates North and 90 indicate East.

GPS_DEST_DISTANCE_REF: GPS Destination Distance Reference. This element contains the units of measure for the GPS Destination Distance element. Table N.29 lists legal values of this element.

Table N.29 – GPS Destination distance reference unit values

Value	Meaning
K	Kilometres per hour
N	Knots

GPS_DEST_DISTANCE: GPS Destination Distance. This element contains a value indicating the distance to the destination from the GPS receiver. The value units are defined by the GPS Destination Distance Reference.

N.7.1.16 Direction type

This type specifies a three-dimensional heading. While this type is primarily used to specify the direction a camera is facing, it may also be used to specify information about an object in a scientific photograph for example. When calculating the direction the camera is facing, first the yaw is applied, then the pitch, then the roll. This type may contain the sub-elements listed below. See Figure N.65.

```
<xsd:complexType name="tDirection">
  <xsd:sequence>
    <xsd:element name="YAW" type="jp:tDegree" minOccurs="0"/>
    <xsd:element name="PITCH" type="jp:tHalfDegree" minOccurs="0"/>
    <xsd:element name="ROLL" type="jp:tDegree" minOccurs="0"/>
    <xsd:element ref="jp:COMMENT" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.65 – Schema of the Direction type

YAW: This element is the direction the capture device is facing. The element is measured in degrees. North is 0, East is 90, South 180 and West is –90.

PITCH: This element is a measure of the elevation angle of the capture device. This element is a Double value between –90 and +90, also measured in degrees. 0 facing horizontal. 90 is facing vertically straight upwards, and –90 vertically downwards.

ROLL: This element is a measure of the rotation angle of the capture device. This element is a Double value between –180 and 180, also measured in degrees. 0 facing horizontal. 90 where the device is rotated clockwise and the left of the device is facing upwards, and –90 where the device is rotated anti-clockwise. 180 is upside down.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the direction types. For example, "Upwards," "To the left." See Comment element (clause N.7.3.1) for more information on this element.

N.7.1.17 Position type

This type is used to specify the position of an object, within an image. The Position type can be one of the following:

– An x, y single point.
– A rectangular area (specified as an x, y, width and height).
– A set of splines that represent an area of the image.
– A free-text comment element.

The image is described in a Cartesian system, with the X-axis horizontal and pointing to the right, the Y-axis vertical and pointing downward, and the origin at the upper left corner. The scale is such that the height of the image is normalized to 1.0. To keep the scale of the X-axis and the Y-axis the same, the image width (R) is its aspect ratio (width/height). Thus, a square part of any image has equal width and height in this coordinate system. The metadata coordinate system refers to the image area on the reference grid as defined in ITU-T T.800 | ISO/IEC 15444-1. See Figure B.1 in ITU-T T.800 | ISO/ IEC 15444-1 for an illustration of the image area. Coordinate (0, 0) refers to the top left of pixel (XOsiz, YOsiz) and coordinate (R, 1) refers to the bottom right of pixel (Xsiz–1, Ysiz–1) on the reference grid where XOsiz, YOsiz, Xsiz and Ysiz are the values of the respective fields in the SIZ marker (see clause A.2.3) in the codestream. Other coordinates map linearly into this image area.

This information may become useless if the image is cropped or manipulated. See Location type (clause N.7.1.15) for the difference between the Position and Location types. See Figure N.66.

```
<xsd:complexType name="tPosition">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element name="POINT" type="jp:tPoint"/>
      <xsd:element name="RECT" type="jp:tRect"/>
      <xsd:sequence>
        <xsd:element name="RECT" type="jp:tRect"/>
        <xsd:element name="REGION" type="jp:tRegion"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:element ref="jp:COMMENT" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="jp:TIMESTAMP"/>
</xsd:complexType>
```

Figure N.66 – Schema of the Position type

POINT: Single point. This element specifies a single point in the coordination system. See Point type (clause N.7.1.18) for more information of this element.

RECT: Rectangular region. This element specifies a rectangular region in the coordinate system. See Rect type (clause N.7.1.19) for more information of this element.

REGION: Arbitrary region. This element specifies an arbitrary region. See Region type (clause N.7.1.20) for more information on this element.

COMMENT: This element can describe the position of an object less accurately than one of the above methods. For example, this element may contain "Bottom left-hand corner" or "Second from the left in the top row." See Comment element (clause N.7.3.1) for more information on this element.

N.7.1.18 Point type

This type specifies details about a single point on an image. This type is used to describe a single point in the coordinate system. This type shall contain the sub-elements listed below. See Figure N.67.

```
<xsd:complexType name="tPoint">
  <xsd:sequence>
    <xsd:element name="X" type="jp:tNonNegativeDouble"/>
    <xsd:element name="Y" type="jp:tNonNegativeDouble"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure N.67 – Schema of the Point type

X: This element specifies the X coordinate of the point.

Y: This element specifies the Y coordinate of the point.

N.7.1.19 Rect type

This type specifies details about a rectangular region on an image. This type is used to describe a rectangular region in the coordinate system. See Point type (clause N.7.1.18) for the base format of this type. Additionally, this type shall contain the sub-elements listed below. See Figure N.68.

```
<xsd:complexType name="tRect">
  <xsd:complexContent>
    <xsd:extension base="jp:tPoint">
      <xsd:sequence>
        <xsd:element name="WIDTH" type="jp:tNonNegativeDouble"/>
        <xsd:element name="HEIGHT" type="jp:tNonNegativeDouble"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Figure N.68 – Schema of the Rect type

X: The left of the rectangle.

Y: The top of the rectangle.

WIDTH: The width of the rectangular (to the right of X).

HEIGHT: The height of the rectangular (below Y).

N.7.1.20 Region type

This type specifies details about an arbitrary region on an image. This type consists of a start point and one or more segments. Each segment may be either a straight line (specified using a point), or a spline.

Where an arbitrary region is specified, a Rectangular Region shall also be specified (which is the bounding box of the Arbitrary Region). A standard JPX compliant metadata reader or editor has the option of not using the Arbitrary Region, even if the Rectangular Region is used.

This type shall contain the sub-elements listed below. See Figure N.69.

```
<xsd:complexType name="tRegion">
  <xsd:sequence>
    <xsd:element name="POINT" type="jp:tPoint"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="POINT" type="jp:tPoint"/>
      <xsd:element name="SPLINE">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="X1" type="jp:tNonNegativeDouble"/>
            <xsd:element name="Y1" type="jp:tNonNegativeDouble"/>
            <xsd:element name="X2" type="jp:tNonNegativeDouble"/>
            <xsd:element name="Y2" type="jp:tNonNegativeDouble"/>
            <xsd:element name="X" type="jp:tNonNegativeDouble"/>
            <xsd:element name="Y" type="jp:tNonNegativeDouble"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Figure N.69 – Schema of the Region type

POINT: Start Point: This is the starting point of the spline in the coordinate system. See Point type (clause N.7.1.18) for the format of this element.

POINT: This element specifies a line starting at the end of the previous spline and ending at the new point. See Point type (clause N.7.1.18) for the format of this element.

SPLINE: This element specifies a Bezier curve starting at the end of the previous spline, and ending at the new end point (x, y), with x1, y1 and x2, y2 being the first and second control points of the spline respectively.

N.7.1.21 Product details type

This type specifies details about a product (hardware or software). By combining these three elements, a unique value shall be created. This type may contain the sub-elements listed below. See Figure N.70.

```
<xsd:complexType name="tProductDetails">
  <xsd:sequence>
    <xsd:element name="MANUFACTURER" type="jp:tOrganization" minOccurs="0"/>
    <xsd:element name="MODEL" type="xsd:string" minOccurs="0"/>
    <xsd:element name="SERIAL" type="xsd:string" minOccurs="0"/>
    <xsd:element name="VERSION" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="jp:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure N.70 – Schema of the Product Details type

MANUFACTURER: Manufacturer name. This element specifies the name of the manufacturer or vendor of the product. It is recommended to set the manufacturer name shown on the device. See Organization type (clause N.7.1.14) for the format of this element.

MODEL: Model name. This element specifies the model name or number of the product.

SERIAL: Serial Number. This element specifies the serial number of a product.

VERSION: Version Number. This element specifies the version number of a product.

N.7.2 Defined attributes

N.7.2.1 Language attribute

The attribute is formatted according to RFC 3066. When a metadata element has a Language attribute, it specifies the language in which the metadata is stored. English (e.g., "en") is assumed where the language is not specified.

Where an element specifies a Language attribute, and also sub-elements, the Language of the sub-elements is the same as the enclosing element unless the Language attribute is specified separately within the sub-element. See Figure N.71.

```
<xsd:attribute name="xml:lang" type="xsd:language"/>
```

Figure N.71 – Schema of the Language attribute

xml:lang: This element contains a string values that is RFC 3066 compliant. The syntax of this element shall match the Language Identification format of XML 1.0.

N.7.2.2 Timestamp attribute

When a metadata element contains a Timestamp attribute, it specifies the time that the metadata was generated. Where an element specifies a Timestamp attribute, and also sub-elements, the Timestamp of the sub-elements is the same as the enclosing element unless the Timestamp attribute is specified separately within the sub-element. See Figure N.72.

```
<xsd:attribute name="TIMESTAMP" type="xsd:dateTime"/>
```

Figure N.72 – Schema of the Timestamp attribute

TIMESTAMP: The syntax of this element is specified in XML Schema Part 2.

N.7.3 Defined elements

N.7.3.1 Comment element

The Comment element is used to specify extra information to the element it contains that cannot be described otherwise within the defined metadata. It is recommended that the Comment element is used as a last resort only when the other metadata elements are not suitable to store a specific piece of metadata.

The content of this element is intended to store human readable data. Storing non-human readable data can be performed using other metadata extension methods. See Figure N.73.

```
<xsd:element name="COMMENT">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="jp:tLangString">
        <xsd:attribute ref="jp:TIMESTAMP"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Figure N.73 – Schema of the Comment element

N.8 JPX extended metadata document type definition

```
<!--
    Copyright      (C)      ISO/IEC      2001 -      All      rights      reserved.

    Permission to copy in any form is granted for use with validating and
conforming
    systems and applications as defined in ISO/IEC 15444-2:2001, provided this
    copyright      notice      is      included      with      all      copies.
-->
```

```
<!-- ============================================================================ -
->
<!-- Fundamental Type and Element Definitions                                     -
->
<!-- ============================================================================ -
->
```

```
<!-- HUMAN_SCHEMA_DTD_LOCATION:http://www.jpeg.org/metadata/15444-2.PDF -->
```

```
<!-- Attribute definitions -->
```

```
<!ENTITY % att-timestamp                    "TIMESTAMP CDATA #IMPLIED">
<!ENTITY % att-lang                         "xml:lang CDATA #IMPLIED">
<!ENTITY % att-lang-ts                      "%att-lang; %att-timestamp;">
<!ENTITY % att-lang-ts-id                  "%att-lang-ts; ID CDATA #IMPLIED">

<!-- Geometric Type -->

<!ENTITY % size                            "(WIDTH, HEIGHT)">

<!-- Date Type -->

<!ENTITY % jp2-tDateTime                   "(EXACT | DATE
                                            (MONTH?, YEAR?, CENTURY?)),
                                            WEEK_DAY?, SEASON?, COMMENT?">

<!ELEMENT EXACT                            (#PCDATA)>
<!ELEMENT DATE                             (#PCDATA)>
<!ELEMENT MONTH                            (#PCDATA)>
<!ELEMENT YEAR                             (#PCDATA)>
<!ELEMENT CENTURY                          (#PCDATA)>
<!ELEMENT WEEK_DAY                         (#PCDATA)>
<!ELEMENT SEASON                           (#PCDATA)>

<!--  Address type -->

<!ENTITY % jp2-tAddress                    "(ADDR_NAME?, ADDR_COMP*,
                                            (POSTCODE | ZIPCODE)?,
                                            COUNTRY?)">
<!ELEMENT ADDRESS                          %jp2-tAddress;>
<!ATTLIST ADDRESS                          TYPE CDATA #IMPLIED
                                           %att-lang-ts;>

<!ELEMENT ADDR_NAME                        (#PCDATA)>
<!ATTLIST ADDR_NAME                        %att-lang;>

<!ELEMENT ADDR_COMP                        (#PCDATA)>
<!ATTLIST ADDR_COMP                        TYPE CDATA #IMPLIED>
```

```
<!ELEMENT POSTCODE                          (#PCDATA)>
<!ELEMENT ZIPCODE                           (#PCDATA)>
```

```
<!ELEMENT COUNTRY                           (#PCDATA)>
<!ATTLIST COUNTRY                           %att-lang;>
```

```
<!--  Phone number type -->
```

```
<!ENTITY % jp2-tPhone                      "(COUNTRY_CODE?, AREA?,
                                            LOCAL?, EXTENSION?)">
<!ATTLIST PHONE                             TYPE CDATA #IMPLIED
                                            %att-timestamp;>
<!ELEMENT PHONE                             %jp2-tPhone;>
```

```
<!ELEMENT COUNTRY_CODE                      (#PCDATA)>
<!ELEMENT AREA                              (#PCDATA)>
<!ELEMENT LOCAL                             (#PCDATA)>
<!ELEMENT EXTENSION                         (#PCDATA)>
```

```
<!-- Email Address Type-->
```

```
<!ELEMENT EMAIL                              (#PCDATA)>
<!ATTLIST EMAIL                              TYPE CDATA #IMPLIED>
```

```
<!-- Web Address Type-->
```

```
<!ELEMENT WEB                                (#PCDATA)>
<!ATTLIST WEB                                TYPE CDATA #IMPLIED>
```

```
<!-- Organization type -->
```

```
<!ENTITY % jp2-tOrganization                "(ORG_NAME?,
                                             ADDRESS*, PHONE*, EMAIL*, WEB*,
                                             LOGO_FILE?,          LOGO_FORMAT?,
MIME_TYPE?,
                                             COMMENT?)">
```

```
<!ELEMENT ORG_NAME                          (#PCDATA)>
<!ATTLIST ORG_NAME                          %att-lang;>

<!ELEMENT LOGO_FILE                         (#PCDATA)>
<!ELEMENT LOGO_FORMAT                       (#PCDATA)>
<!ELEMENT MIME_TYPE                         (#PCDATA)>

<!-- Person Type-->

<!ENTITY % jp2-tPerson                      "(NAME_TITLE?,
                                             PERSON_NAME*, NICK_NAME*,
                                             JOB_TITLE?,
                                             (PERSON_ORG | ORG_REF)?,
                                             ADDRESS*, PHONE*, EMAIL*, WEB*,
                                             BIRTH_DATE?, AGE?,
                                             COMMENT?)">

<!ELEMENT NAME_TITLE                        (#PCDATA)>
<!ATTLIST NAME_TITLE                        %att-lang;>

<!ELEMENT PERSON_NAME                       (NAME_COMP+)>
<!ATTLIST PERSON_NAME                       %att-lang-ts;>

<!ELEMENT NAME_COMP                         (#PCDATA)>
<!ATTLIST NAME_COMP                         TYPE (Prefix | Given | Family |
                                                  Suffix | Maiden) "Given">

<!ELEMENT NICK_NAME                         (#PCDATA)>
<!ELEMENT JOB_TITLE                         (#PCDATA)>

<!ELEMENT PERSON_ORG                        %jp2-tOrganization;>
<!ATTLIST PERSON_ORG                        %att-lang-ts-id;>

<!ELEMENT ORG_REF                           (#PCDATA)>
```

```
<!ELEMENT BIRTH_DATE                    (#PCDATA)>
<!ELEMENT AGE                           (#PCDATA)>
```

```
<!-- Location type -->
```

```
<!ENTITY % jp2-tLocation                "(COORD_LOC?, ADDRESS?,
                                         GPS?, COMMENT?)">
<!ELEMENT LOCATION                      %jp2-tLocation;>
<!ATTLIST LOCATION                      %att-lang-ts;>
```

```
<!ELEMENT COORD_LOC                     (LONGITUDE?, LATITUDE?, ALTITUDE?)>
<!ATTLIST COORD_LOC                     %att-timestamp;>
```

```
<!ELEMENT LONGITUDE                     (#PCDATA)>
<!ELEMENT LATITUDE                      (#PCDATA)>
<!ELEMENT ALTITUDE                      (#PCDATA)>
```

```
<!-- GPS type -->
```

```
<!ELEMENT GPS                           (GPS_LAT_REF?, GPS_LATITUDE?,
                                         GPS_LONG_REF?, GPS_LONGITUDE?,
                                         GPS_ALTITUDE?, GPS_TIME?,
                                         GPS_SATELLITES?, GPS_STATUS?,
                                         GPS_MEASURE_MODE?, GPS_DOP?,
                                         GPS_SPEED_REF?, GPS_SPEED?,
                                         GPS_TRACK_REF?, GPS_TRACK?,
                                         GPS_IMAGE_DIR_REF?,
GPS_IMAGE_DIR?,
                                         GPS_MAP_DATUM?,
                                         GPS_DEST_LAT_REF?,
                                         GPS_DEST_LATITUDE?,
                                         GPS_DEST_LONG_REF?,
                                         GPS_DEST_LONGITUDE?,
                                         GPS_DEST_BEARING_REF?,
                                         GPS_DEST_BEARING?,
                                         GPS_DEST_DISTANCE_REF?,
                                         GPS_DEST_DISTANCE?)>
```

```
<!ELEMENT GPS_LAT_REF                    (#PCDATA)>
<!ELEMENT GPS_LATITUDE                   (D, M, S?)>
<!ELEMENT GPS_LONG_REF                   (#PCDATA)>
<!ELEMENT GPS_LONGITUDE                  (D, M, S?)>
<!ELEMENT GPS_ALTITUDE                   (#PCDATA)>
<!ELEMENT GPS_TIME                       (#PCDATA)>
<!ELEMENT GPS_SATELLITES                 (#PCDATA)>
<!ELEMENT GPS_STATUS                     (#PCDATA)>
<!ELEMENT GPS_MEASURE_MODE               (#PCDATA)>
<!ELEMENT GPS_DOP                        (#PCDATA)>
<!ELEMENT GPS_SPEED_REF                  (#PCDATA)>
<!ELEMENT GPS_SPEED                      (#PCDATA)>
<!ELEMENT GPS_TRACK_REF                  (#PCDATA)>
<!ELEMENT GPS_TRACK                      (#PCDATA)>
<!ELEMENT GPS_IMAGE_DIR_REF              (#PCDATA)>
<!ELEMENT GPS_IMAGE_DIR                  (#PCDATA)>
<!ELEMENT GPS_MAP_DATUM                  (#PCDATA)>
<!ELEMENT GPS_DEST_LAT_REF               (#PCDATA)>
<!ELEMENT GPS_DEST_LATITUDE              (D, M, S?)>
<!ELEMENT GPS_DEST_LONG_REF              (#PCDATA)>
<!ELEMENT GPS_DEST_LONGITUDE             (D, M, S?)>
<!ELEMENT GPS_DEST_BEARING_REF           (#PCDATA)>
<!ELEMENT GPS_DEST_BEARING               (#PCDATA)>
<!ELEMENT GPS_DEST_DISTANCE_REF          (#PCDATA)>
<!ELEMENT GPS_DEST_DISTANCE              (#PCDATA)>
```

```
<!ELEMENT D                              (#PCDATA)>
<!ELEMENT M                              (#PCDATA)>
<!ELEMENT S                              (#PCDATA)>
```

```
<!-- Direction type-->
```

```
<!ENTITY % jp2-tDirection               "(YAW?, PITCH?, ROLL?, COMMENT?)">
<!ELEMENT DIRECTION                      %jp2-tDirection;>
<!ATTLIST DIRECTION                      %att-lang-ts;>
```

```
<!ELEMENT YAW                            (#PCDATA)>
<!ELEMENT PITCH                          (#PCDATA)>
<!ELEMENT ROLL                           (#PCDATA)>
```

```
<!-- Position type -->
<!ENTITY % jp2-tPosition                    "((POINT | RECT | (RECT, REGION))?,
                                             COMMENT?)">
<!ELEMENT POSITION                          %jp2-tPosition;>
<!ATTLIST POSITION                          %att-lang-ts;>
```

```
<!ELEMENT POINT                             (X, Y)>
<!ELEMENT RECT                              (X, Y, WIDTH, HEIGHT)>
<!ELEMENT SPLINE                            (X1, Y1, X2, Y2, X, Y)>
<!ELEMENT REGION                            (POINT, (POINT | SPLINE)*)>
```

```
<!ELEMENT X                                 (#PCDATA)>
<!ELEMENT Y                                 (#PCDATA)>
<!ELEMENT WIDTH                             (#PCDATA)>
<!ELEMENT HEIGHT                            (#PCDATA)>
<!ELEMENT X1                                (#PCDATA)>
<!ELEMENT Y1                                (#PCDATA)>
<!ELEMENT X2                                (#PCDATA)>
<!ELEMENT Y2                                (#PCDATA)>
```

```
<!-- Product Details Type -->
```

```
<!ENTITY % jp2-tProductDetails               "(MANUFACTURER?, MODEL?, SERIAL?,
VERSION?)">
```

```
<!ELEMENT MANUFACTURER                      %jp2-tOrganization;>
<!ATTLIST MANUFACTURER                      %att-lang-ts-id;>
<!ELEMENT MODEL                             (#PCDATA)>
<!ELEMENT SERIAL                            (#PCDATA)>
<!ELEMENT VERSION                           (#PCDATA)>
```

```
<!-- Comment element -->
```

```
<!ELEMENT COMMENT                           (#PCDATA)>
<!ATTLIST COMMENT                           %att-lang-ts;>
```

```
<!-- ========================================================================== -
->
<!-- Image Creation Metadata                                                    -
->
<!-- ========================================================================== -
->
```

```
<!ELEMENT IMAGE_CREATION                    (GENERAL_CREATION_INFO?,
                                             CAMERA_CAPTURE?,
                                             SCANNER_CAPTURE?,
                                             SOFTWARE_CREATION?,
                                             CAPTURED_ITEM?)>
<!ATTLIST IMAGE_CREATION                    %att-lang-ts;>
```

```
<!-- General Image Creation -->
```

```
<!ELEMENT GENERAL_CREATION_INFO             (CREATION_TIME?, IMAGE_SOURCE?,
                                             SCENE_TYPE?, IMAGE_CREATOR?,
                                             OPERATOR_ORG?, OPERATOR_ID?)>
<!ATTLIST GENERAL_CREATION_INFO             %att-lang-ts;>
```

```
<!ELEMENT CREATION_TIME                     (#PCDATA)>
```

```
<!ELEMENT IMAGE_SOURCE                      (#PCDATA)>
<!ATTLIST IMAGE_SOURCE                      %att-lang;>
```

```
<!ELEMENT SCENE_TYPE                        (#PCDATA)>
<!ATTLIST SCENE_TYPE                        %att-lang;>
```

```
<!ELEMENT IMAGE_CREATOR                     %jp2-tPerson;>
<!ATTLIST IMAGE_CREATOR                     %att-lang-ts-id;>
```

```
<!ELEMENT OPERATOR_ORG                      %jp2-tOrganization;>
<!ATTLIST OPERATOR_ORG                      %att-lang-ts-id;>
```

```
<!ELEMENT OPERATOR_ID                       (#PCDATA)>
<!ATTLIST OPERATOR_ID                       %att-lang;>
```

```
<!-- Camera capture -->

<!ELEMENT CAMERA_CAPTURE                    (CAMERA_INFO?, SOFTWARE_INFO?,
                                             LENS_INFO?, DEVICE_CHARACTER?,
                                             CAMERA_SETTINGS?, ACCESSORY*)>
<!ATTLIST CAMERA_CAPTURE                    %att-lang-ts;>

<!ELEMENT CAMERA_INFO                       %jp2-tProductDetails;>
<!ATTLIST CAMERA_INFO                       %att-lang-ts;>

<!ELEMENT SOFTWARE_INFO                     %jp2-tProductDetails;>
<!ATTLIST SOFTWARE_INFO                     %att-lang-ts;>

<!ELEMENT LENS_INFO                         %jp2-tProductDetails;>
<!ATTLIST LENS_INFO                         %att-lang-ts;>

<!ELEMENT DEVICE_CHARACTER                  (SENSOR_TECHNOLOGY?,
                                             FOCAL_PLANE_RES?,
                                             SPECTRAL_SENSITIVITY?,
                                             ISO_SATURATION?, ISO_NOISE?,
                                             SPATIAL_FREQ_RESPONSE?,
                                             CFA_PATTERN?,              OECF?,
MIN_F_NUMBER?)>
<!ATTLIST DEVICE_CHARACTER                  %att-lang-ts;>

<!ELEMENT SENSOR_TECHNOLOGY                 (#PCDATA)>

<!ELEMENT FOCAL_PLANE_RES                   %size;>

<!ELEMENT SPECTRAL_SENSITIVITY              ANY>
<!ELEMENT ISO_SATURATION                    (#PCDATA)>
<!ELEMENT ISO_NOISE                         (#PCDATA)>
```

```
<!ELEMENT SPATIAL_FREQ_RESPONSE          (SPATIAL_FREQ_VAL+)>
<!ELEMENT SPATIAL_FREQ_VAL              (SPATIAL_FREQ, HORIZ_SFR, VERT_SFR)>
<!ELEMENT SPATIAL_FREQ                   (#PCDATA)>
<!ELEMENT HORIZ_SFR                      (#PCDATA)>
<!ELEMENT VERT_SFR                       (#PCDATA)>
```

```
<!ELEMENT CFA_PATTERN                    (COLOR_ROW+)>
<!ELEMENT COLOR_ROW                      (COLOR+)>
<!ELEMENT COLOR                          (#PCDATA)>
```

```
<!ELEMENT OECF                           (LOG_VAL+)>
<!ELEMENT LOG_VAL                        (LOG_EXPOSURE, OUTPUT_LEVEL+)>
<!ELEMENT LOG_EXPOSURE                   (#PCDATA)>
<!ELEMENT OUTPUT_LEVEL                   (#PCDATA)>
```

```
<!ELEMENT MIN_F_NUMBER                   (#PCDATA)>
```

```
<!-- Camera Capture Settings -->
```

```
<!ELEMENT CAMERA_SETTINGS                ((EXP_TIME | R_EXP_TIME)?,
                                          F_NUMBER?, EXP_PROGRAM?,
                                          BRIGHTNESS?, EXPOSURE_BIAS?,
                                          SUBJECT_DISTANCE?,
METERING_MODE?,
                                          SCENE_ILLUMINANT?, COLOR_TEMP?,
                                          FOCAL_LENGTH?, FLASH?,
                                          FLASH_ENERGY?, FLASH_RETURN?,
                                          BACK_LIGHT?, SUBJECT_POSITION?,
                                          EXPOSURE_INDEX?, AUTO_FOCUS?,
                                          SPECIAL_EFFECT*,
CAMERA_LOCATION?,
                                          ORIENTATION?, PAR?)>
<!ATTLIST CAMERA_SETTINGS                 %att-lang-ts;>
```

```
<!ELEMENT EXP_TIME                      (#PCDATA)>
<!ELEMENT R_EXP_TIME                    (#PCDATA)>
<!ELEMENT F_NUMBER                      (#PCDATA)>
<!ELEMENT EXP_PROGRAM                   (#PCDATA)>
<!ATTLIST EXP_PROGRAM                   %att-lang;>
<!ELEMENT BRIGHTNESS                    (#PCDATA)>
<!ELEMENT EXPOSURE_BIAS                 (#PCDATA)>
<!ELEMENT SUBJECT_DISTANCE              (#PCDATA)>
<!ELEMENT METERING_MODE                 (#PCDATA)>
<!ATTLIST METERING_MODE                 %att-lang;>
<!ELEMENT SCENE_ILLUMINANT              (#PCDATA)>
<!ATTLIST SCENE_ILLUMINANT              %att-lang;>
<!ELEMENT COLOR_TEMP                    (#PCDATA)>
<!ELEMENT FOCAL_LENGTH                  (#PCDATA)>
<!ELEMENT FLASH                         (#PCDATA)>
<!ELEMENT FLASH_ENERGY                  (#PCDATA)>
<!ELEMENT FLASH_RETURN                  (#PCDATA)>
<!ELEMENT BACK_LIGHT                    (#PCDATA)>
<!ELEMENT SUBJECT_POSITION              %jp2-tPosition;>
<!ATTLIST SUBJECT_POSITION              %att-lang-ts;>
<!ELEMENT EXPOSURE_INDEX                (#PCDATA)>
<!ELEMENT AUTO_FOCUS                    (#PCDATA)>
<!ELEMENT SPECIAL_EFFECT                (#PCDATA)>
<!ELEMENT CAMERA_LOCATION               %jp2-tLocation;>
<!ATTLIST CAMERA_LOCATION               %att-lang-ts;>
<!ELEMENT ORIENTATION                   %jp2-tDirection;>
<!ATTLIST ORIENTATION                   %att-lang-ts;>
<!ELEMENT PAR                           (#PCDATA)>
```

```
<!ELEMENT ACCESSORY                     %jp2-tProductDetails;>
<!ATTLIST ACCESSORY                     %att-lang-ts;>
```

```
<!-- Scanner Capture -->
```

```
<!ELEMENT SCANNER_CAPTURE               (SCANNER_INFO?, SOFTWARE_INFO?,
                                         SCANNER_SETTINGS?)>
<!ATTLIST SCANNER_CAPTURE               %att-lang-ts;>
```

```
<!ELEMENT SCANNER_INFO                  %jp2-tProductDetails;>
<!ATTLIST SCANNER_INFO                  %att-lang-ts;>
```

```
<!ELEMENT SCANNER_SETTINGS                (PIXEL_SIZE?, PHYSICAL_SCAN_RES?)>
<!ATTLIST SCANNER_SETTINGS                 %att-timestamp;>

<!ELEMENT PIXEL_SIZE                       (#PCDATA)>
<!ELEMENT PHYSICAL_SCAN_RES                %size;>

<!-- Software Creation -->
<!ELEMENT SOFTWARE_CREATION                (SOFTWARE_INFO?)>

<!-- Captured Item -->

<!ELEMENT CAPTURED_ITEM                    (REFLECTION_PRINT | FILM)>
<!ATTLIST CAPTURED_ITEM                    %att-lang-ts;>

<!-- Reflection print -->

<!ELEMENT REFLECTION_PRINT                 (DOCUMENT_SIZE?, MEDIUM?, RP_TYPE?)>

<!ELEMENT DOCUMENT_SIZE                    %size;>
<!ELEMENT MEDIUM                           (#PCDATA)>
<!ELEMENT RP_TYPE                          (#PCDATA)>

<!-- Film -->

<!ELEMENT FILM                             (BRAND?, CATEGORY?, FILM_SIZE?,
                                            ROLL_ID?, FRAME_ID?, FILM_SPEED?)>
<!ATTLIST FILM                             %att-lang-ts;>

<!ELEMENT BRAND                            %jp2-tProductDetails;>
<!ATTLIST BRAND                            %att-lang-ts;>
<!ELEMENT CATEGORY                         (#PCDATA)>
<!ELEMENT FILM_SIZE                        %size;>
<!ELEMENT ROLL_ID                          (#PCDATA)>
<!ATTLIST ROLL_ID                          %att-lang;>
<!ELEMENT FRAME_ID                         (#PCDATA)>
<!ELEMENT FILM_SPEED                       (#PCDATA)>
```

```
<!-- ================================================================== -
->
<!-- Content Description                                                -
->
<!-- ================================================================== -
->

<!ELEMENT CONTENT_DESCRIPTION              (GROUP_CAPTION?, CAPTION?,
                                            CAPTURE_TIME?, LOCATION?,
                                            PERSON*, THING*, ORGANIZATION*,
                                            EVENT*, AUDIO*, PROPERTY*,
                                            DICTIONARY*, COMMENT?)>
<!ATTLIST CONTENT_DESCRIPTION              %att-lang-ts;>

<!ELEMENT GROUP_CAPTION                    (#PCDATA)>
<!ATTLIST GROUP_CAPTION                    %att-lang;>

<!ELEMENT CAPTION                          (#PCDATA)>
<!ATTLIST CAPTION                          %att-lang;>

<!ELEMENT CAPTURE_TIME                     (%jp2-tDateTime;)>
<!ATTLIST CAPTURE_TIME                     %att-lang-ts;>

<!-- Person -->

<!ELEMENT PERSON                           (%jp2-tPerson;, POSITION?,
                                            LOCATION?, PROPERTY*)>
<!ATTLIST PERSON                           %att-lang-ts-id;>

<!-- Thing -->

<!ELEMENT THING                            (NAME?, COMMENT?, POSITION?,
                                            LOCATION?, PROPERTY*, THING*)>
<!ATTLIST THING                            %att-lang-ts-id;>

<!-- Organization -->
```

```
<!ELEMENT ORGANIZATION                    (%jp2-tOrganization;, POSITION?,
                                           LOCATION?, PROPERTY*)>
<!ATTLIST ORGANIZATION                     %att-lang-ts-id;>
```

```
<!-- Event -->
```

```
<!ELEMENT EVENT                           (EVENT_TYPE?, DESCRIPTION?,
                                           LOCATION?, EVENT_TIME?, DURATION?,
                                           COMMENT?, PARTICIPANT*,
                                           EVENT_RELATION*,
                                           (EVENT | EVENT_REF)*)>
<!ATTLIST EVENT                            %att-lang-ts-id;>
```

```
<!ELEMENT EVENT_TYPE                       (#PCDATA)>
<!ATTLIST EVENT_TYPE                       %att-lang;>
```

```
<!ELEMENT DESCRIPTION                      (#PCDATA)>
<!ATTLIST DESCRIPTION                      %att-lang;>
```

```
<!ELEMENT EVENT_TIME                       (%jp2-tDateTime;)>
<!ATTLIST EVENT_TIME                       %att-lang-ts;>
```

```
<!ELEMENT DURATION                         (#PCDATA)>
```

```
<!ELEMENT PARTICIPANT                      (ROLE+,
                                           (OBJECT_REF | PERSON | THING |
ORGANIZATION))>
<!ATTLIST PARTICIPANT                      %att-lang;>
```

```
<!ELEMENT ROLE                             (#PCDATA)>
<!ATTLIST ROLE                             %att-lang;>
```

```
<!ELEMENT OBJECT_REF                       (#PCDATA)>
```

```
<!ELEMENT EVENT_RELATION                   (RELATION*, EVENT_REF+)>
```

```
<!ELEMENT RELATION                          (#PCDATA)>
<!ATTLIST RELATION                          %att-lang;>

<!ELEMENT EVENT_REF                         (#PCDATA)>

<!-- Audio -->

<!ELEMENT AUDIO                            (AUDIO_STREAM?, AUDIO_FORMAT?,
                                            MIME_TYPE?,          DESCRIPTION?,
COMMENT?)>
<!ATTLIST AUDIO                             %att-lang-ts;>

<!ELEMENT AUDIO_STREAM                      (#PCDATA)>
<!ELEMENT AUDIO_FORMAT                      (#PCDATA)>

<!-- Property -->

<!ELEMENT PROPERTY                         (NAME?, VALUE*, COMMENT?, PROPERTY*)>
<!ATTLIST PROPERTY                          %att-lang-ts;
                                            DICT_REF CDATA #IMPLIED>

<!ELEMENT NAME                             (#PCDATA)>
<!ATTLIST NAME                              %att-lang;>

<!ELEMENT VALUE                            (#PCDATA)>
<!ATTLIST VALUE                             %att-lang;>

<!-- Dictionary Reference -->

<!ELEMENT DICTIONARY                       (DICT_NAME?, COMMENT?)>
<!ATTLIST DICTIONARY                        %att-lang-ts-id;>

<!ELEMENT DICT_NAME                         (#PCDATA)>
<!ATTLIST DICT_NAME                         %att-lang;>
```



```
<!-- ========================================================================== -
->
<!-- History                                                                    -
->
<!-- ========================================================================== -
->
```

```
<!ELEMENT HISTORY                          (PROCESSING_SUMMARY?,
                                            IMAGE_PROCESSING_HINTS?,
METADATA*)>
<!ATTLIST HISTORY                           %att-lang-ts;>
<!ELEMENT METADATA                         (BASIC_IMAGE_PARAM?, IMAGE_CREATION?,
                                             CONTENT_DESCRIPTION?,
                                             HISTORY?,
                                             IPR?)>
```

```
<!-- Summary -->
```

```
<!ELEMENT PROCESSING_SUMMARY               (IMG_CREATED?, IMG_CROPPED?,
                                            IMG_TRANSFORMED?, IMG_GTC_ADJ?,
                                            IMG_STC_ADJ?, IMG_SPATIAL_ADJ?,
                                            IMG_EXT_EDITED?, IMG_RETOUCHED?,
                                            IMG_COMPOSITED?, IMG_METADATA?)>
<!ATTLIST PROCESSING_SUMMARY               %att-timestamp;>
```

```
<!ELEMENT IMAGE_PROCESSING_HINTS(MODIFIER?, (IMG_CREATED | IMG_CROPPED |
                                            IMG_TRANSFORMED | IMG_GTC_ADJ |
                                            IMG_STC_ADJ | IMG_SPATIAL_ADJ |
                                            IMG_EXT_EDITED | IMG_RETOUCHED |
<!ELEMENT MODIFIER%jp2-tProductDetails;>    IMG_COMPOSITED | IMG_METADATA)*)*>
<!ATTLIST IMAGE_PROCESSING_HINTS           %att-lang-ts;>
```

```
<!ELEMENT IMG_CREATED                          (#PCDATA)>
<!ELEMENT IMG_CROPPED                          (#PCDATA)>
<!ELEMENT IMG_TRANSFORMED                      (#PCDATA)>
<!ELEMENT IMG_GTC_ADJ                          (#PCDATA)>
<!ELEMENT IMG_STC_ADJ                          (#PCDATA)>
<!ELEMENT IMG_SPATIAL_ADJ                      (#PCDATA)>
<!ELEMENT IMG_EXT_EDITED                       (#PCDATA)>
<!ELEMENT IMG_RETOUCHED                        (#PCDATA)>
<!ELEMENT IMG_COMPOSITED                       (#PCDATA)>
<!ELEMENT IMG_METADATA                         (#PCDATA)>
```

```
<!-- Previous -->
<!ELEMENT BASIC_IMAGE_PARAM                   (BASIC_IMAGE_INFO)>
<!ATTLIST BASIC_IMAGE_PARAM                    %att-lang-ts;>
```

```
<!ELEMENT BASIC_IMAGE_INFO                    (FILE_FORMAT?, IMAGE_ID?)>
<!ATTLIST BASIC_IMAGE_INFO                     %att-lang-ts;>
```

```
<!-- ================================================================== -
->
<!-- Intellectual Property Rights                                       -
->
<!-- ================================================================== -
->
```

```
<!ELEMENT IPR                                 (IPR_NAMES?, IPR_DESCRIPTION?,
                                               IPR_DATES?, IPR_EXPLOITATION?,
                                               IPR_IDENTIFICATION?,
                                               IPR_CONTACT_POINT?,
IPR_HISTORY?)>
<!ATTLIST IPR                                  %att-lang-ts;>
```

```
<!-- IPR people -->
```

```
<!ELEMENT IPR_NAMES                         (IPR_PERSON?, IPR_ORG?, IPR_NAME_REF?)+>
<!ATTLIST IPR_NAMES                              %att-lang-ts;>
```

```
<!ELEMENT IPR_PERSON                    %jp2-tPerson;>
<!ATTLIST IPR_PERSON                    DESCRIPTION CDATA #IMPLIED
                                        %att-lang-ts-id;>

<!ELEMENT IPR_ORG                       %jp2-tOrganization;>
<!ATTLIST IPR_ORG                       DESCRIPTION CDATA #IMPLIED
                                        %att-lang-ts-id;>

<!ELEMENT IPR_NAME_REF                  (#PCDATA)>
<!ATTLIST IPR_NAME_REF                   DESCRIPTION CDATA #IMPLIED>

<!-- IPR description -->

<!ELEMENT IPR_DESCRIPTION               (IPR_TITLE?, IPR_LEGEND?,
                                         IPR_CAPTION?, COPYRIGHT?)>

<!ELEMENT IPR_TITLE                     (#PCDATA)>
<!ATTLIST IPR_TITLE                      %att-lang-ts;>

<!ELEMENT IPR_LEGEND                    (#PCDATA)>
<!ATTLIST IPR_LEGEND                     %att-lang-ts;>

<!ELEMENT IPR_CAPTION                   (#PCDATA)>
<!ATTLIST IPR_CAPTION                    %att-lang-ts;>

<!ELEMENT COPYRIGHT                     (#PCDATA)>
<!ATTLIST COPYRIGHT                      %att-lang-ts;>

<!ELEMENT IPR_DATES                     (IPR_DATE+)>
<!ATTLIST IPR_DATES                      %att-lang-ts;>

<!ELEMENT IPR_DATE                      (%jp2-tDateTime;)>
<!ATTLIST IPR_DATE                      DESCRIPTION CDATA #IMPLIED
                                        %att-lang-ts;>
```

```
<!-- IPR exploitation -->

<!ELEMENT IPR_EXPLOITATION                (IPR_PROTECTION?,
                                           IPR_USE_RESTRICTION?,
                                           IPR_OBLIGATION?,
                                           IPR_MGMT_SYS?)>
<!ATTLIST IPR_EXPLOITATION                 %att-lang-ts;>

<!ELEMENT IPR_PROTECTION                  (#PCDATA)>

<!ELEMENT IPR_USE_RESTRICTION             (#PCDATA)>
<!ATTLIST IPR_USE_RESTRICTION              %att-lang;>

<!ELEMENT IPR_OBLIGATION                  (#PCDATA)>
<!ATTLIST IPR_OBLIGATION                   %att-lang;>

<!-- IPR management system -->

<!ELEMENT IPR_MGMT_SYS                    (IPR_MGMT_TYPE?,
                                           IPR_MGMT_SYS_ID?,
                                           IPR_MGMT_SYS_LOCATION?)>
<!ATTLIST IPR_MGMT_SYS                     %att-lang-ts;>

<!ELEMENT IPR_MGMT_TYPE                   (#PCDATA)>
<!ELEMENT IPR_MGMT_SYS_ID                 (#PCDATA)>

<!ELEMENT IPR_MGMT_SYS_LOCATION           (#PCDATA)>

<!-- IPR identification -->

<!ELEMENT IPR_IDENTIFICATION              (IPR_IDENTIFIER?,
                                           LICENCE_PLATE?)>
<!ATTLIST IPR_IDENTIFICATION               %att-lang-ts;>
```

```
<!ELEMENT IPR_IDENTIFIER                    (IPR_ID_MODE?, IPR_ID?)>

<!ELEMENT IPR_ID_MODE                       (#PCDATA)>
<!ATTLIST IPR_ID_MODE                        %att-lang;>
<!ELEMENT IPR_ID                            (#PCDATA)>
<!ATTLIST IPR_ID                             %att-lang;>

<!ELEMENT LICENCE_PLATE                     (LP_COUNTRY?,
                                             LP_REG_AUT?,
                                             LP_REG_NUM?,
                                             LP_DELIVERY_DATE?)>

<!ELEMENT LP_COUNTRY                        (#PCDATA)>
<!ELEMENT LP_REG_AUT                        (#PCDATA)>
<!ELEMENT LP_REG_NUM                        (#PCDATA)>
<!ELEMENT LP_DELIVERY_DATE                  (#PCDATA)>

<!-- IPR contact point -->

<!ELEMENT IPR_CONTACT_POINT                (IPR_PERSON | IPR_ORG | IPR_NAME_REF)>
<!ATTLIST IPR_CONTACT_POINT                  %att-lang-ts;>

<!-- IPR History -->

<!ELEMENT IPR_HISTORY                       (IPR+)>
<!ATTLIST IPR_HISTORY                       %att-lang-ts;>

<!-- ========================================================================= -
->
<!-- Image Identifier                                                          -
->
<!-- ========================================================================= -
->

<!ELEMENT IMAGE_ID                           (UID?, ID_TYPE?)>
<!ELEMENT UID                               (#PCDATA)>
<!ELEMENT ID_TYPE                           (#PCDATA)>
```

N.9 JPX extended metadata XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xsd:schema PUBLIC "-//W3C//DTD XMLSchema 200102//EN"

"http://www.w3.org/2001/XMLSchema.dtd" [
  <!ENTITY % p 'xsd:'>
  <!ENTITY % s ':xsd'>
]>
```

```
<!--
    Copyright (C) ISO/IEC 2001 - All rights reserved.

    Permission to copy in any form is granted for use with validating and
conforming
    systems and applications as defined in ISO/IEC 15444-2:2001, provided this
    copyright notice is included with all copies.
-->
```

```
<!-- HUMAN_SCHEMA_DTD_LOCATION:http://www.jpeg.org/metadata/15444-2.PDF -->
```

```
<xsd:schema                 targetNamespace="http://www.jpeg.org/jpx/1.0/xml"
            xmlns:jp="http://www.jpeg.org/jpx/1.0/xml"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xml="http://www.w3.org/XML/1998/namespace"
             xmlns="http://www.jpeg.org/jpx/1.0/xml"

             elementFormDefault="qualified">
```

```
   <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- -
     -  Fundamental Metadata Types, Fields and Attributes
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-->
```

```
   <!--
     -  See clause N.7.2.1 Language attribute
     -->
   <!-- Import the xml:lang attribute definition defined by W3C -->
   <xsd:import namespace="http://www.w3.org/XML/1998/namespace"

schemaLocation="http://www.w3.org/2001/xml.xsd"/>
```

```
<!--
  -  See clause N.7.2.2 Timestamp attribute
  -->
<xsd:attribute name="TIMESTAMP" type="xsd:dateTime"/>
```

```
<!--
  -  See clause N.7.1.3 String including language attribute type
  -->
<xsd:complexType name="tLangString">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute ref="xml:lang"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

```
<!--
  -  See clause N.7.1.1 Non-negative double type
  -->
<xsd:simpleType name="tNonNegativeDouble">
  <xsd:restriction base="xsd:double">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
  -  See clause N.7.1.2 Rational type
  -->
<xsd:simpleType name="tRational">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(\-|\+)?[0-9]+/[0-9]+"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
  -  See clause N.7.1.4 Degree type
  -->
<xsd:simpleType name="tDegree">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="-180"/>
    <xsd:maxInclusive value="180"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
  -  See clause N.7.1.5 Half degree type
  -->
<xsd:simpleType name="tHalfDegree">
 <xsd:restriction base="xsd:double">
  <xsd:minExclusive value="-90"/>
  <xsd:maxInclusive value="90"/>
 </xsd:restriction>
</xsd:simpleType>
```

```
<!--
  -  See clause N.7.1.6 Double size type  and
  -->
<xsd:complexType name="tDoubleSize">
  <xsd:sequence>
    <xsd:element name="WIDTH" type="jp:tNonNegativeDouble"/>
    <xsd:element name="HEIGHT" type="jp:tNonNegativeDouble"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<!--
  -  See clause N.7.1.7 Integer size type
  -->
<xsd:complexType name="tIntSize">
  <xsd:sequence>
    <xsd:element name="WIDTH" type="xsd:positiveInteger"/>
    <xsd:element name="HEIGHT" type="xsd:positiveInteger"/>
  </xsd:sequence>
</xsd:complexType>
```