ISO/IEC JTC 1/SC 29/WG 1 N2000

Date: 7 December 2000

ISO/IEC JTC 1/SC 29/WG 1
(ITU-T SG8)

Coding of Still Pictures

JBIG
Joint Bi-level Image Experts Group

J P E G
Joint Photographic Experts Group

TITLE: JPEG 2000 Part II Final Committee Draft

SOURCE: ISO/IEC JTC1/SC29 WG1, JPEG 2000 Editor Martin Boliek, Co-editors Eric Majani, J. Scott Houchin, James Kasner, and Mathias Larsson Carlander

PROJECT: 1.29.15444 (JPEG 2000)

STATUS:

REQUESTED ACTION: For review and action as appropriate

DISTRIBUTION: WG1

Contact:

Convenor, ISO/IEC JTC 1/SC 29/WG 1
Dr. Daniel T. Lee
Yahoo!
3420 Central Expressway,
Santa Clara, California 95051, USA
Tel: +1 408 992 7051, Fax: +1 253 830 0372,
E-mail: dlee@yahoo-inc.com

Editor, 15444 (JPEG-2000)
Mr. Martin Boliek
Ricoh Innovations, Inc.
2882 Sand Hill Road, Suite 115
Menlo Park, CA 94025, USA
Tel: 1 650 496 5705, Fax: 1 650 854 8740
Internet: boliek@rsv.ricoh.com

INFORMATION TECHNOLOGY

JPEG 2000 IMAGE CODING SYSTEM: EXTENSIONS

JPEG 2000 PART II FINAL COMMITTEE DRAFT, 7 DECEMBER 2000

THE ISO AND ITU WILL PROVIDE COVER PAGES.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Foreword

Forward to be supplied by ISO and ITU.

Introduction

Introduction to be supplied by ISO and ITU.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY — JPEG 2000 IMAGE CODING SYSTEM: PART II 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, or colour digital still 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 References

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

2.1 Identical Recommendations | International Standards

ITU-T Recommendation T.81 | ISO/IEC 10918-1:1994, Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines.

ITU-T Recommendation T.82 | ISO/IEC 11544:1994, Information technology - Coded representation of picture and audio information Progressive bi-level image compression

ITU-T Recommendation T.83 | ISO/IEC 10918-2:1995, Information technology - Digital compression and coding of continuous-tone still images: Compliance testing.

ITU-T Recommendation T.84 | ISO/IEC 10918-3:1996, Information technology - Digital compression and coding of continuous-tone still images: Extensions.

ITU-T Recommendation T.84 | ISO/IEC 10918-3 Amd 1 (In preparation), Information technology - Digital compression and coding of continuous-tone still images: Extensions - Amendment 1.

ITU-T Recommendation T.86 | ISO/IEC 10918-4, Information technology - Digital compression and coding of continuous-tone still images: Registration of JPEG Profiles, SPIFF Profiles, SPIFF Tags, SPIFF colour Spaces, APPn Markers, SPIFF, Compression types and Registration authorities (REGAUT).

ITU-T Recommendation T.87 | ISO/IEC 14495-1, Lossless and near-lossless compression of continuous-tone still images-baseline.

ITU-T Recommendation T.88 | ISO/IEC 14492-1, Lossy/lossless coding of bi-level images

ITU-T Recommendation T.800 | ISO/IEC 15444-1, JPEG 2000 Image Coding System

ITU—T T.42. Continuous tone colour representation method for facsimile. October 1996.

ISO/IEC 8859-1:1998, Information technology 8-bit single-byte coded graphic character sets Part 1: Latin alphabet No. 1.

ISO 8601:1988, Data elements and interchange formats Information interchange Representation of dates and times.

ISO 3166-1:1997, Codes for the representation of names of countries and their subdivisions Part 1: Country codes.

ISO 3166-2:1998, Codes for the representation of names of countries and their subdivisions Part 2: Country subdivision code.

ISO/IEC 11578:1996 Information technology Open Systems Interconnection Remote Procedure Call, <http://www.iso.ch/cate/d2229.html>.

ISO/IEC 646:1991, ISO 7-bit coded character set for information interchange.

ISO 5807:1985, Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts.

ISO/IEC 15938. MPEG—7

ISO 10126—2. Banking Procedures for message encipherment (wholesale) Part 2: DEA algorithm

IEEE Std. 754-1985 R1990, IEEE Standard for Binary Floating-Point Arithmetic

IETF RFC 1321. The MD5 Message-Digest Algorithm, April 1992. <http://www.ietf.org/rfc/rfc1321.txt>

IETF RFC 1766, Tags for Identification of Languages, March 1995.

IETF RFC 2279, UTF—8, A transformation format of ISO 10646, January 1998.

IETF RFC 2630. R. Housley, Cryptographic Message Syntax, June 1999 (available at http://www.ietf.org/rfc/rfc2630.txt)

IETF RFC 2313. B. Kaliski, PKCS #1: RSA Encryption, Version 1.5, March 1998 (available at http://www.ietf.org/rfc/rfc2313.txt)

International Color Consortium, ICC profile format specification. ICC.1:1998—09

International Electrotechnical Commission. Color management in multimedia systems: Part 2: Colour Management, Part 2—1: Default RGB colour space sRGB. IEC 61966—2—1 1998. 9 October 1998.

W3C, Extensible Markup Language (XML 1.0), Rec-xml-19980210

Digital Imaging Group, Flashpix digital image file format. Version 1.0.1. 10 July 1997

PIMA 7666. Photography Electronics still picture imaging Reference Output Medium Metric RGB Color encoding: ROMM—RGB

PIMA 7667:2001. Photography Electronics still picture imaging Extended sRGB color encoding e-sRGB

Federal Information Processing Standard Publication (FIPS PUB) 186—2, Digital Signature Standard (DSS). <http://www.itl.nist.gov/fipspubs/fip186-2.pdf>

ANSI X9.30.2. Public Key Cryptography for the Financial Services Industry Part 2: The Secure Hash Algorithm (SHA—1). <http://www.itl.nist.gov/fipspubs/fip180-1.htm>

W3C, Extensible Markup Language (XML 1.0), Rec-xml-19980210, <http://www.w3.org/TR/REC-xml>.

W3C, Namespaces in XML, Rec-xml-names-19990114, <http://www.w3.org/TR/1999/REC-xml-names>.

W3C, XML Schema Part 1: Structures, WD-xmlschema-1-20000407, <http://www.w3.org/TR/xmlschema-1>.

W3C, XML Schema Part 2: Datatypes, WD-xmlschema-2-20000407, <http://www.w3.org/TR/xmlschema-2>.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. The definitions defined in ITU-T T.800 | IS 15444-1 Section 3 also apply to this Recommendation | International Standard except for the terms decomposition level, subband and resolution which are re-defined in this section.

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

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

3.3 **component**: Compressed data from the codestream representing a single set of two-dimensional data.

3.4 **component collection**: A subset of spatially reconstructed components used as inputs to the inverse multiple component transform process, or a subset of image components obtained as outputs from the inverse multiple component transform process. The subset's constituent components may occur in an arbitrary order, i.e., input spatially reconstructed components may occur other than in order of appearance in the codestream, while output image components may occur other than in order of image component.

3.5 **component reconstruction matrices:** A general term that refers to any of the following; decorrelation transform matrix, decorrelation offset matrix, dependency transform matrix or dependency offset matrix.

3.6 **compositing:** the act of combining two compositing layers into a single, non-redundant set of image channels.

3.7 **compositing layer**: a set of non-redundant channels drawn from one or more codestreams that shall be treated as a group. The set of compositing layers within the JPX file may then be combined by compositing or animation instructions to form a rendered result. 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.

3.8 **deadzone**: The interval within which all subband coefficients are quantized to 0.

3.9 **decomposition level**: A collection of subbands where each coefficient has the same spatial impact or span with respect to the original samples. These include the LL, LH, HL, HH, LX, HX, XL, and XH subband splits out of decomposition sublevels.

3.10 **decomposition sub-level**: A collection of subbands that result from splits of a subband from a lower decomposition sub-level or splits of either LL, LX or XL subbands from a higher decomposition level.

3.11 **decorrelation offset matrix:** A floating-point N x 1 matrix containing offsets which are added to spatially reconstructed components prior to the application of a decorrelation matrix.

3.12 decorrelation transform matrix: A floating-point N x M matrix that maps M spatially reconstructed components, by linear combination, into N intermediate components. This matrix is not necessarily square in dimension.

3.13 dependency offset matrix: A floating point Q x 1 matrix that contain offsets which are added to intermediate components prior to application of a dependency transformation matrix.

3.14 dependency transform matrix: A floating point Q x N matrix that maps N intermediate components into Q reconstructed image components. This matrix is not necessarily square in dimension.

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

3.16 HX subband: The subband obtained by forward horizontal high-pass analysis filtering and no vertical analysis filtering. This subband contributes to reconstruction with inverse horizontal high-pass synthesis filtering and no vertical synthesis filtering.

3.17 image component: A single two-dimensional array of data resulting from application of the inverse dependency removal process.

3.18 intermediate component: A single two-dimensional array of data resulting from application of the inverse decorrelation process.

3.19 inverse decorrelation process: An application of the decorrelation transform and decorrelation offset matrices to a set of spatially reconstructed components.

3.20 inverse dependency removal process: An application of the dependency transform and dependency offset matrices to a set of intermediate components.

3.21 JPX file: The name of file in the file format described in this Recommendation | International Standard. Structurally, a JPX file is a contiguous sequence of boxes.

3.22 LX subband: The subband obtained by forward horizontal low-pass analysis filtering and no vertical analysis filtering. This subband contributes to reconstruction with inverse horizontal low-pass synthesis filtering and no vertical synthesis filtering.

3.23 metadata: Additional data associated with the image data beyond the image data.

3.24 namespace: A 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.

3.25 reconstructed image component: Intermediate components that have been processed by the dependency offset and dependency transform matrices form reconstructed image components.

3.26 rendered result: The result generated by combining the compositing layers in the JPX file, either by composition or animation.

3.27 resolution: The spatial relation of samples to a physical space. 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 decomposition level includes either an LL, LX or XL subband which is considered to be a lower resolution. Therefore, there is one more resolution level than decomposition levels.

3.28 subband: A 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.

3.29 spatially reconstructed component: A component which has been extracted from the codestream and passed through the decoding and inverse wavelet transform process as specified by this standard.

3.30 visual masking: Visual masking is a mechanism where artifacts are masked by the image acting as a background signal.

3.31 XH subband: The subband obtained by no forward horizontal analysis filtering and vertical high-pass analysis filtering. This subband contributes to reconstruction with vertical high-pass synthesis filtering and no inverse horizontal synthesis filtering.

3.32 XL subband: The subband obtained by no forward horizontal analysis filtering and vertical low-pass analysis filtering. This subband 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 ITU-T T.800 | IS 15444-1 section 4 also apply to this Recommendation | International Standard

IPR: Intellectual Property Rights

UUID: Universal Unique Identifier

CCITT: International Telegraph and Telephone Consultative Committee, now ITU-T

DPI: Dots per inch.

5 Symbols

For the purposes of this Recommendation | International Standard, the following symbols apply. The symbols defined in ITU-T T.800 | IS 15444-1 section 4 also apply to this Recommendation | International Standard.

DCO: Variable DC offset marker

VMS: Visual masking marker

ATK: Arbitrary transformation kernels marker

ADS: Arbitrary decomposition marker

MCT: Multiple component transformation definition marker

MCC: Multiple component collection marker

NLT: Non-linear point transformation marker

ARN: Arbitrary region of interest marker

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 clause 3 and 4 in ITU-T Recommendation 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, or colour digital still images. This set of methods extends the elements in the core coding system described in ITU-T Recommendation 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 ITU-T Recommendation 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 theses extensions will also support configuration subsets that are currently defined by ITU-T Recommendation 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 signaling in this Recommendation | International Standard. Further, it anticipates signaling needed for future specifications that include this Recommendation | International Standard as a normative reference. In addition to the codestream syntax defined in ITU-T Recommendation 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 ITU-T Recommendation 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 multi component transformation 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 offset

An extension that provides for variable scalar quantization offset 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 artifacts are masked by the image acting as a background signal. The main goal is to improve the image quality with low 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 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 components. is described in Annex F This extension can control the bandpass extent of wavelet subbands 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 Annex G. Annex G also describes wavelet transformation for a number of specific wavelet filters.

6.1.8 Single sample overlap discrete wavelet transformations

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

6.1.9 Multiple component transforms

An extension which provides for multiple component transformations is described in Annex I. 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 multi-dimensional wavelet transformation.

6.1.10 Regions of interest

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

6.1.11 Non-linear transformation

This Annex specifies two 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, gamma and look-up table (LUT) style non-linearities, may be employed by encoders prior to multiple component transformation and encoding to increase compression efficiency. A common usage of these transformations might be to perceptually flatten a scanner or sensor with a linear response, from 12 bits to 8 bits precision prior to compression.

6.1.12 File format

An extension of the file format is described in Annex L. 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 ITU-T T.800 | IS 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 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.

Annex A

Compressed data syntax, extension

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

All positive (unsigned) integers values of parameters are placed in the codestream as unsigned integers. All other (signed) integers are expressed in twos complement.

Some parameters have values defined with bits. In some cases there are bits, denoted by x, that do not correspond to any value. The codestream shall have zero value bits for those cases. The decoder shall ignore these bits.

A.1 Extended capabilities

The syntax in this Annex supports the extensions in this Recommendation | International Standard. This is achieved by the addition of parameter values to some marker segments in ITU-T T.800 | IS 15444-1 and the addition of new marker segments. These marker segments shall not be discarded by a decoder. Table A-1 shows the marker segments affected by this Recommendation | International Standard.

Table A-1 Syntax support for extensions

Extension	Extended ITU-T T.800 | IS 15444-1 marker segments	New marker segments
All extensions	SIZ	
Variable DC offset		DCO
Variable scalar quantization offset	QCD, QCC	
Trellis coded quantization	QCD, QCC	
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 transform		CBD, MCT, MCC, MIC
Non-linearity point transformation		CBD, NLT
Arbitrary shaped region of interest	RGN	

These marker segments conform to the same rules as the syntax in ITU-T T.800 | IS 15444-1 Annex A.

A.2 Extensions to ITU-T T.800 | IS 15444-1 marker segment parameters

This section describes the extensions to marker segments defined in ITU-T T.800 | IS 15444-1 Annex A.

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

The capability parameter Rsiz of ITU-T T.800 | IS 15444-1 denotes the necessity or usefulness of extensions of this Recommendation | International Standard for decoding the codestream. Table A-2 is used to define this parameter.

Table A-2 — Capability Rsiz parameter, extended

Value (bits) MSB LSB	Capability
0000 0000 0000 0000	Capabilities specified in ITU-T T.800 I IS 15444-1 only
1xxx xxxx xxxx xxxx	At least one of the extended capabilities specified in this Recommendation I International Standard is present
1xxx xxxx 0xxx xxx1	Variable DC offset capability is required to decode this codestream[a]
1xxx xxxx xxxx xx1x	Variable scalar quantization offset capability is required to decode this codestream[a]
1xxx xxxx xxxx x1xx	Trellis coded quantization capability is useful to decode this codestream[b]
1xxx xxxx xxxx 1xxx	Visual masking capability is useful to decode this codestream[b]
1xxx xxxx xxx1 xxxx	Single sample offset transformation capability is required to decode this codestream[a]
1xxx xxxx xx1x xxxx	Custom decomposition style capability is required to decode this codestream[a]
1xxx xxxx x1xx xxxx	Custom transformation kernel capability is required to decode this codestream[a]
1xxx xxxx 1xxx xxx0	Multiple component transformation capability is useful to decode this codestream[b]
1xxx xxx1 xxxx xxxx	Non-linearity point transformation capability is useful to decode this codestream[b]
1xxx xx1x xxxx xxxx	Arbitrary shaped region of interest capability 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. "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.

Geometric manipulation is enabled with two bits in the Csiz parameter.

Table A-3 Csiz parameters, extended

Values (bits) MSB LSB	Csiz parameter values
xx00 0000 0000 0001 xx11 1111 1111 1110	Number of components, 1 16 384
x0xx xxxx xxxx xxxx x1xx xxxx xxxx xxxx	Offset in the vertical dimension is 0 Offset in the vertical dimension is 1
0xxx xxxx xxxx xxxx 1xxx xxxx xxxx xxxx	Offset in the horizontal dimension is 0 Offset in the horizontal dimension is 1
	All other values reserved

A.2.2 Coding style (COD, COC), extended

If the Rsiz indicates that the arbitrary transformation kernels are used then Table A-6 replaces ITU-T T.800 | IS 15444-1 Table A-20.

If Rsiz indicates that the single sample offset transformation capability is necessary, then an extra 8 bit field is added to ITU-T T.800 | IS 15444-1 Table A-13 after the transformation field as shown in Table A-4. The SSO values are found in Table A-7.

If Rsiz indicates that the arbitrary decomposition styles are used then the maximum number of decomposition levels field definitions are found in Table A-5 rather than in ITU-T T.800 | IS 15444-1 Table A-13. This is shown in Table A-4.

Table A-4 — 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-5	Decomposition mapping and levels
Code-block width	8	ITU-T T.800 I IS 15444-1 Table A-18	Code-block width exponent offset value, *xcb*
Code-block height	8	ITU-T T.800 I IS 15444-1 Table A-18	Code-block height exponent offset value, *ycb*
Code-block style	8	ITU-T T.800 I IS 15444-1 Table A-19	Style of the code-block coding passes
Transform	8	Table A-6	Wavelet transformation used.
SSO overlap	16	Table A-7	SSO overlap values
Precinct size	variable	ITU-T T.800 I IS 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 subband. Each successive parameter corresponds to each successive resolution in order.

Table A-5 — Decomposition for the SPcod and SPcoc parameters, extended

Values (bits) MSB LSB	Decomposition type
0000 0000 — 0001 0000	Number of levels of wavelet decomposition, dyadic decomposition, N_L (Zero implies no transform.).
1000 0000— 1000 1111	If in the main header: Downsampling factor style index value (0 to 15). (See Annex A.3.3) If in tile-part header: Arbitrary decomposition style index value (0 to 15). (See Annex A.3.4)
	All other values reserved

Table A-6 — Transformation for the SPcod and SPcoc parameters, extended

Values (bits) MSB LSB	Transformation type
0000 0000	9-7 irreversible wavelet transform
0000 0001	5-3 reversible wavelet transform
1000 0xxx — 1111 1xxx	Arbitrary transformation kernel definition index value (0 — 15). Definitions are found in the appropriate ATK marker segment (see Annex A.3.5).
	All other values reserved

Table A-7 SSO parameters, extended

Values (bits) MSB LSB	SSO size (see Annex H)
0xxx xxxx xxxx xxxx 1xxx xxxx xxxx xxxx	SSO not used SSO used
x0xx xxxx xxxx xxxx x1xx xxxx xxxx xxxx	TSSO not used TSSO used
xxxx xxxx xxxx 0000 xxxx xxxx xxxx 1111	Cell width exponent value, $XC = 2^{value}$
xxxx xxxx 0000 xxxx xxxx xxxx 1111 xxxx	Cell height exponent value, $YC = 2^{value}$
	All other values reserved

A.2.3 Quantization (QCD, QCC), extended

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

Table A-8 replaces ITU-T T.800 | IS 15444-1 Table A-28 and Table A-9 replaces ITU-T T.800 | IS 15444-1 Table A-30.

Table A-8 — Quantization default values for the Sqcd and Sqcc parameters, extended

Values (bits) MSB LSB	Quantization style	SPqcx size (bits)	SPqcx usage
xxx0 0000	No quantization	8	ITU-T T.800 I IS 15444-1 Annex A
xxx0 0001	Scalar derived (values signalled for N_LLL subband only). Use ITU-T T.800 I IS 15444-1 Equation E.5.	16	ITU-T T.800 I IS 15444-1 Annex A
xxx0 0010	Scalar expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands.	16	ITU-T T.800 I IS 15444-1 Annex A
xxx0 0011	Variable offset and scalar derived (values signalled for N_LLL subband only). Use ITU-T T.800 I IS 15444-1 Equation E.5.	16	Table A-9
xxx0 0100	Variable offset derived and scalar expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands	16	Table A-10 then Table A-11
xxx0 0101	Variable offset and scalar expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands.	16	Table A-9
xxx0 0110	Trellis coded quantization derived (values signalled for N_LLL subband only). Use ITU-T T.800 I IS 15444-1 Equation E.5.	16	ITU-T T.800 I IS 15444-1 Annex A
xxx0 0111	Trellis coded quantization expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands.	16	ITU-T T.800 I IS 15444-1 Annex A
xxx0 1000	Variable offset and trellis coded quantization derived (values signalled for N_LLL subband only). Use ITU-T T.800 I IS 15444-1 Equation E.5.	16	Table A-9
xxx0 1001	Variable offset derived and trellis coded quantization expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands.	16	Table A-10 then Table A-11
xxx0 1010	Variable offset and trellis coded quantization expounded (values signalled for each subband). There are as many step sizes signalled as there are subbands.	16	Table A-9
000x xxxx — 111x xxxx	Number of guard bits 0 — 7		
	All other values reserved		

Table A-9 — Quantization values (irreversible transformation only), extended

Values (bits) MSB LSB	Quantization step size values
0000 0000 0000 0000 xxxx xxxx xxxx xxxx — 1111 1111 1111 1111 xxxx xxxx xxxx xxxx	Variable scalar quantization offset, num_nz_b, value -32 768 — 32 767 (see Equation C.1)
xxxx xxxx xxxx xxxx xxxx x000 0000 0000 — xxxx xxxx xxxx xxxx xxxx x111 1111 1111	Mantissa, μ_b, of the quantization step size value 0 — 2 047 (see ITU-T T.800 | IS 15444-1 Equation E.3)
xxxx xxxx xxxx xxxx 0000 0xxx xxxx xxxx — xxxx xxxx xxxx xxxx 1111 1xxx xxxx xxxx	Exponent, ε_b, of the quantization step size value 0 — 31 (see ITU-T T.800 | IS 15444-1 Equation E.3)

Table A-10 — SPqcd and SPqcc parameters (irreversible transformation only), extended

Values (bits) MSB LSB	Quantization step size values
0000 0000 0000 0000 — 1111 1111 1111 1111	First two bytes of data run are the variable scalar quantization offset, num_nz_b, value -32 768 — 32 767 (see Equation C.1)

Table A-11 — SPqcd and SPqcc parameters (irreversible transformation only), extended

Values (bits) MSB LSB	Quantization step size values (one for each subband)
xxxx x000 0000 0000 — xxxx x111 1111 1111	After first two bytes of data run are the mantissa, μ_b, of the quantization step size value 0 — 2 047 (see ITU-T T.800 | IS 15444-1 Equation E.3)
0000 0xxx xxxx xxxx — 1111 1xxx xxxx xxxx	After first two bytes of data run are the exponent, ε_b, of the quantization step size value 0 — 31 (see ITU-T T.800 | IS 15444-1 Equation E.3)

A.2.4 Region of interest (RGN), extended

If Rsiz indicates that an arbitrary region of interest is used (see Annex K) then the description of a coefficient shift and a mask are signalled in a modified RGN marker segment from ITU-T T.800 | IS 15444-1 Annex A. Table A-12 replaces ITU-T T.800 | IS 15444-1 Table A-25.

Table A-12 — Region-of-interest parameter values for the Srgn parameter

Values	ROI style (Srgn)	SPrgn usage
0	Implicit ROI (maximum shift)	ITU-T T.800 I IS 15444-1 Table A-26
1	Arbitrary region of interest, rectangle	Table A-13
2	Arbitrary region of interest, ellipse	Table A-13
	All other values reserved	

Table A-13 — Component index parameter values for the Carn parameter

Parameter	Size (bits)	Values	Components index parameter
Component	16	0 — 16 383 16 394 — 65 354 65 535	Specifies component to which these region of interest descriptions apply Reserved Region of interest descriptions apply to all components
Binary shift	8	0 — 255	Binary shifting of coefficients in the region of interest above the background.
Horizontal (left)	32	0 — $(2^{32}-1)$	Horizontal reference grid point from the origin of the first point.
Vertical (top)	32	0 — $(2^{32}-1)$	Vertical reference grid point from the origin of the first point.
Horizontal (right)	32	0 — $(2^{32}-1)$	Horizontal reference grid point from the origin of the second point.
Vertical (bottom)	32	0 — $(2^{32}-1)$	Vertical reference grid point from the origin of the second point.

A.3 Extended marker segments

Table A-14 lists the markers specified in this Recommendation | International Standard.

Table A-14 — 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	0xFF72	optional	optional
Component bit depth	CBD	0xFF78	optional	optional
Multiple component transformation definition	MCT	0xFF74	optional	optional
Multiple component collection	MCC	0xFF75	optional	optional
Multiple component intermediate collection	MIC	0xFF77	optional	optional
Non-linearity point transformation	NLT	0xFF76	optional	optional

a. Required means the marker or marker segment shall be in this header if this extension is used. Optional means it may be used in the header if this extension is used.

A.3.1 Variable DC offset (DCO)

Function: Describes the variable DC offset for every component.

Usage: 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 lessor marker segment.

Shall not be used with the multiple component transform.

Length: Variable depending on the number of components.

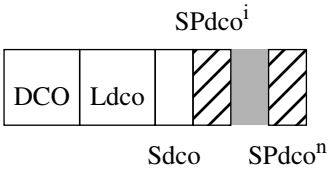

Figure A-1 Variable DC offset syntax

DCO: Marker code. Table A-15 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 ITU-T T.800 | IS 15444-1 Annex A.

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-15 — Variable DC offset parameter values

Parameter	Size (bits)	Values
DCO	16	0xFF70
Ldco	16	5 — 32 770
Sdco	8	Table A-16
SPdco^{i}	variable	Table A-16

Table A-16 — Variable DC offset parameter values for the Sdco parameter

Values (bits) MSB LSB	Matrix index, type, and parameter type
0000 0000	Offsets are 8 bit unsigned integers
0000 0001	Offsets are 16 bit signed integers
0000 0010	Offsets are 32 bit floating point (IEEE Std. 754-1985 R1990)
0000 0011	Offsets are 64 bit floating point (IEEE Std. 754-1985 R1990)
	All other values reserved

A.3.2 Visual masking (VMS)

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

Usage: 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 visual masking for every component in every tile. If present in the first tile-part header of a given tile, it describes the visual masking for every component in that tile only. When used in both the main header and the first tile-part header the VMS in the first tile part header overrides the main for that tile. Thus the order of precedence is the following:

Tile-part VMS > Main VMS

where the “greater than” sign, >, means that the greater overrides the lessor marker segment.

Length: Fixed at 7 bytes.

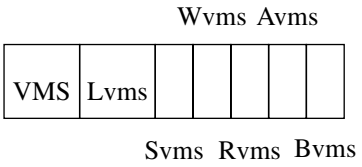

Figure A-2 — Visual masking syntax

VMS: Marker code. Table A-17 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.

Svms: Minimal resolution level and respect block boundaries flag.

Wvms:Window width variable, *win_width* (see Annex E.3).

Rvms: Bits retained variable, *bits_retained* (see Annex E.3).

Avms: Value of the numerator of the α parameter, $\alpha = Avms/128$ (see Annex E.3).

Bvms: Value of the numerator of the β parameter, $\beta = Bvms/128$ (see Annex E.3).

Table A-17 — Visual masking parameter values

Parameter	Size (bits)	Values
VMS	16	0xFF71
Lvms	16	7
Svms	8	Table A-27
Wvms	8	0 — 8
Rvms	8	0 — 255
Avms	8	0 — 255
Bvms	8	0 — 255

Table A-18 — Visual masking for the Svms parameters

Values (bits) MSB LSB	Visual masking parameters
x000 0000 — x001 0000	Minimum resolution level value, *minlevel* (0 — 32) (see Annex E.3)
0xxx xxxx 1xxx xxxx	Variable *respect_block_boundaries* = 0 (see Annex E.3) Variable *respect_block_boundaries* = 1 (see Annex E.3)
	All other values reserved

A.3.3 Downsampling factor styles (DFS)

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

Usage: Main header. Assigned to a component by an index in the main header COD or COC makers.

Length: Variable.

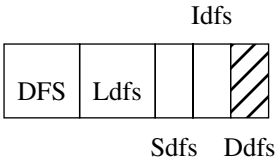

Figure A-3 — Downsampling factor styles syntax

DFS: Marker code. Table A-20 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:

$$Ldfs = 4 + \left\lceil \frac{Idfs}{4} \right\rceil . \qquad \text{A.2}$$

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-19 — Downsampling factor styles parameter values

Parameter	Size (bits)	Values
DFS	16	0xFF72
Ldfs	16	5 — 65 535
Sdfs	16	0 — 15
Idfs	8	0 — 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: Main and first tile-part header of a given tile. There may be up to 16 such markers 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.

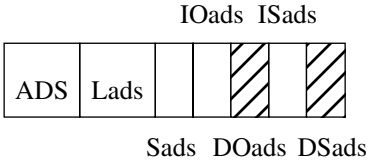

Figure A-4 — Arbitrary decomposition styles syntax

ADS: Marker code. Table A-20 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:

$$Lads = 5 + \left\lceil \frac{IOads + ISads}{4} \right\rceil . \qquad \text{A.3}$$

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-20 — Arbitrary decomposition styles parameter values

Parameter	Size (bits)	Values
ADS	16	0xFF73
Lads	16	
Sads	8	0 — 15
IOads	8	0 — 255
DOads	variable	String of elements
ISads	8	0 — 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: Main and first tile-part header of a given tile. May be up to 16 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.

Length: Variable.

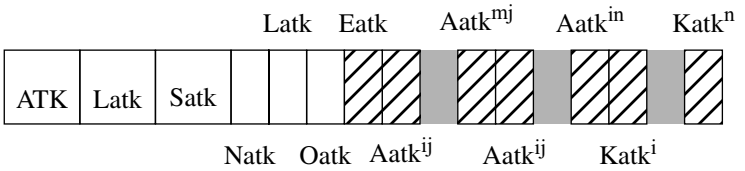

Figure A-5 — Arbitrary transformation default syntax

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:

$$Latk = \begin{cases} 8 + \text{Coeff\_Typ (Natk*Latk+Natk)} & \text{WT\_Typ} = 1 \\ 7 + \text{Coeff\_Typ (Natk*Latk+1)} & \text{WT\_Typ} = 0 \end{cases} \quad \text{A.4}$$

Satk: Index of the marker segment, size of the scaling factor and lifting step parameters *Coeff_Typ*, and wavelet filter category, *Filt_Cat*, wavelet transform type, *WT_Typ,* odd or even subsequence, m_0.

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

Latk: Number of lifting coefficients at lifting step s, L_s.

Oatk: Offset for lifting step s, off_s.

Eatk: Base two scaling exponent for lifting step s, ε_s for the reversible transform only (*WT_Typ*=1). Otherwise, it is the scaling factor, K, for the irreversible transform only (*WT_Typ*=0).

Aatk^{ij}:The ith lifting coefficient for the jth lifting step,$\alpha_{s,k}$. The index, i, ranges from i = 0 to Natk-1 and is the inner loop (present for all of j). The index, j, ranges from j = 0 to Latk-1 and is the outer loop (incremented after a full run of i).

Katk^{i}: The ith additive residue for lifting step, s. The index, i, ranges from i = 0 to Natk-1. Present for reversible transformations (*WT_Typ*=1)

Table A-21 — Arbitrary transformation parameter values

Parameter	Size (bits)	Values
ATK	16	0xFF72
Latk	16	9 — 65 535
Satk	16	Table A-22
Natk	8	0 — 255
Latk	8	0 — 255
Oatk	8	0 — 255
Eatk	8 8 16 32 64 128	*WT_Typ*=1 *WT_Typ*=0, *Coeff_Typ*=0 *WT_Typ*=0, *Coeff_Typ*=1 *WT_Typ*=0, *Coeff_Typ*=2 *WT_Typ*=0, *Coeff_Typ*=3 *WT_Typ*=0, *Coeff_Typ*=4
$Aatk^{ij}$	8 16 32 64 128	*Coeff_Typ*=0 *Coeff_Typ*=1 *Coeff_Typ*=2 *Coeff_Typ*=3 *Coeff_Typ*=4
$Katk^{i}$	0 8 16 32 64 128	*WT_Typ*=1 *Coeff_Typ*=0 *Coeff_Typ*=1 *Coeff_Typ*=2 *Coeff_Typ*=3 *Coeff_Typ*=4

Table A-22 — Arbitrary transformation values for the Satk parameter

Values (bits) MSB LSB	Index
xxxx xxxx xxxx 0000 — xxxx xxxx xxxx 1111	Index of this marker segment
xxxx xxxx x000 xxxx	Katk coefficient parameters 8-bit unsigned integer, *Coeff_Typ*=0
xxxx xxxx x001 xxxx	Katk coefficient parameters 16-bit signed integer, *Coeff_Typ*=1
xxxx xxxx x010 xxxx	Katk coefficient parameters 32-bit floating point (IEEE Std. 754-1985 R1990), *Coeff_Typ*=2
xxxx xxxx x011 xxxx	Katk coefficient parameters 64-bit floating point (IEEE Std. 754-1985 R1990), *Coeff_Typ*=3
xxxx xxxx x100 xxxx	Katk coefficient parameters 128-bit floating point (IEEE Std. 754-1985 R1990), *Coeff_Typ*=4
xxxx xxx0 0xxx xxxx	Arbitrary filters, *Filt_Cat*=0
xxxx xxx0 1xxx xxxx	WSS filters, *Filt_Cat*=1
xxxx xxx1 0xxx xxxx	HSS filters, *Filt_Cat*=2
xxxx xx0x xxxx xxxx	Irreversible filter, WT_typ=0
xxxx xx1x xxxx xxxx	Reversible filter, WT_typ=1
xxxx x0xx xxxx xxxx	Even-indexed subsequence, m_0=0
xxxx x1xx xxxx xxxx	Odd-indexed subsequence, m_0=1
	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: Main and first tile-part header of a given tile. The CBD marker segment is required if the multiple component transform processes are used. There may be as few as one CBD marker segment in the main header that covers all tiles or as many as one CBD marker segment for each tile. If the CBD marker segment is used for a specific tile, it shall appear in the first tile-part header for that tile and it overrides any CBD marker marker in the main header (if present).

The presence of a CBD marker segment in a codestream alters the procedures used to determine coefficient bit depths in the codestream and the interpretation of the SIZ marker. See Annex I for further details.

Length: Variable depending on the number of reconstructed image component bit depths signaled.

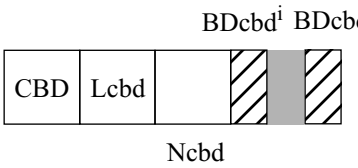

Figure A-6 Component bit depth definition syntax

CBD: Marker code. Table A-23 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:

$$Lmct = \begin{cases} 5 & \text{Ncbd} \geq 2^{15} \\ 4 + \text{Ncbd} & \text{Ncbd} < 2^{15} \end{cases} \qquad \text{A.5}$$

Ncbd: Number of component bit depths included in marker segment. Table A-24 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 MIC marker segments. Either one value is signalled for all components (see Table A-24) or an individual bit depth is given for each component.

Table A-23 Component bit depth definition parameter values

Parameter	Size (bits)	Values
CBD	16	0xFF78
Lcbd	16	4 16 388
Ncbd	16	Table A-24
$BDcbd^i$	8	Table A-25

Table A-24 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 0000 - x011 1111 1111 1111	Number of bit depths in marker segment - 1 (0 16 383)
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-25 Component bit depth definition values for the $BDcbd^i$ parameter

Values (bits) MSB LSB	Reconstructed image component bit depths
x000 0000 - x010 0101	Component sample bit depth = value + 1. From 1bit 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 matrices per marker segment. The type and index of the matrix defined in this marker distinguishes it from other MCT marker segments in a given header. This matrix can be assigned to a collection of components within the MCC marker segment.

Usage: Main and first tile-part header of a given tile. There can be many of these markers in either the main or first tile-part header. MCT marker segments in the first tile-part header apply only for that tile. MCT marker segments in the first tile-part header with the same type and matrix definition index of an MCT in the main header override the matrix definition in main header. The type and matrix definition index shall be unique within a given header.

To apply the transformation matrix included in an MCT marker segment, an MCC or MIC marker segment must exist that associates the MCT marker segment with a component collection. This association is made through the matrix definition index of the MCT marker segment (see Table A-27) and the $Tmcc^i$ (or $Tmic^i$) fields of MCC (or MIC) marker segments (see Table A-28 and Table A-35). If no such MCC or MIC marker segment exists, then the transformation matrix included in the MCT marker segment shall not be used in the decoding process.

Length: Variable depending on the size of the matrix.

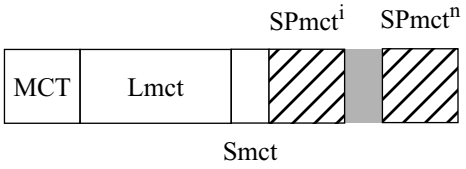

Figure A-7 Multiple component transformation definition syntax

MCT: Marker code. Table A-26 shows the size and parameter values for multiple component transformation definition marker segment.

Lmct: Length of marker segment in bytes (not including the marker). Due to the amount of data that may be contained within an MCT marker segment, the length field of this marker segment is *four* bytes. The value of this parameter is determined by the following equation:

$$Lmct = \begin{cases} 5 + \text{number\_of\_matrix\_elements} & \text{SPmct = 8 bits} \\ 5 + 2 \cdot \text{number\_of\_matrix\_elements} & \text{SPmct = 16 bits} \\ 5 + 4 \cdot \text{number\_of\_matrix\_elements} & \text{SPmct = 32 bits} \\ 5 + 8 \cdot \text{number\_of\_matrix\_elements} & \text{SPmct = 64 bits} \end{cases} \quad \text{A.6}$$

where number_of_matrix_elements is the number of SPmct parameters in this marker segment.

Smct: Multiple component transformation definition. Table A-27 shows the value for the Smct parameter.

$SPmct^i$:Parameters for the multiple component transformation definition. One parameter value for each element in the matrix. The elements are in raster scan order (top row from left to right, then next row from left to right, etc.). The number of elements in a row and the number of rows (elements in a column) are determined by the type of matrix and the number of the input and output components to which it is assigned.

Table A-26 Multiple component transformation definition parameter values

Parameter	Size (bits)	Values
MCT	16	0xFF74
Lmct	32	4 4 294 967 295
Smct	8	Table A-27
$SPmct^i$	variable	Table A-27

Table A-27 Multiple component transformation definition values for the Smct parameter

Values (bits) MSB LSB	Matrix index, type, and parameter type
xxxx 0001 xxxx 1111	Matrix definition index
xx00 xxxx	Decorrelation transformation matrix type
xx01 xxxx	Dependency transformation matrix type
xx10 xxxx	Decorrelation offset matrix type
xx11 xxxx	Dependency offset matrix type
00xx xxxx	Matrix elements are 8 bit integers
01xx xxxx	Matrix elements are 16 bit integers
10xx xxxx	Matrix elements are 32-bit floating point (IEEE Std. 754-1985 R1990)
11xx xxxx	Matrix elements are 64-bit floating point (IEEE Std. 754-1985 R1990)
	All other values reserved

A.3.8 Multiple component collection (MCC)

Function: Describes the collection of input spatially reconstructed components, the collection of output intermediate components, and the associated wavelets or matrices for a multiple component decorrelation transform. 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: Main and first tile-part header of a given tile. May be up to 256 MCC marker segments in the main header. If used in the main header and the default is indicated (Imcc = 0) it is used in all tiles without MCC marker segments. If used in the main header and the index is non-zero (Imcc ≠ 0), then it may be referred to by an MCC marker segment in a tile-part header.

No more than one marker segment in the tile-part header. If used in the tile-part header and the default is indicated (Imcc = 0) then this overrides the default in the main header, if any. If used in the tile-part header and the index is non-zero (Imcc ≠ 0) then the MCC marker segment in the main header with this index is used. In this case, there are no further parameters in this marker segment.

Length: Variable depending on the number of component collections.

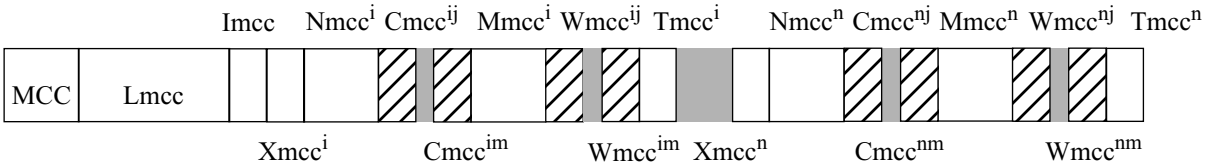

Figure A-8 Multiple component collection syntax

MCC: Marker code. Table A-28 shows the size and parameter values for multiple component collection marker segment.

Lmcc: Length of marker segment in bytes (not including the marker). The length field for the MCC marker segment is *four* bytes long:

$$Lmcc = 5 + 6 \cdot Q + \sum_{i=0}^{Q-1} Nmcc^i \Gamma^i + Mmcc^i \Psi^i$$

where,

$$\Gamma^i = \begin{cases} 1 & Nmcc^i < 2^{14} - 1 \\ 2 & Nmcc^i \geq 2^{14} \end{cases} \qquad \text{A.7}$$

$$\Psi^i = \begin{cases} 1 & Mmcc^i < 2^{14} - 1 \\ 2 & Mmcc^i \geq 2^{14} \end{cases}$$

Q = Number of component collections

Imcc: Index of this marker segment. A zero value implies the default for all tiles when used in the main header or overrides the default when used in the tile-part header. A non-zero value in the main header defines the index of this marker. A non-zero value in the tile-part header implies the assignment of the main header MCC marker segment with the same Imcc value to this tile. No other parameter values follow in this case.

Xmcc^i:Indicates type of decorrelation transform used for the ith component collection (wavelet or matrix-based). Defines the interpretation applied to Tmcc. If a wavelet transform is indicated, Xmcc also indicates if it is low-pass or high-pass first. Present only if used in the main header or when Imcc is zero in the first tile-part header.

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. Defines the number of column elements in assigned matrices (if applicable). Present only if used in the main header or when Imcc is zero in the first tile-part header.

Cmcc^{ij}:Spatially reconstructed component indices included the ith component collection. The number of indices in the ith component collection is Nmcc^{i}. Each index denotes a unique spatially reconstructed component. The order of the indices defines the ordering applied to spatially reconstructed components prior to application of the inverse decorrelation transform. Present only if used in the main header or when Imcc is zero in the first tile-part header.

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. Defines the number of rows in assigned matrices for matrix-based component collection decorrelation. If wavelet-based decorrelation is used, Mmcc^{i} must equal Nmcc^{i}. Present only if used in the main header or when Imcc is zero in the first tile-part header.

Wmcc^{ij}:Intermediate component indices included the ith output component collection. The number of indices in the ith component collection is Mmcc^{i}. All intermediate component indices in a given MCC marker segment must be unique across all collections in that MCC marker. Present only if used in the main header or when Imcc is zero in the first tile-part header.

Tmcc^{i}:For matrix-based component collection decorrelation, Tmcc^{i} assigns matrices 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.

For wavelet-based component collection decorrelation, Tmcc^{i} assigns a wavelet kernel defined in an ATK marker segment and arbitrary decomposition defined in an ADS marker segment to the ith component collection. An ATK or ADS marker segment with the proper index in the first tile-part header of a tile is used before an ATK or ADS marker segment with the proper index in the main header. Present only if used in the main header or when Imcc is zero in the first tile-part header.

Table A-28 Multiple component collection parameter values

Parameter	Size (bits)	Values
MCC	16	0xFF75
Lmcc	32	3 4 294 967 295
Imcc	8	Table A-29
Xmcc^{i}	8	Table A-30
Nmcc^{i}	16	Table A-31
Cmcc^{ij}	8 16	0 255 0 16 383
Mmcc^{i}	16	Table A-32
Wmcc^{ij}	8 16	0 255 0 16 383
Tmcc^{i}	8	Table A-33 and Table A-34

Table A-29 Multiple component collection values for the Imcc parameter

Values	Multiple component collection index parameter
0	Default MCC marker segment for all tiles when used in the main header. Explicit MCC marker segment for this tile when used in the tile-part header.
1 255	Index of the MCC maker segment when used in the main header. Index of MCC marker segment to be used for this tile when used in the tile-part header. (No other parameters are present in this case.)

Table A-30 Multiple component collection values for the Xmcc ^{i} parameter

Values (bits) MSB LSB	Coding Style
xxxx xx0x xxxx xx10 xxxx xx11	Component collection decorrelation transform is matrix-based Component collection decorrelation transform is wavelet-based, low-pass first Component collection decorrelation transform is wavelet-based, high-pass first

Table A-31 Multiple component collection values for the Nmcc ^{i} parameter

Values (bits) MSB LSB	Coding Style
00xx xxxx xxxx xxxx 01xx xxxx xxxx xxxx	Input component collection indices (Cmcc^{i}) are 8 bit integers Input component collection indices (Cmcc^{i}) are 16 bit integers
0x00 0000 0000 0000 0x11 1111 1111 1111	Number of input components in ith component collection (0 16 383)

Table A-32 Multiple component collection values for the Mmcc ^{i} parameter

Values (bits) MSB LSB	Coding Style
00xx xxxx xxxx xxxx 01xx xxxx xxxx xxxx	Output component collection indices (Wmcc^{i}) are 8 bit integers Output component collection indices (Wmcc^{i}) are 16 bit integers
0x00 0000 0000 0000 0x11 1111 1111 1111	Number of output components in ith component collection (0 16 383)

Table A-33 Multiple component collection values for the Tmcc ^{i} parameter (matrix-based)

Values (bits) MSB LSB	Coding style
xxxx 0000 xxxx 1111	Decorrelation transform matrix with the matrix definition index (value = 0 means no assigned matrix)
0000 xxxx 1111 xxxx	Decorrelation offset matrix with the matrix definition index (value = 0 means no assigned matrix)

Table A-34 Multiple component collection values for the Tmcc ^{i} parameter (wavelet-based)

Values (bits) MSB LSB	Coding style
xxxx 0000 xxxx 1111	Index of ATK marker segment containing wavelet kernel for component collection decorrelation
0000 xxxx 1111 xxxx	Index of ADS marker segment containing arbitrary wavelet decomposition for component collection decorrelation

A.3.9 Multiple component intermediate collection (MIC)

Function: Describes the collection of input intermediate components, the collection of output reconstructed image components, and the matrices for a multiple component dependency transform. 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: Main and first tile-part header of a given tile. May be up to 256 MIC marker segments in the main header. If used in the main header and the default is indicated (Imic = 0) it is used in all tiles without MIC marker segments. If used in the main header and the index is non-zero (Imic ≠ 0), then it may be referred to by an MIC marker segment in a tile-part header.

No more than one marker segment in the tile-part header. If used in the tile-part header and the default is indicated (Imic = 0) then this overrides the default in the main header, if any. If used in the tile-part header and the index is non-zero (Imic ≠ 0) then the MIC marker segment in the main header with this index is used. In this case, there are no further parameters in this marker segment.

Length: Variable depending on the number of component collections.

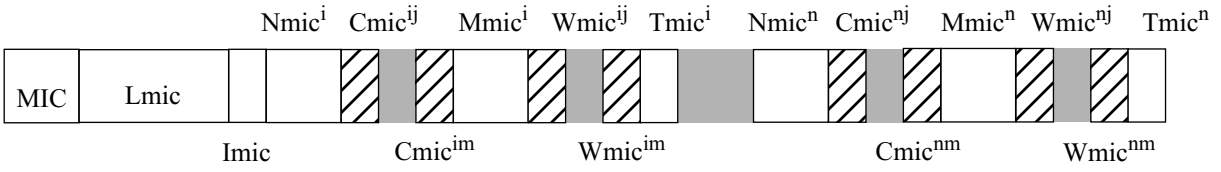

Figure A-9 Multiple component intermediate collection syntax

MIC: Marker code. Table A-35 shows the size and parameter values for multiple component intermediate collection marker segment.

Lmic: Length of marker segment in bytes (not including the marker). The length field for the MIC marker segment is *four* bytes long:

$$Lmic = 5 + 5 \cdot Q + \sum_{i=0}^{Q-1} Nmic^i \Gamma^i + Mmic^i \Psi^i$$

where,

$$\Gamma^i = \begin{cases} 1 & Nmic^i < 2^{14} - 1 \\ 2 & Nmic^i \geq 2^{14} \end{cases} \qquad \text{A.8}$$

$$\Psi^i = \begin{cases} 1 & Mmic^i < 2^{14} - 1 \\ 2 & Mmic^i \geq 2^{14} \end{cases}$$

Q = Number of component collections

Imic: Index of this marker segment. A zero value implies the default for all tiles when used in the main header or overrides the default when used in the tile-part header. A non-zero value in the main header defines the index of this marker. A non-zero value in the tile-part header implies the assignment of the main header MIC marker segment with the same Imcc value to this tile. No other parameter values follow in this case.

Nmic^i: Number of input components for the ith component collection. Defines the number of column elements in assigned matrices. Present only if used in the main header or when Imic is zero in the first tile-part header.

Cmic^{ij}: Intermediate component indices included the ith component collection. The number of indices in the ith component collection is Nmic^i. Each index refers to an output component index specified in the

currently active MCC. The order of the indices defines the ordering applied to intermediate and reconstructed image components prior to application of the inverse dependency transform. Present only if used in the main header or when Imic is zero in the first tile-part header.

$Mmic^i$:Number of output reconstructed image components for the ith component collection. Note for a given component collection, $Mmic^i = Nmic^i$ (dependency transform matrices are always square). Defines the number of rows in assigned matrices. Present only if used in the main header or when Imic is zero in the first tile-part header.

$Wmic^{ij}$:Reconstructed image component indices included the ith output component collection. The number of indices in the ith component collection is $Mmic^i$. All reconstructed image component indices in a given MIC marker segment must be unique across all collections in that MIC marker. Present only if used in the main header or when Imic is zero in the first tile-part header.

$Tmic^i$: Assigns matrices 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. Present only if used in the main header or when Imic is zero in the first tile-part header

Table A-35 Multiple component intermediate collection parameter values

Parameter	Size (bits)	Values
MIC	16	0xFF78
Lmic	32	3 4 294 967 295
Imic	8	Table A-36
$Nmic^i$	16	Table A-37
$Cmic^{ij}$	8 16	0 255 0 16 383
$Mmic^i$	16	Table A-38
$Wmic^{ij}$	8 16	0 255 0 16 383
$Tmic^i$	8	Table A-39

Table A-36 Multiple component intermediate collection values for the Imic parameter

Values	Multiple component intermediate collection index parameter
0	Default MIC marker segment for all tiles when used in the main header. Explicit MIC marker segment for this tile when used in the tile-part header.
1 255	Index of the MIC maker segment when used in the main header. Index of MIC marker segment to be used for this tile when used in the tile-part header. (No other parameters are present in this case.)

Table A-37 Multiple component intermediate collection values for the Nmic ^{i} parameter

Values (bits) MSB LSB	Coding style
00xx xxxx xxxx xxxx 01xx xxxx xxxx xxxx	Input component collection indices (Cmic^{i}) are 8 bit integers Input component collection indices (Cmic^{i}) are 16 bit integers
0x00 0000 0000 0000 0x11 1111 1111 1111	Number of input components in ith component collection (0 16 383)

Table A-38 Multiple component intermediate collection values for the Mmic ^{i} parameter

Values (bits) MSB LSB	Coding style
00xx xxxx xxxx xxxx 01xx xxxx xxxx xxxx	Input component collection indices (Wmic^{i}) are 8 bit integers Input component collection indices (Wmic^{i}) are 16 bit integers
0x00 0000 0000 0000 0x11 1111 1111 1111	Number of input components in ith component collection (0 16 383)

Table A-39 Multiple component intermediate collection values for the Tmic ^{i} parameter

Values (bits) MSB LSB	Coding style
xxxx 0000 xxxx 1111	Dependency transform matrix with the matrix definition index (value = 0 means no assigned matrix)
0000 xxxx 1111 xxxx	Dependency offset matrix with the matrix definition index (value = 0 means no assigned matrix)

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: 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 lessor marker segment.

Length: Variable depending on the value of Tnlt.

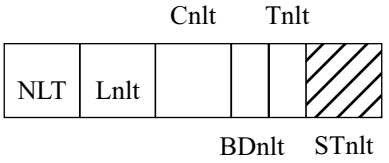

Figure A-10 Non-linearity point transformation syntax

NLT: Marker code. Table A-40 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:

$$
\begin{aligned}
Lnlt &= 6 + \begin{cases} 12 & Tnlt = 1 \\ 4 + 2 \cdot \Gamma_{Dval} + N_{points} \cdot \Psi_{Tval} & Tnlt = 2 \end{cases} \\
\Gamma_{Dval} &= \begin{cases} 1 & PDval \text{ MOD } 2^7 \in [1, 8] \\ 2 & PDval \text{ MOD } 2^7 \in [9, 16] \\ 4 & PDval \text{ MOD } 2^7 \in [17, 32] \end{cases} \\
\Psi_{Tval} &= \begin{cases} 1 & PTval \text{ MOD } 2^7 \in [1, 8] \\ 2 & PTval \text{ MOD } 2^7 \in [9, 16] \\ 4 & PTval \text{ MOD } 2^7 \in [17, 32] \end{cases}
\end{aligned}
\qquad \text{A.9}
$$

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-41 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-42 shows the values for the BDnlt parameter.

Tnlt: Non-linearity type. Table A-43 shows the value for the Scod parameter.

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

Table A-40 Non-linearity transformation parameter values

Parameter	Size (bits)	Values
NLT	16	0xFF76
Lnlt	16	12 65 535
Cnlt	16	Table A-41
BDnlt	8	Table A-42
Tnlt	8	Table A-43
STnlt	variable	Table A-43

Table A-41 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-42 — Decoded image component bit depth parameter values for the BDnlt parameter

Values (bits) MSB LSB	Decoded image component bit depth
x000 0000 - x010 0101	Component sample bit depth = value + 1. From 1bit 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-43 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-44
0000 0010	LUT-style non-linearity transformation	Table A-45
	All other values reserved	

Table A-44 Non-linearity transformation parameter values of the STnlt parameter (Tnlt = 1)

Parameters (in order)	Size (bits)	Values	Meaning of STnlt values
E	16	0 255.(0 255)/256	Non-linearity exponent (8-bit integer + 8-bit fraction)
S	16	0 255.(0 255)/256	Non-linearity toe slope (8-bit integer + 8-bit fraction)
T	16	0 255.(0 255)/256	Non-linearity threshold (8-bit integer + 8-bit fraction)
A	24	0 65535.(0 255)/256	Non-linearity continuity parameter A (16-bit integer + 8-bit fraction)
B	24	0 65535.(0 255)/256	Non-linearity continuity parameter B (16-bit integer + 8-bit fraction)

Table A-45 Non-linearity transformation parameter values of the STnlt parameter (Tnlt = 2)

Parameters (in order)	Size (bits)	Values	Meaning of STnlt values
Npoints	16	1 8 192	Number of points in the LUT-style non-linearity definition
PDval	8	0xxx xxxx 1xxx xxxx x000 0001 x010 0000	Dvalues are unsigned integers Dvalues are signed integers Dvalues precision in bits (1 32). This also implies how many bytes are used to express the Dvalues
Dvalues	8, 16, 32	variable	Minimum and maximum Dvalues for the LUT-style non-linearity. There are two Dvalues, D_{min} followed by D_{max}.
PTval	8	0xxx xxxx 1xxx xxxx x000 0001 x010 0000	Tvalues are unsigned integers Tvalues are signed integers Tvalues precision in bits (1 32). This also implies how many bytes are used to express the Tvalues
Tvalues	8, 16, 32	variable	Run of table values for the LUT-style non-linearity. There are Npoints Tvalues.

Annex B

Variable DC offset, extension

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 I. The capabilities of the codestream are defined by the SIZ marker segment parameter Rsiz (see Annex A.2.1).

This Annex specifies variable DC offset that converts the signed values resulting from the decoding process to the proper reconstructed samples.

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 ITU-T T.800 | IS 15444-1 Annex G.

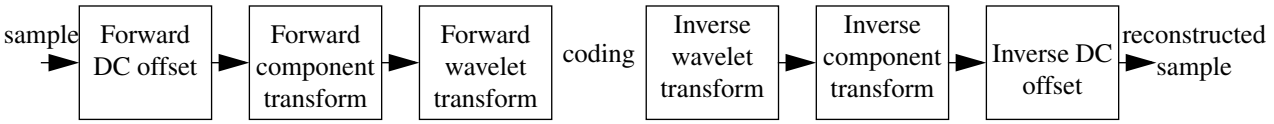

Figure B-1 — Placement of the DC offset with multiple component transformation

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

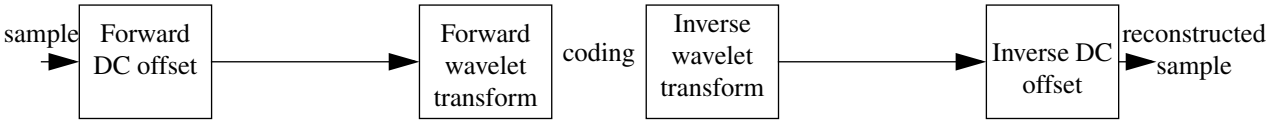

Figure B-2 — Placement of the DC offset without multiple component transformation

B.2 Inverse DC offset

If the DCO marker segment is present in the main or tile-part header (see Annex 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:

$$I'(x, y) = I(x, y) + O_i \qquad \text{B.1}$$

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

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

B.3 Forward DC offset (informative)

The generalized 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

$$O_i = 2^{Ssiz^i - 1} \quad \text{B.2}$$

where $Ssiz^i$ comes from ITU-T T.800 | IS 15444-1 Annex A. For signed data the default in ITU-T T.800 | IS 15444-1 Annex G is to have no offset ($O_i = 0$). Any other value must 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.

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

When a non-default offset is used, then care must 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 offset, extension

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1).

Variable scalar quantization offset extends the default scalar quantization described in ITU-T T.800 | IS 15444-1 Annex E to allow deadzones of variable width (up to twice the step size). Variable scalar quantization offset shall only be used with irreversible filters.

C.1 Variable scalar quantization offset

All terminology and variables described in ITU-T T.800 | IS 15444-1 Annex E remain the same for variable scalar quantization offset. 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 ITU-T T.800 | IS 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 must lie in the range $[-1,1)$[1].

When $nz = 0$ for all subbands, there is no need to transmit adjusted deadzone factors. However, if $nz_b \neq 0$ for at least one subband, then extended QCD and/or QCC marker segments shall appear in either the main header or the first tile-part header of a given tile (see Annex A.2.3). The value nz_b is represented as

$$nz_b = \frac{\text{num\_nz}_b}{2^{15}} \tag{C.1}$$

When an extended QCD or QCC marker segment appears, the adjustment factors are either signalled for each subband explicitly, or else signalled only for the LL subband. 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 Generalized scalar dequantization for irreversible filters

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

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

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

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

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

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

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

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

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

Annex D

Trellis coded quantization extensions

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex 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-2 shows the combination of these scalar quantizers into union quantizers, A_0 and A_1, along with the indices m_{A_i} available with each quantizer and the corresponding reconstruction levels $R_{A_i}(m_{A_i})$.

The eight state trellis that shows possible quantizer transitions is shown in Figure D-3 and is simply a directed graph 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 labeled as $N_{k,s}$, where k corresponds to the stage index for the node and s is the trellis state of the node.

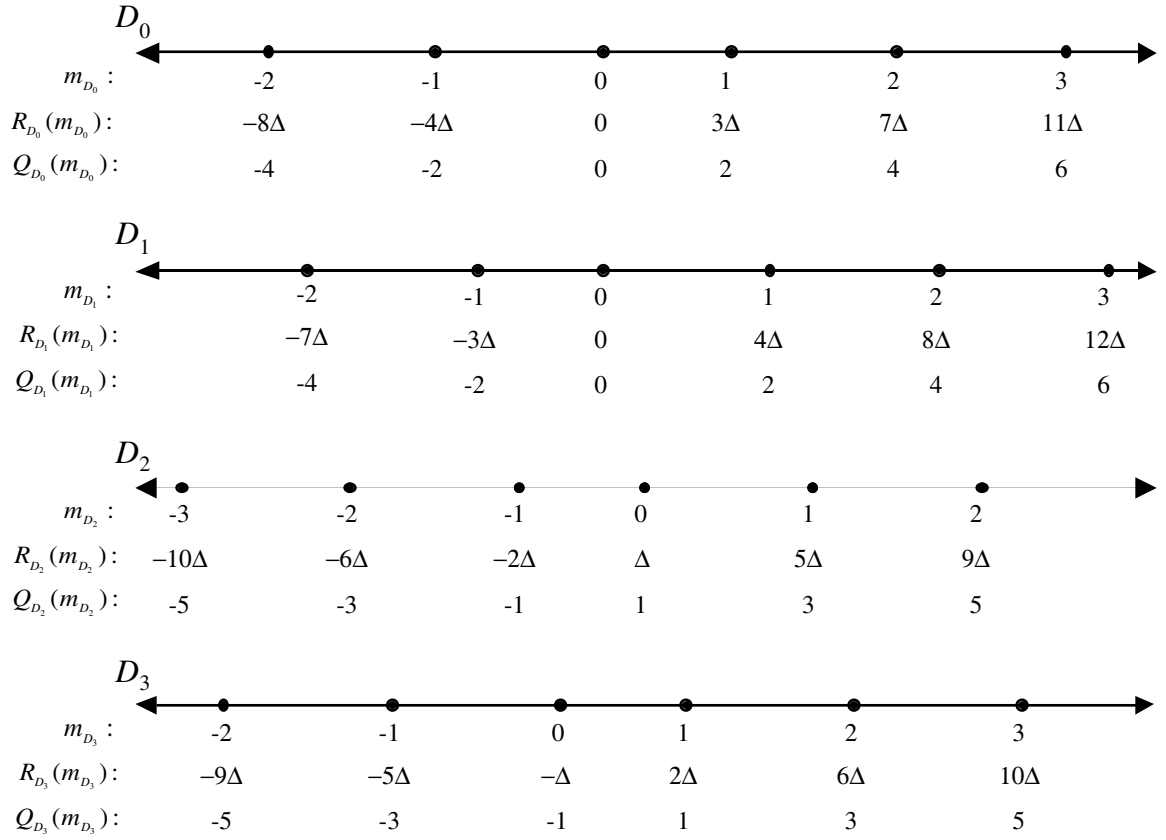

Figure D-1 — Scalar quantizers used for TCQ.

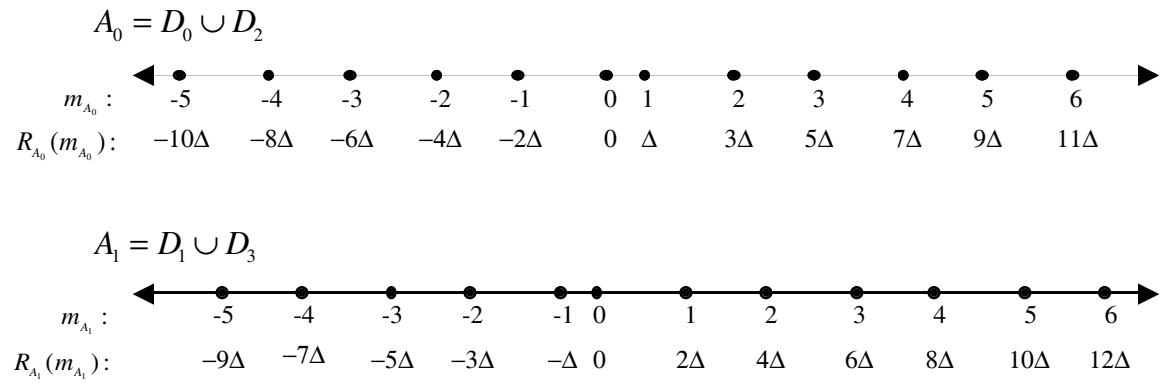

Figure D-2 — Union quantizers for TCQ.

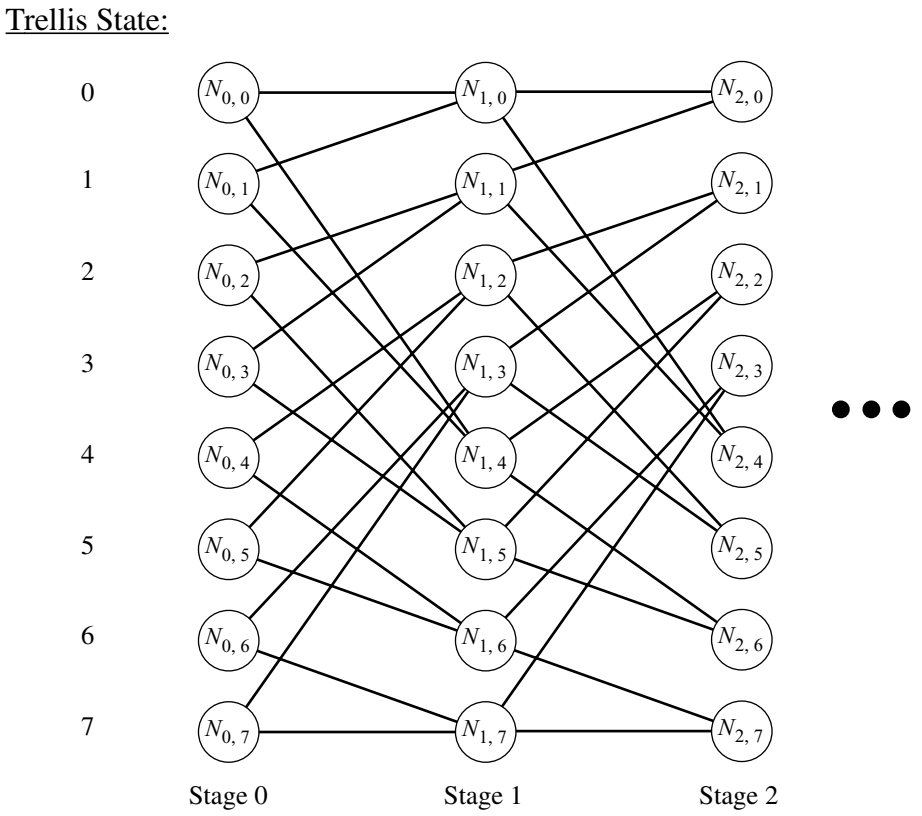

Figure D-3 — Trellis showing node indices.

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 ITU-T T.800 | IS 15444-1 Annex D.1 to form a sequence of coefficients processed by TCQ.

D.3 Forward quantization (informative)

All TCQ code-block sequences within a particular subband b use the same quantization step-size Δ_b. As with the scalar quantization option described in ITU-T T.800 | IS 15444-1 Annex 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 could be set according to ITU-T T.800 | IS 15444-1 Equation E.5. A recommended algorithm for choosing the set of Δ_b is that of Lagrangian Rate Allocation (LRA) defined later in this annex.

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:

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

where, as described in ITU-T T.800 | IS 15444-1 Annex E.1, R_b is the dynamic range of subband b. The ε_b and μ_b values are then used to set the SPqcd^i/SPqcc^i parameters in the QCD/QCC markers (see ITU-T T.800 | IS 15444-1 Annex A.6.4 and Annex 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 survivor distortion at node $N_{k,s}$; $B(N_{k,s})$ holds the parent node which yields lowest survivor distortion at node $N_{k,s}$; and $q_N(N_{k,s})$ 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, note that optimum progressive TCQ performance is achieved when the three entropy coder passes (see ITU-T T.800 | IS 15444-1 Annex D.3) for the last bit-plane of a given code block are concentrated into a single packet.

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

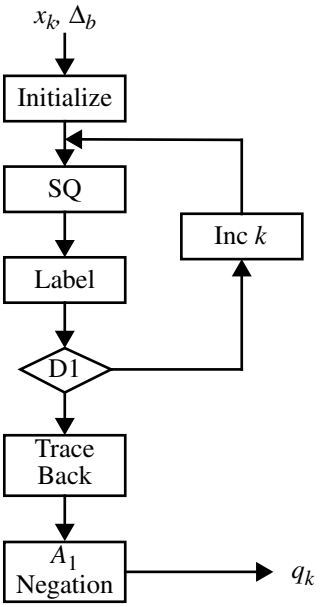

Figure D-4 — Forward TCQ processing

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., $$d_D(D_i) = \min_m \left\{ (x_k - R_{D_i}(m))^2 \right\},$$ with $q_D(D_i)$ set equal to the minimizing index value m.

Table D-2 — Description of functional blocks in Figure D-4

Block	Description
Label	For each stage s=0,..., 7: if $(d_N(N_0^P(N_{k+1,s})) + d_D(D_0^P(N_{k+1,s})) \le d_N(N_1^P(N_{k+1,s})) + d_D(D_1^P(N_{k+1,s})))$: $d_N(N_{k+1,s}) = d_N(N_0^P(N_{k+1,s})) + d_D(D_0^P(N_{k+1,s}))$ $q_N(N_{k+1,s}) = q_D(D_0^P(N_{k+1,s}))$ $B(N_{k+1,s}) = P_0^P(N_{k+1,s})$ else: $d_N(N_{k+1,s}) = d_N(N_1^P(N_{k+1,s})) + d_D(D_1^P(N_{k+1,s}))$ $q_N(N_{k+1,s}) = q_D(D_1^P(N_{k+1,s}))$ $B(N_{k+1,s}) = P_1^P(N_{k+1,s})$
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,..., 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_i Negation	Negate all ± 1 indices in union quantizer A_1.

D.4 Inverse quantization (normative)

Decoded TCQ quantized indices may be reconstructed to wavelet coefficients using either the recommended full inverse TCQ process or an approximate scalar dequantization. The full dequantization approach should not be used, however, when the cleanup pass of the last bit-plane for the current code block sequence is not fully decoded due to progressive decompression.

D.4.1 Full 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 parameters defined in the QCD or QCC markers (see ITU-T T.800 | IS 15444-1 Annex A.6.4 and Annex A.6.5.

The full dequantization process is outlined below in both Figure D-5 and Table D-3. Input to this process is the code block sequence of TCQ indices q_k and its output is the sequence x_k of reconstructed wavelet coefficients for the current code block.

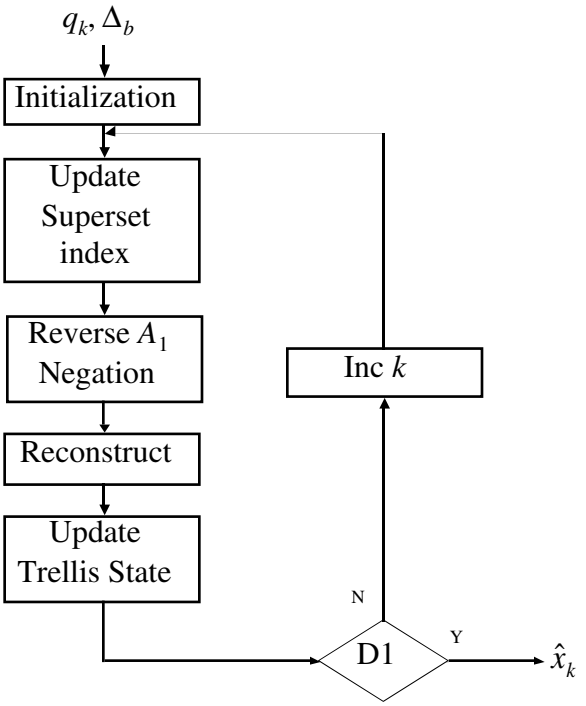

Figure D-5 — Full inverse processing for TCQ indices

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 $q_k = \pm 1$, reverse negation of current quantized index.
Reconstruct	Form reconstructed coefficient x_k $x_k = R_a(q_k)$ where $R_{A_i}(m)$ is defined in Figure D-2.
Update Trellis State	Update the current trellis state using Table D-5: $s = S(s,q_k)$.
D1	At end of 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

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

Current State s	q_k Odd/Even	$S(s,q_k)$
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,

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

The approximate dequantization process is outlined in both Figure D-6 and Table 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 ITU-T T.800 | IS 15444-1, and as such, does not require special processing to reasonably decode TCQ indices.

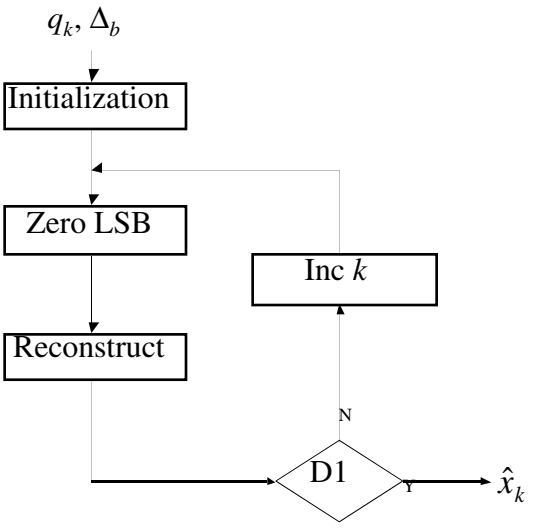

Figure D-6 — Approximate dequantization of TCQ indices

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.

Table D-6 — Description of functional blocks for Figure D-6

Block	Description
Reconstruct	Form reconstructed coefficient x_k $\hat{x}_k = \begin{cases} (q_k + r2^{M_b - N_b(k)}) \bullet \Delta_b, & q_k > 0 \\ (q_k - r2^{M_b - N_b(k)}) \bullet \Delta_b, & q_k < 0 \\ 0, & q_k = 0 \end{cases}$ where M_b, $N_b(k)$, and r are as defined for ITU-T T.800 | IS 15444-1 Equation E.8.
D1	At end of 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 subband step sizes. The first set is listed in Table D-7 and provides statistics for each subband, 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 Table D-8, Table D-9, Table D-10, and Table D-11. The parameters listed in these tables are based on experimentally obtained data for both TCQ and scalar quantization. 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 — subband statistics required for LRA

Statistic	Description
σ_b	Standard deviation for subband b.
β_b	Size weight for subband b.
Υ_b	Energy weight for subband b.
α_b	GGD parameter for subband 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 subband b and $\bar{x}_b$ is its mean.

The LRA process is defined in Figure D-7 and Table D-12. Note the outside loop in the block diagram of Figure D-7 which provides iterations through the compression process.

Table D-8 — R_b parameters for TCQ

	$\alpha_b = 0.5$	$\alpha_b = 0.75$	$\alpha_b = 1.0$	$\alpha_b = 1.5$	$\alpha_b = 2.0$
m_h	-1.660 964	-1.660 964	-1.660 964	-1.660 964	-1.660 964
a_h	-0.298 477 4	0.076 461 3	0.214 364 3	0.302 328 3	0.318 613 8
y_1	-2.3	3	-0.823 9	-0.522 9	0.221 8
m_l	0.056 304 7	0	-0.195 004	-0.334 404 7	-1.491 667
a_l	0.148 039 4	0	-0.124 035	-0.152 609 7	-0.331 146 0
y_2	-2.3	-2.208	-0.823 7	-0.522 9	-0.221 8
a	72.078 073	2.254 300	70.188 52	1.215 312 0	1.326 688
ζ	-0.093 750 7	0.046 034 3	0.0486 695 1	0.075 033	-0.003 975
p	2.832414x10+2	14.772 349	598.091 3	32.754 752	70.803 16
m_c	1.660 964	1.660 964	1.660 964	1.660 964	1.660 964

Table D-9 — Δ_b parameters for TCQ

	$\alpha_b = 0.5$	$\alpha_b = 0.75$	$\alpha_b = 1.0$	$\alpha_b = 1.5$	$\alpha_b = 2.0$
m_h'	0.5	0.5	0.5	0.5	0.5
a_h'	0.224 935	0.224 935	0.224 935	0.224 935	0.224 935
y_1'	-4	-3.398	-3	-3	-2.398
m_l'	0.027 569	0.023 69	0.031 087	0.021 283 5	0.047 27
a_l'	0.109 64	0.082 817	0.092 541	0.062 685	0.108 13
y_2'	-4	-3.398	-3	-3	-2.398
a'	293.33	3.26066x107	3.9963x102	8.1289x107	1.8067x103
ζ'	-1.506 7	-1.132 93	-0.875 9	-0.592 2	0.581 76
p'	8.5537x102	1.02504x108	1.5239x103	3.2113x108	6.8090x103
m_c'	0.211 731	0.302 78	0.390 313	0.651 786 5	15.378 33

Table D-10 — R_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	-1.660 964	-1.660 964	-1.660 964	-1.660 964	-1.660 964
a_h	-0.102 61	0.274 38	0.424 85	0.509 52	0.528 49

Table D-10 — R_b parameters for SQ

	$\alpha_b = 0.5$	$\alpha_b = 0.75$	$\alpha_b = 1.0$	$\alpha_b = 1.5$	$\alpha_b = 2.0$
y_1	-0.784 7	-1.386 3	-0.779 1	-0.648 2	-0.396 8
m_1	0.406 01	-0.094 20	-0.431 48	-0.703 42	-1.532 15
a_1	0.318 57	-0.130 59	-0.336 16	-0.455 93	-0.607 92
y_2	-0.419 1	-0.411 5	-0.343 5	-0.1282	-0.0599
a	0.591 15	0.493 44	0.385 87	0.150 084	0.035 03
ζ	0.172 08	0.081 93	0.042 38	0.022 67	-0.024 88
p	3.222 5	4.091 51	3.867 34	2.588 86	1.416 27
m_c	1.660 964	1.660 964	1.660 964	1.660 964	1.660 964

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'	0.5	0.5	0.5	0.5	0.5
a_h'	0.468 66	0.468 66	0.468 66	0.468 66	0.468 66
y_1'	-2.400 0	-1.937 6	-1.477 1	-1.456 9	-1.502 5
m_1'	0.049 82	0.083 73	0.064 25	0.043 91	0.036 37
a_1'	0.119 55	0.162 24	0.094 90	0.063 97	0.054 65
y_2'	-2.400 0	-1.937 6	-1.477 1	-1.456 9	-1.502 5
a'	3.474 58	10.733 49	4.890 79	3.805 14	5.363 54
ζ'	-0.635 78	-0.526 64	-0.600 12	-0.513 08	-0.028 00
p'	16.961 5	85.298 6	82.061 2	32.483 8	29.983 9
m_c'	0.185 09	0.146 91	0.283 73	0.405 10	2.085 1

NOTE — Special care should be taken during Δ_b computation in order to avoid infinite loops which repeat step sizes.

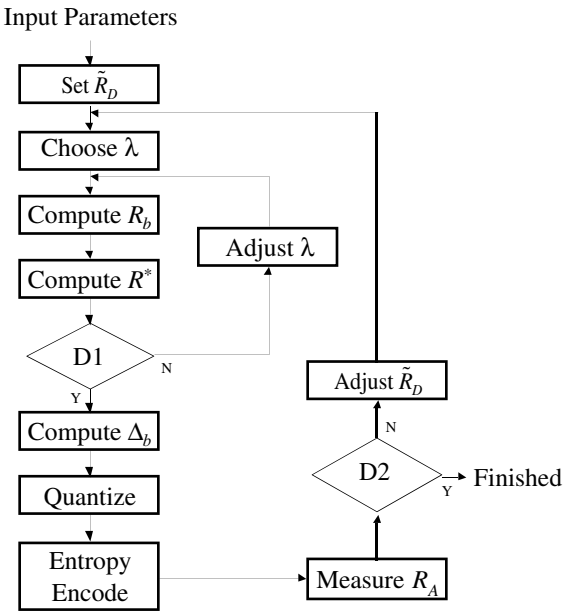

Figure D-7 — Lagrangian rate allocation

Table D-12 — Description of functional blocks in Figure D-7

Block	Description
Set R_D	Set R_D equal to desired bit-rate R_D.
Choose λ	Provide initial estimate for Lagrangian multiplier.
Compute R_b	$R_b = R_h + R_c + R_l$ for each subband b, where $R_h = m_h v + a_h$, $R_c = \begin{cases} m_c a \left\{ \left[1 + \left(\frac{v + a - \zeta}{a} \right)^p \right]^{\frac{1}{p}} - 1 \right\}, \text{for } v > y_2 \\ 0, \text{otherwise} \end{cases}$ $R_l = \begin{cases} m_l v + a_1, v > y_1 \\ 0, \text{otherwise} \end{cases}$ $v = \log\left(\frac{\lambda}{\Upsilon_b \sigma_b^2} \right)$
Compute R^*	$R^* = \sum_b \beta_b R_b$

Table D-12 — Description of functional blocks in Figure D-7

Block	Description
D1	R^* within tolerance of R_D ?
Adjust λ	Properly tune λ so R_D and R^* converge within tolerance.
Compute Δ_b	$\Delta_b = 10^{\Delta_h + \Delta_l + \Delta_c}$ for each subband b where $\Delta_h = m_n' v + a_h'$ $\Delta_c = \begin{cases} m_c' a' \left\{ \left[1 + \left(\frac{v + a' - \zeta'}{a'} \right)^{p'} \right]^{\frac{1}{p'}} - 1 \right\}, \text{for } v > y_2' \\ 0, \text{otherwise} \end{cases}$ $\Delta_l = \begin{cases} m_l' v + a_1', v > y_1' \\ 0, \text{otherwise} \end{cases}$ $v = \log\left(\frac{\lambda}{\Upsilon_b \sigma_b^2} \right)$
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 R_D	Properly tune R_D so R_A and R_D converge within tolerance.

Annex E

Visual masking, extensions

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1).

This Annex describes an option of the standard 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 artifacts are masked by the image acting as a background signal. The main goal is to improve the image quality with 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 transform stage and the quantization stage. At the decoder, the inverse non-linearity is applied after dequantization, prior to the inverse wavelet transform (see Figure E-1).

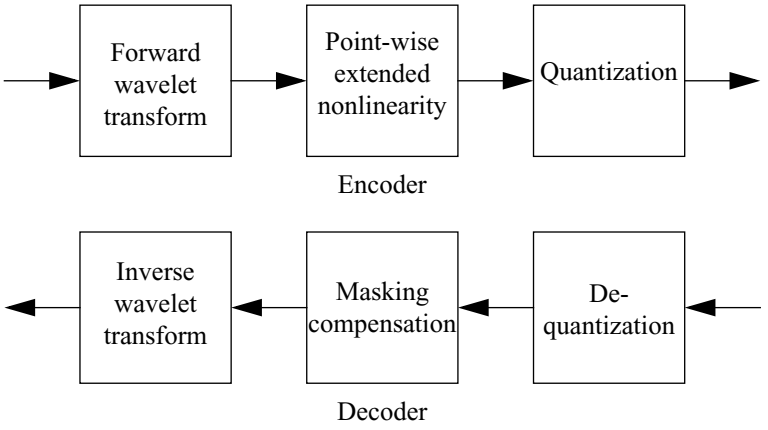

Figure E-1 System diagram for point-wise extended masking extension

E.2 Point-wise extended non-linearity

The extended masking option treats visual masking as a combination of two separate processes, i.e., self-contrast masking and neighborhood 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 that has been normalized to have a unit absolute DC gain, i.e.,

$$x_i \rightarrow y_i = sign(x_i)|x_i|^{\alpha} \qquad \text{E.1}$$

Equation E.1 is applied to all subbands except the LL subband using same value for α. The parameter α assumes a value between 0 and 1. A typical value of α is 0,7.

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 neighboring signals. There is some masking effect contributed from spatially neighboring signals due to the phase uncertainty, receptive field sizes, as well as possible longer range effects. To further exploit this neighborhood masking effect, the second step normalizes y_i by a neighborhood masking weighting factor w_i which is a function of the amplitudes of the neighboring signals, i.e.,

$$y_i \rightarrow z_i = \frac{y_i}{w_i} = \frac{sign(x_i)|x_i|^{\alpha}}{g(N_i(\{x_k\}))} \qquad \text{E.2}$$

where w_i is a function $g(.)$ of the neighboring signals denoted in a vector form as $N_i(\{x_k\})$, i.e., $w_i = g(N_i\{x_k\})$. 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.

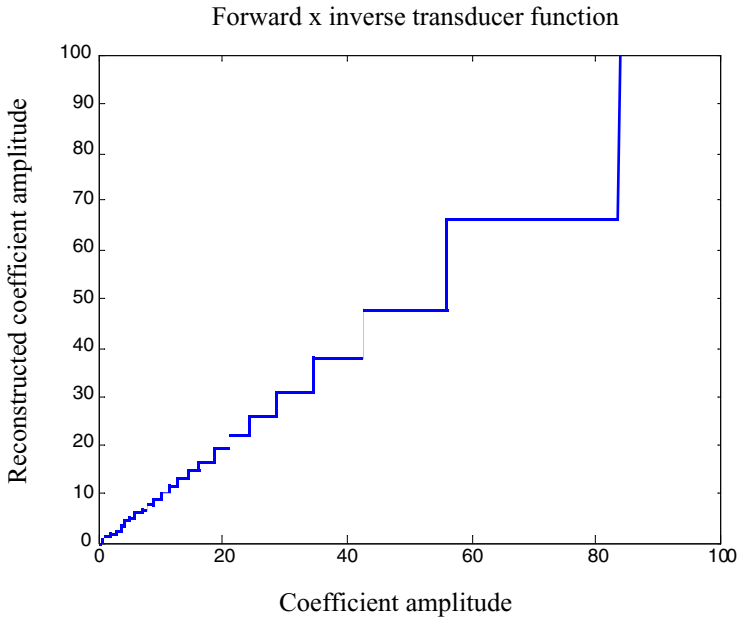

Figure E-2 Nonuniform quantization for self-contrast masking

Figure E-1 shows the system diagram for the point-wise extended masking option. To make sure the inverse process is feasible, quantized version of the neighboring coefficients that are also available at the decoder should be used, and the neighborhood has to be causal in the sense that each coefficient x_k in this neighborhood has to be recovered prior to the current x_i so that the decoder can perform exactly the same operation to reconstruct w_i. The Recommendation | International Standard adopts the following extended non-linearity

$$z_i = \frac{sign(x_i)|x_i|^{\alpha}}{1 + \left(a \sum_{k \in neighborhood} |\hat{x}_k|^{\beta} \right) / |\phi_i|} \qquad \text{E.3}$$

where $|\phi_i|$ denotes the size of the *neighborhood*, a is a normalization factor with a constant value of (10 000/ $2^{component\_bit\_depth-1}$)$^{\beta}$, and *component_bit_depth* denotes the bit depth of the component, $\hat{x_k}$ denotes the quantized neighboring coefficients, and the *neighborhood* contains coefficients in the same band that lie within an N x N window centered at the current coefficient and also appear earlier than the current coefficient in the raster scan order (see Figure E-3 for an example). The *neighborhood* does not include the current coefficient itself in order to allow an explicit solution for the inverse process. The causal *neighborhood* also respects code-block boundaries when a *respect_block_boundaries* switch is selected at the encoder. This switch should cause the neighborhood masking weighting factor not to include coefficients outside of the current code-block. Also, this switch must be transmitted to the decoder to tell it exactly how the *neighborhood* is formed.

N = 5, $|\phi_i|$ = 12, * current coefficient, x are the causal surrounding coefficients.

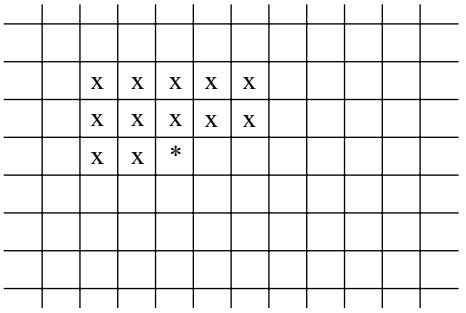

Figure E-3 Causal neighborhood

NOTE When the switch is on, it allows parallel implementation and restricts error propagation, but it may sacrifice some performance.

The parameter β also assumes a value between 0 and 1, and, together with N, are used to control the degree of neighborhood masking. The parameters β and N play important roles in differentiating coefficients around simple edge from those in the complex area. The parameter N controls the degree of averaging; β controls the influence of the amplitude of each neighboring 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.

At the decoder the inverse extended non-linearity follows

$$x_i = sign(z_i)\left[|z_i|\left(1+\left(a \sum_{k \in neighborhood} |\hat{x_k}|^{\beta}\right)/|\phi_i|\right)\right]^{1/\alpha} \qquad \text{E.4}$$

where z_i is the inverse quantized coefficient value.

Quantized neighboring coefficients will be used at 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 may just use the final quantized neighboring coefficients to calculate the neighborhood masking factor.

For embedded coding the encoder can not 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 neighboring coefficients to calculate the masking weighting factor w_i at the encoder and the decoder. After a neighboring coefficient z_i is quantized, only the *bits_retained* most significant bits of the quantization index will be retained (the rest of the bits are replaced with 0). This modified quantization index is then converted back to the x domain by Equation E.4 and is used for calculating w_i. 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 neighboring 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 neighboring coefficients is used.

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

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.

E.3 Bit stream syntax

The point-wise extended masking works with both scalar quantization and TCQ (see Annex D). A set of parameters that are encoded into the bit stream are α, β, *bits_retained* (the number of most significant bits to be retained for obtaining quantized neighboring coefficients), *win_width* (half of the causal neighborhood 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. Each of these parameters is encoded using 8 bits into the bit stream and normalized by 128. The switch *respect_block_boundaries* is also encoded into the bit stream (see Annex A.3.2).

The extended masking option can be combined with visual weighting to further improve the visual quality. Usually, to take advantage of the frequency sensitivity property of the human visual systems, the transform coefficients have to be multiplied by the corresponding CSF weights before they are subject to the transducer function. However, in the implementation, these two features in fact can be de-coupled, i.e., the frequency weights (designed for x domain) need to be altered to compensate for the non-linearity (power function with an exponent of α), prior to its use in the z domain for each subband.

Annex F

Arbitrary decomposition of tile-components, extensions

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1).

The extension described in this Annex specifies options available for forming wavelet subband decompositions. The notational conventions are first introduced followed by updates to various equations, text, decompositions and procedures from ITU-T T.800 | IS 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 subbands

This Recommendation | International Standard provides four tiers of detail for specifying two-dimensional bandpass signals (called subbands) at various spatial resolutions. Each tier provides more information in defining finer detail in the subband 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 subband decompositions is the number of wavelet decomposition levels, N_L. This value is signaled for each tile-component in the COD or COC markers as specified in Table A-15 of ITU-T T.800 | IS 15444-1. As with that Recommendation | International Standard, decomposition level indices are 1 for highest resolution subbands and N_L for the lowest resolution subband. Resolution indices, on the other hand, are labeled with value zero for the lowest resolution and N_L for the highest resolution. A value of zero for N_L indicates no wavelet transform for the tile-component.

F.1.2 Tier 2: Resolution formation

The various spatial resolutions are obtained through joint/dis-joint 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 ITU-T T.800 | IS 15444-1, the orientation of each spatial resolution (or subband) 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 XL. The signalling required to specify resolution formation is achieved via the COD, COC, and DFS marker segments as described in Annex F.2.5.

F.1.3 Tier 3: Sub-level decompositions

Wavelet subbands resulting from the first two tiers of wavelet processing may be further decomposed into new subbands 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 markers for signaling the maximum number of sub-levels, $\theta(lev)$, for each decomposition level is described in Annex F.2.

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

Not all subbands must be decomposed to the maximum sub-level depth. As a result, sets of subbands with non-uniform size at the same decomposition level may be produced. The look-up-tables (LUTs) *S*() and *J*() defined in Annex F.2 show how information in the ADS markers is used to signal the varying sub-level depth throughout complete wavelet decompositions.

Subbands can also be split disjointly in the horizontal and vertical directions, thus allowing subbands with subsampling factors from the original image that differ in the horizontal and vertical directions. As a result, subbands may be further decomposed into three separate sets of new subbands, 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 ITU-T T.800 | IS 15444-1. The second set yields only LX and HX orientations that result from just horizontal splits of a subband. 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, subbands 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 Annex 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.

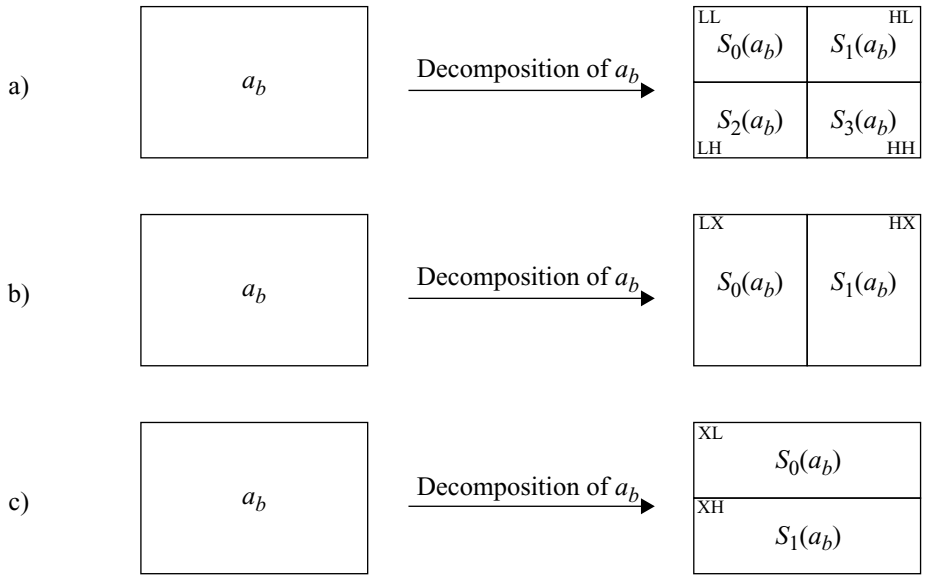

Figure F-1 Possible splits of subbands.

F.1.5 Complete subband notation

A colon separated notation is used to label the index *b* of subband a_b for the four tiered wavelet decomposition approach defined in this Annex. This notation (which is a simple extension of that given in ITU-T T.800 | IS 15444-1) begins with an index *lev* corresponding to the decomposition level of the subband, followed by the two-letter orientation for the first sub-level decomposition. For subbands 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 subbands with greater than two sub-levels.

F.1.6 HorOrient, VerOrient and PrimeOrient subband operators

To aid definitions of procedures in this Annex, the operators HorOrient( ) and VerOrient( ) are used to refer to the orientation of the deepest sub-level for any subband. The operator PrimeOrient( ) is used to refer to the orientation of the highest sub-level subband which eventually spawns into subband a_b within the same decomposition level.

F.2 Equation, text and decomposition updates

The arbitrary decomposition capability introduced in this Annex impacts several sections of ITU-T T.800 | IS 15444-1 outside of the wavelet transform. These affected sections are specified below along with updates to allow conformance with this Annex.

F.2.1 Updates to N_LLL references

General references to subband N_LLL are made throughout ITU-T T.800 | IS 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 subbands.

F.2.2 Context updates

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

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

Subbands with primary orientation of LL, LH, LX, XL, or XH			Subbands with primary orientation of HL and HX			Subbands 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. Note that 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 ITU.T Rec. T.800 | IS 15444-1 Equation B.14

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

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

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

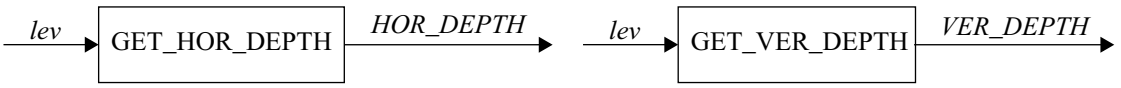

Figure F-2 Parameters for the GET_HOR_DEPTH and GET_ VER_DEPTH procedures.

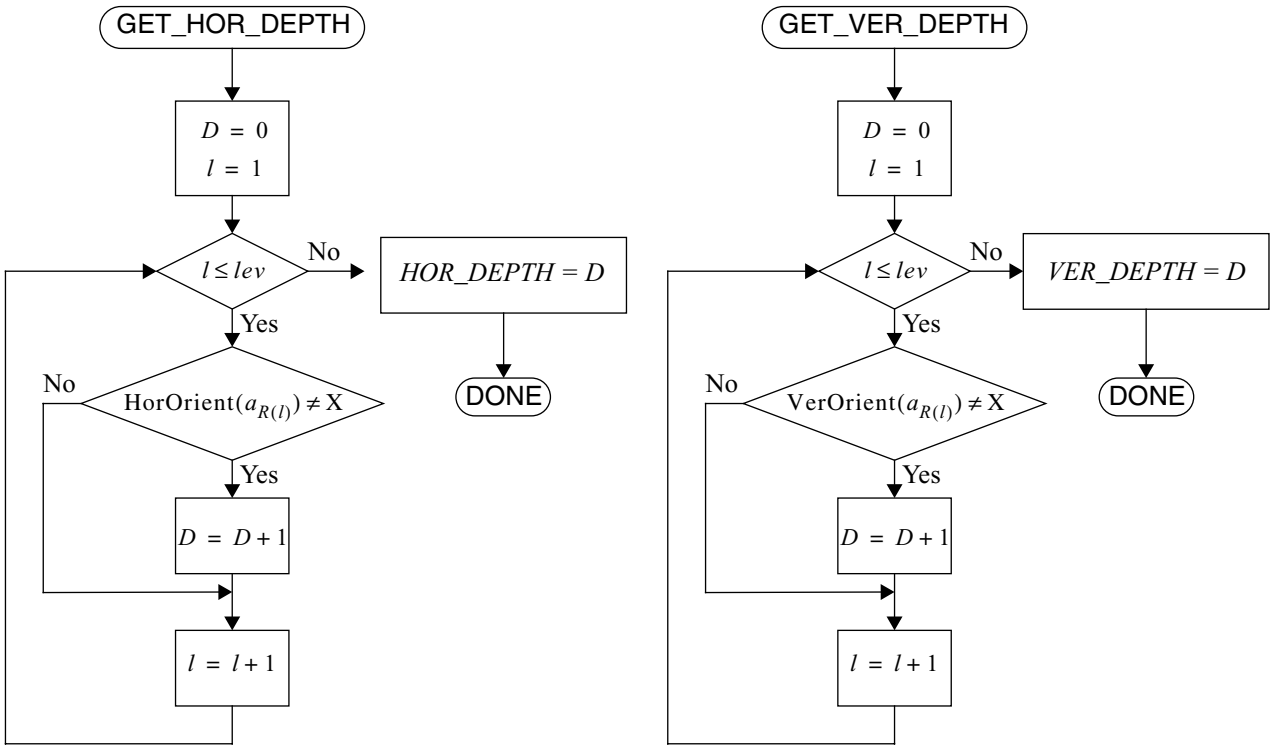

Figure F-3 The GET_HOR_DEPTH and GET_VER_DEPTH procedures.

F.2.4 Remaining updates

Equations B.15 (defining subband dimensions tbx_i and tby_i), E.4 (defining subband gains $gain_b$) and E.5 (defining the number of subband decomposition levels, n_b) from ITU-T T.800 | IS 15444-1 as well as the order, $O()$, of compressed subband data inside packet bitstreams and the dimensions *xpb* and *ypb* for precincts in each subband are defined for this Annex by the front end procedure SET_SUBBAND_INFO shown in Figure F-4 and Figure F-5 along with the recursive procedure RECUR_INFO defined by both Figure F-6 and Figure F-7. The SET_SUBBAND_INFO procedure first calls the procedures INIT_θ and INIT_S_R (defined in Annex 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 ADS markers segments (see Annex A.3.3 and Annex F.2.5). The B.15 updates also refer to Table F-2 based on the orientations of various subbands. 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 subband 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 signaled through COD and COC markers for each resolution *r* as described in Annex B.6 in ITU-T T.800 | IS 15444-1. To avoid *xpb* and *ypb* values less than one, both of these *rPPx* and *rPPy* values must also be greater than or equal to $\theta(N_L\text{-}r)$ 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.

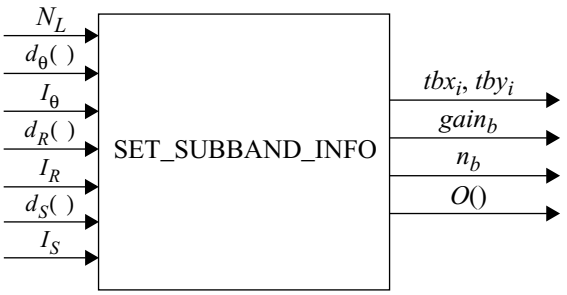

Figure F-4 Parameters for the SET_SUBBAND_INFO procedure.

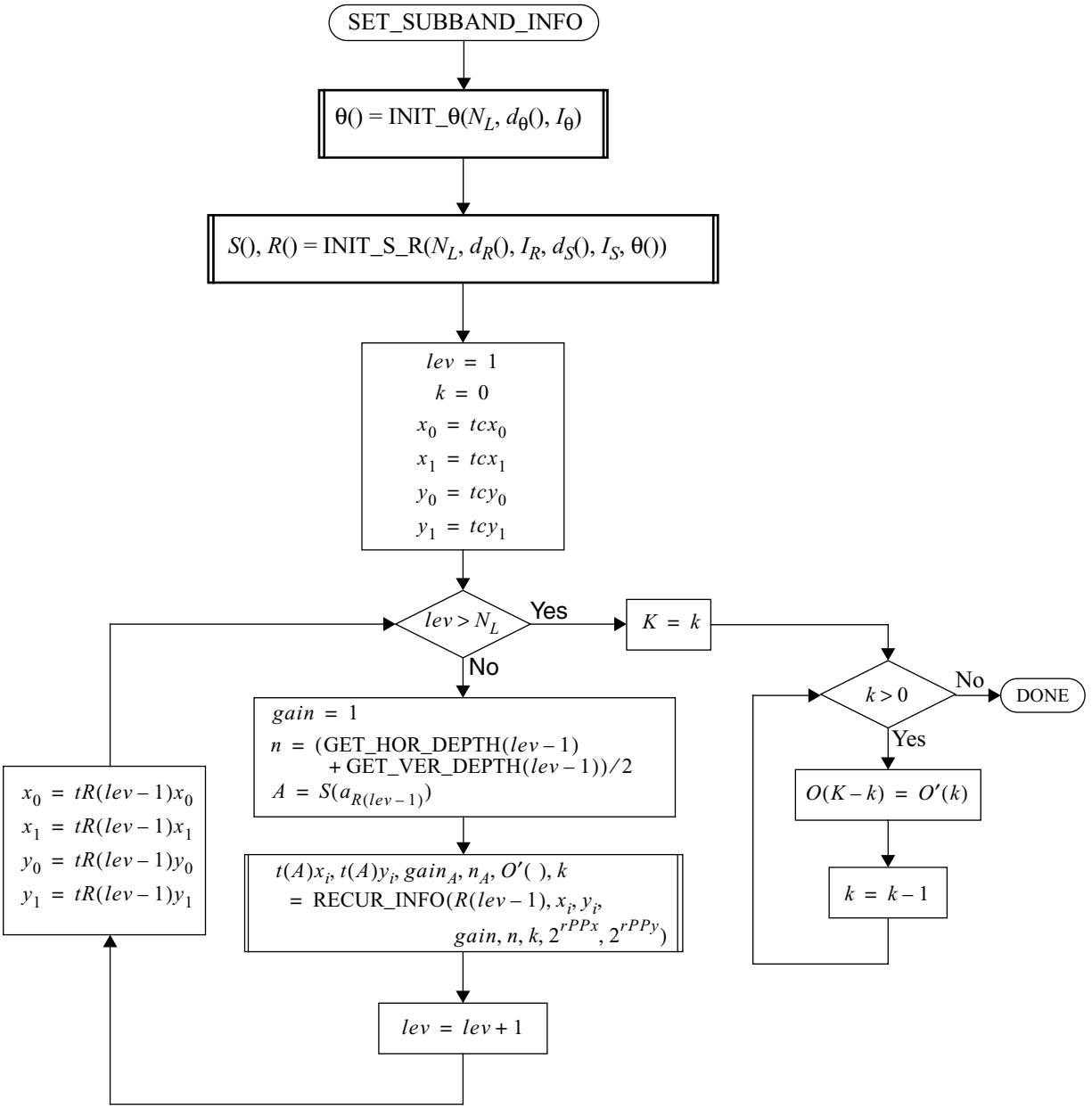

Figure F-5 The SET_SUBBAND_INFO procedure.

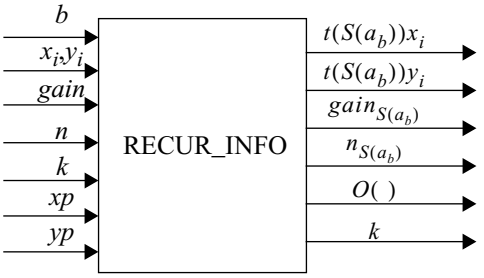

Figure F-6 Parameters for the RECUR_INFO procedure.

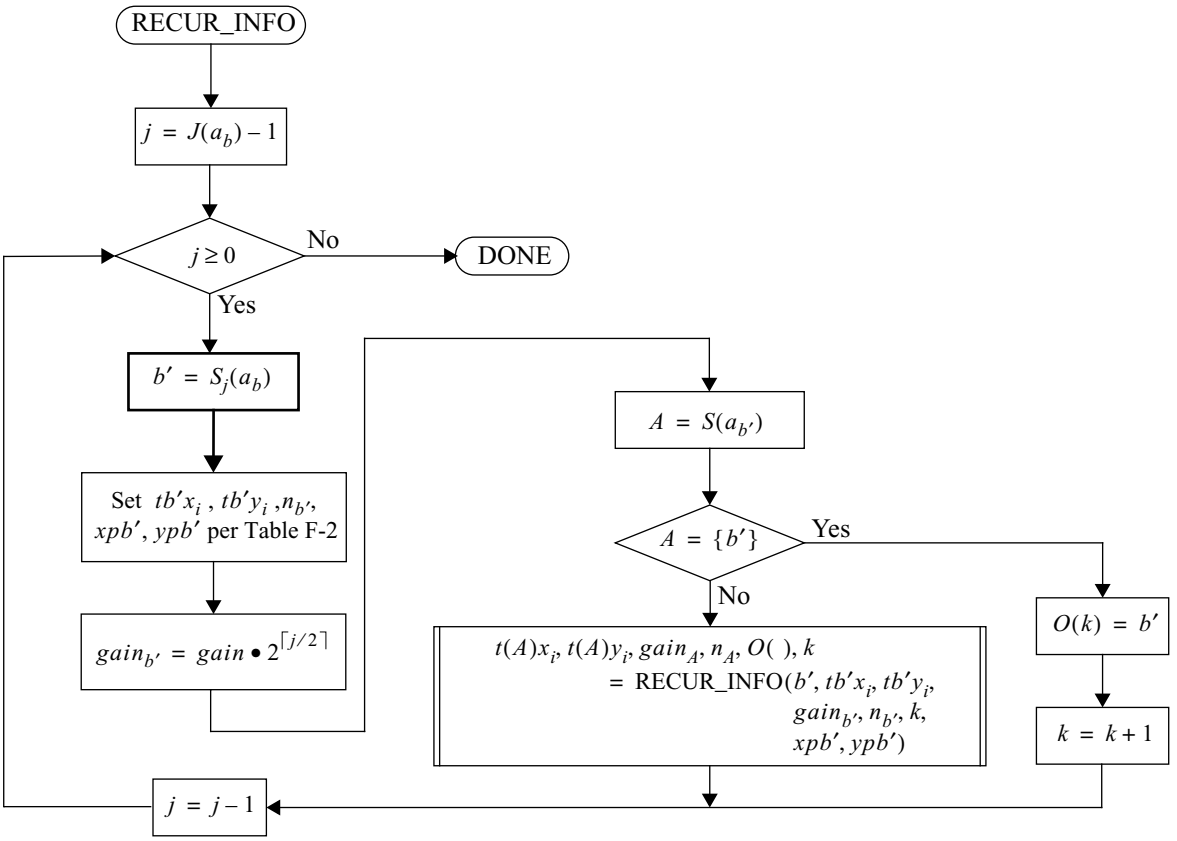

Figure F-7 The RECUR_INFO procedure.

Table F-2 Quantities for subband 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	$\lfloor x_i/2 \rfloor$	$\lceil y_i/2 \rceil$	$n+1$	$xp/2$	$yp/2$
L	H	$\lceil x_i/2 \rceil$	$\lfloor y_i/2 \rfloor$	$n+1$	$xp/2$	$yp/2$
H	H	$\lfloor x_i/2 \rfloor$	$\lfloor y_i/2 \rfloor$	$n+1$	$xp/2$	$yp/2$
L	X	$\lceil x_i/2 \rceil$	y_i	$n+1/2$	$xp/2$	yp
H	X	$\lfloor x_i/2 \rfloor$	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	$\lfloor y_i/2 \rfloor$	$n+1/2$	xp	$yp/2$

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 signaled in ADS markers (see Annex A.3.4). The first array, $d_\theta(i)$, i=0, , I_θ-1, is defined by the DOads and IOads ADS marker segments and is used by the INIT_θ procedure to determine the maximum number of sub-levels, $\theta(lev)$, 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 is not defined for the current tile-component, the length I_θ is set to zero.

NOTE The values of $d_\theta(i)$ used to set all $\theta(lev)$ 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, , I_R-1, is defined by COD, COC, and DFS marker segments (see Annex A.3.3) and specifies the dimensionality of each resolution level. The third array, $d_S(i)$, i=0, , I_S-1, is defined by the DSads and ISads ADS marker segments (see Annex A.3.3) 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. The I/O structure for these procedures are depicted in Figure F-10 and Figure F-12 and the corresponding algorithmic structures are defined in Figure F-11, Figure F-13, Table F-3 and Table F-4. As with I_θ, if the ADS marker is not defined for the current tile-component, I_R and I_S are 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 subbands in the wavelet decomposition. The first purpose of these procedures is to determine the LUT $S(a_b)$ which defines how a subband a_b decomposes into other subbands. This LUT is defined so that $S(a_b)$ equals the set of subband indices for decomposed subbands from subband a_b. The length LUT $J(a_b)$ is also defined by these routines to be the number of subbands which decompose from subband 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 subband index of the j-th subband 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 subband a_b equals the maximum, $\theta(lev)$, allowed for its level. Finally, the notation $a_{S(a_b)}$ is used to denote the set $\{a_{S_0(a_b)}, \ldots, a_{S_{J(a_b)-1}(a_b)}\}$.

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 subband orientation which results from the first sub-level of wavelet processing for a decomposition level. The corresponding subband is taken as the resolution at decomposition level *lev*. That is, resolution N_L-l*ev* is given by $a_{R(lev)}$.

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.

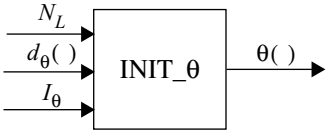

Figure F-8 Parameters for the INIT_ θ procedure.

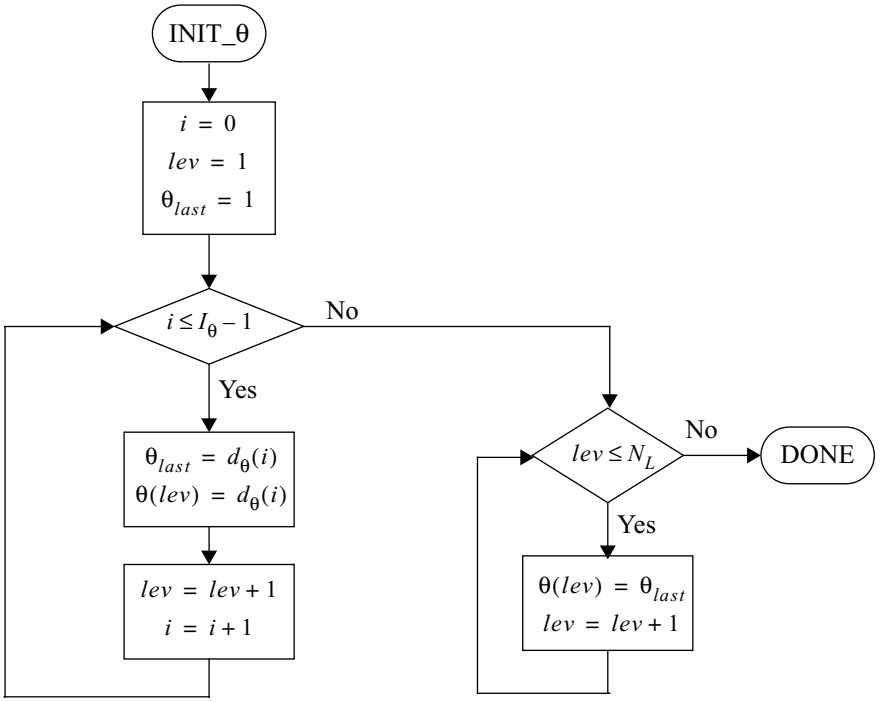

Figure F-9 Procedure for setting maximum number of sub-levels, $\theta(lev)$

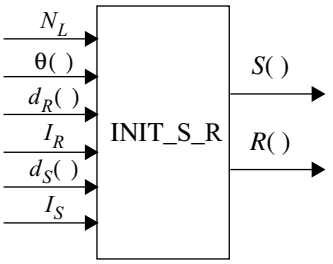

Figure F-10 Parameters for the INIT_S_R procedure.

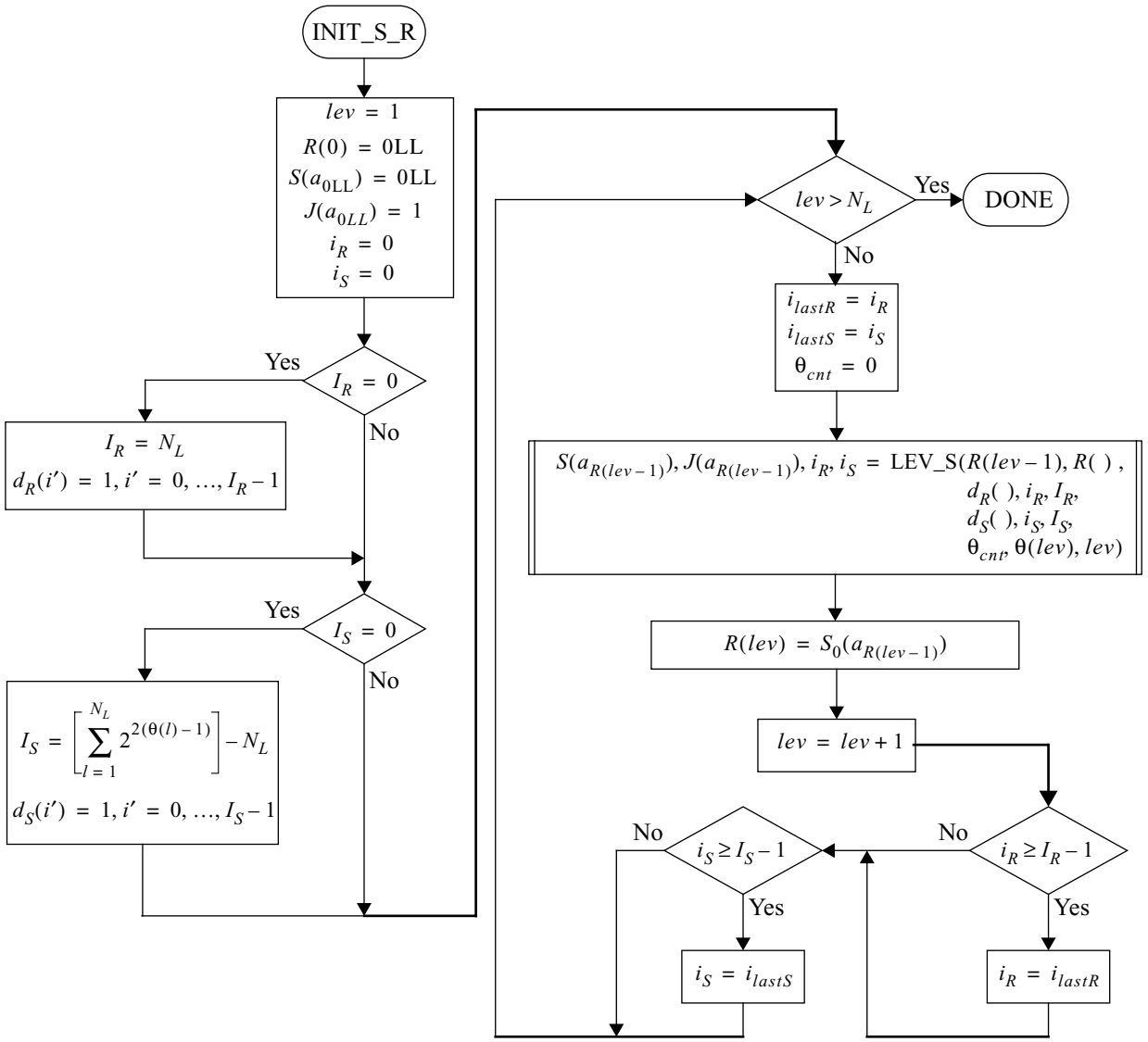

Figure F-11 Upper level procedure for defining $S(a_b)$ and $R(lev)$.

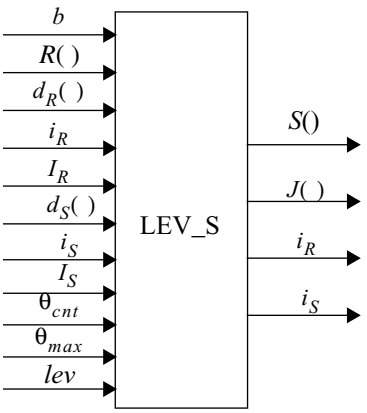

Figure F-12 Parameters for the LEV_S procedure.

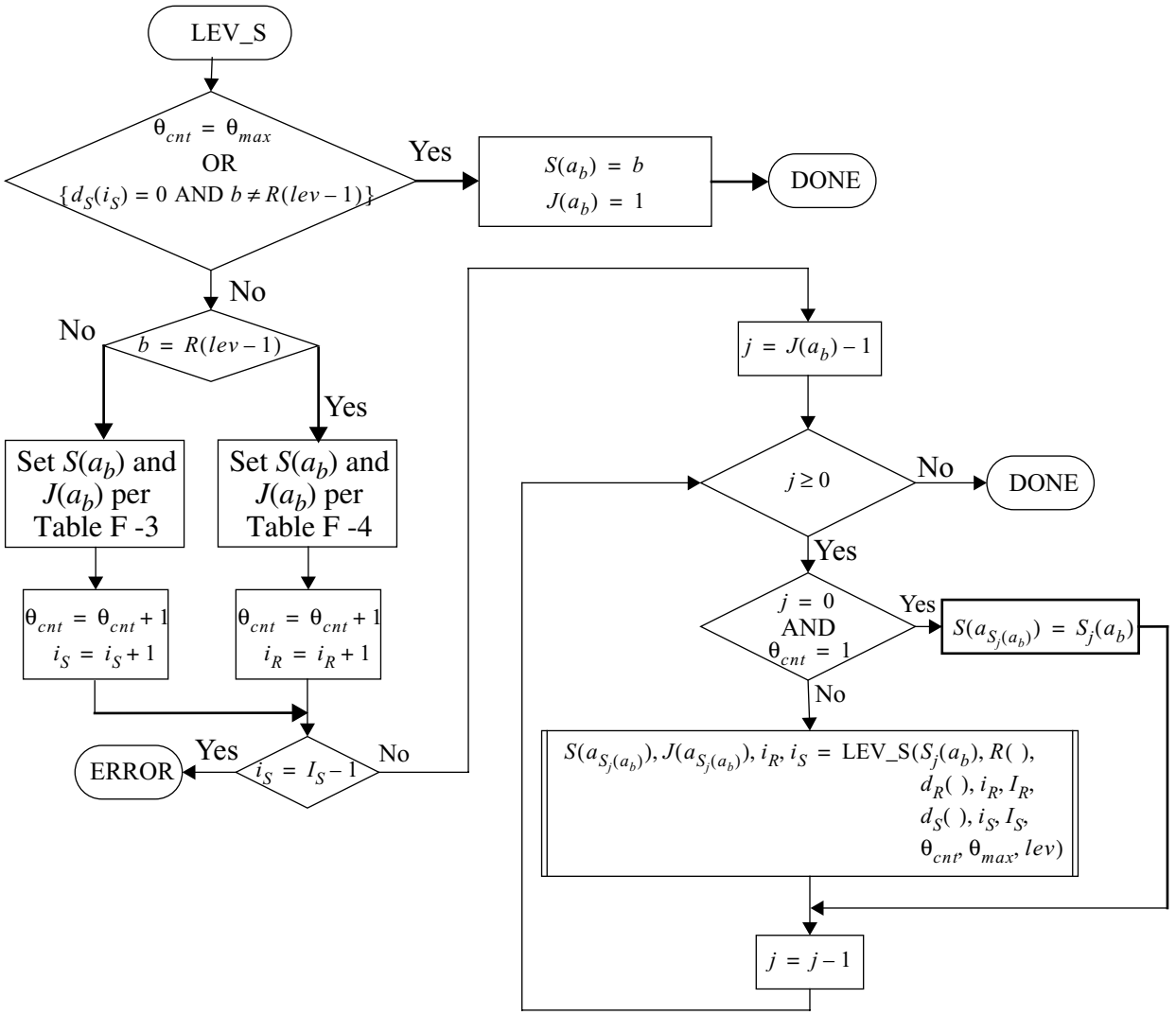

Figure F-13 Procedure for defining $S(a_b)$.

Table F-3 Concatenation table rom subband a_b at current decomposition level.

$d_S(i_S)$	$S(a_b)$ = Set of indices for decomposed subbands 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 Concatenation table indices from subband a_b at current decomposition level

$d_R(i_R)$	$S(a_b)$ = Set of indices for decomposed subbands 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

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

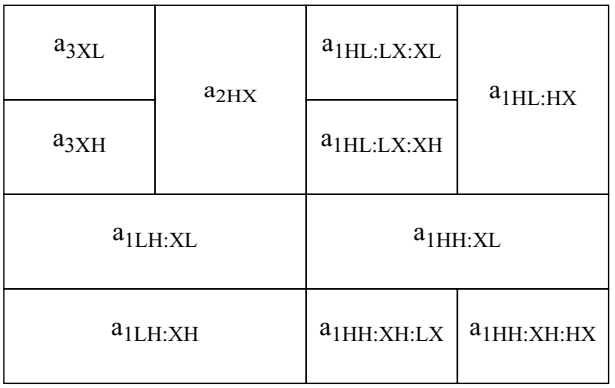

Figure F-14 Sample wavelet decomposition with labeled subbands.

F.3 Inverse discrete wavelet transformation for general decompositions

The inverse transformation process is much like that described in Annex F.3 of ITU-T T.800 | IS 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.

Table F-5 Attributes for sample wavelet decomposition in Figure F-14.

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

F.3.1 Modified IDWT procedure

The IDWT procedure is essentially the same as that in ITU-T T.800 | IS 15444-1. Differences deal with changes in the call to the MOD_2D_SR procedure 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. Subbands stored in the codestream are provided to MOD_2D_SR in the same order as provided by the order LUT *O*() defined in Annex F.2.4.

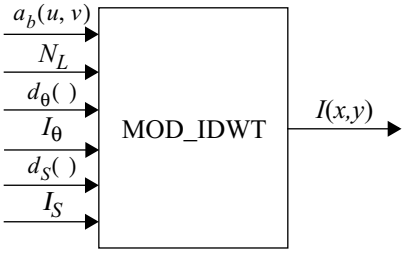

Figure F-15 Parameters for the MOD_IDWT procedure.

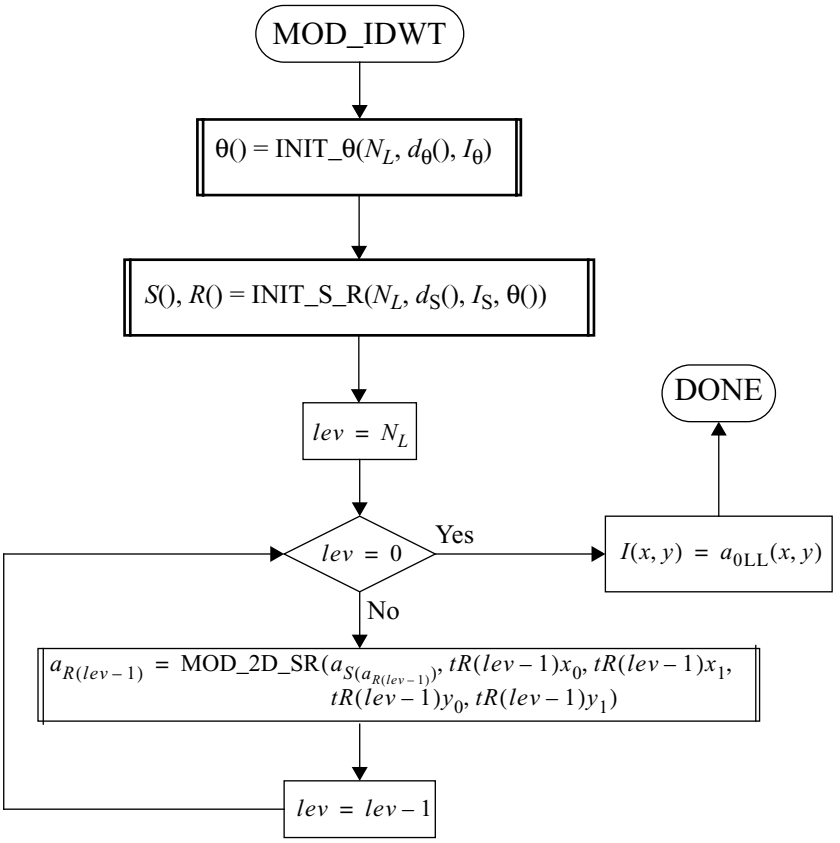

Figure F-16 The MOD_IDWT procedure.

F.3.2 Modified 2D_SR procedure

Major changes are required for 2D_SR from that in ITU-T T.800 | IS 15444-1. This procedure is composed of operations which combine 2 or 4 subbands into a resulting subband. This procedure must 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.

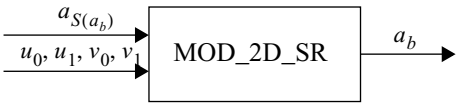

Figure F-17 Parameters for the MOD_2D_SR procedure.

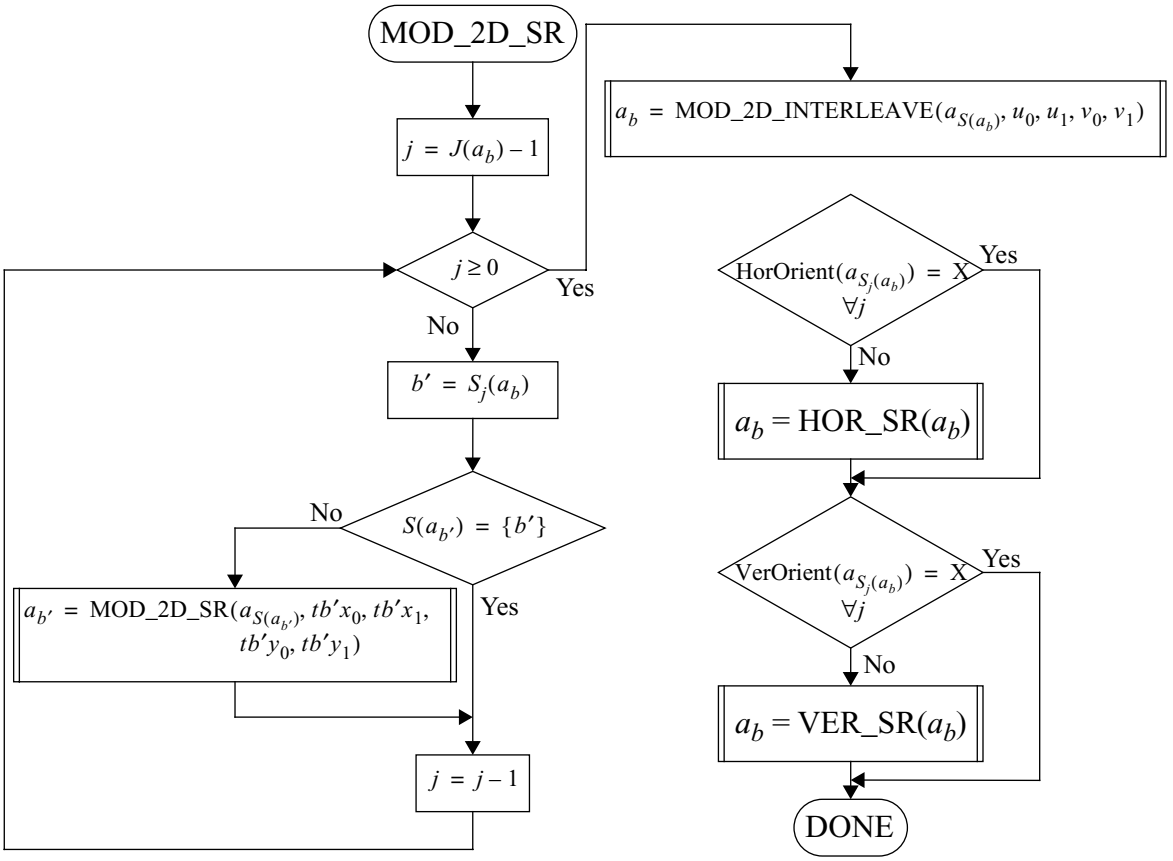

Figure F-18 The MOD_2D_SR procedure.

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 dis-joint horizontal/vertical subband splits. This procedure is shown in both Figure F-19 and Figure F-20. As shown in the later of these two figures, this routine now decides which of three lower level procedures must 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 Annex F.2.4, where a_b is the subband currently interleaved and eventually reconstructed.

Figure F-19 Parameters for the MOD_2D_INTERLEAVE procedure.

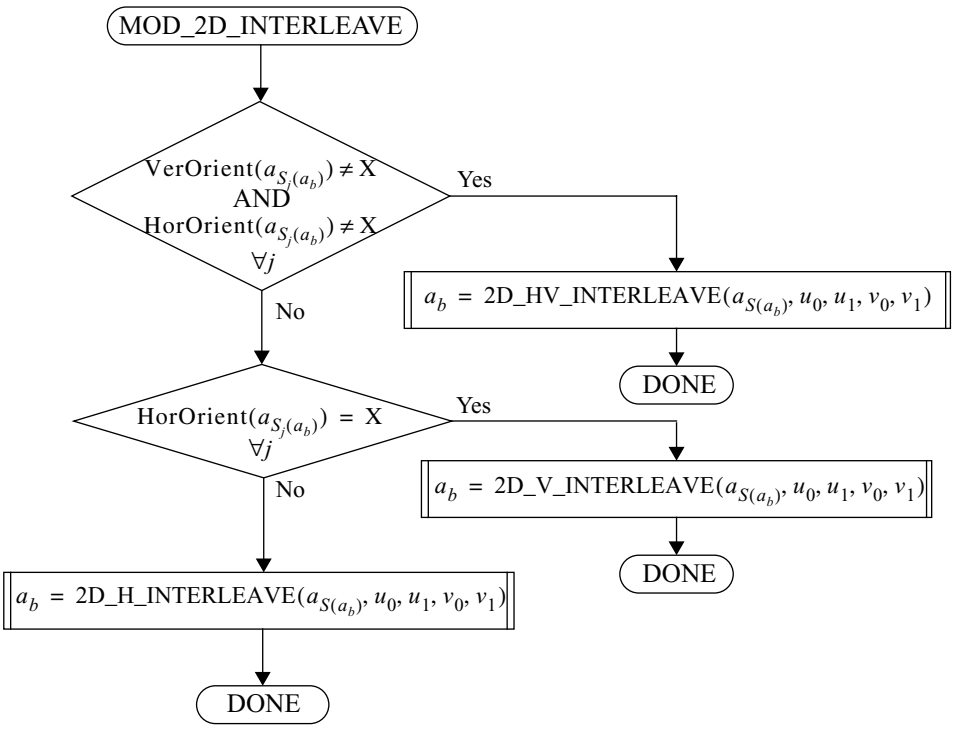

Figure F-20 The MOD_2D_INTERLEAVE procedure.

F.3.3.1 The 2D_HV_INTERLEAVE procedure

The 2D_HV_INTERLEAVE procedure is similar to the 2D_INTERLEAVE procedure from ITU-T T.800 | IS 15444-1. Differences come from only the sub-level processing necessary to interleave samples on a wavelet decomposition level. 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.

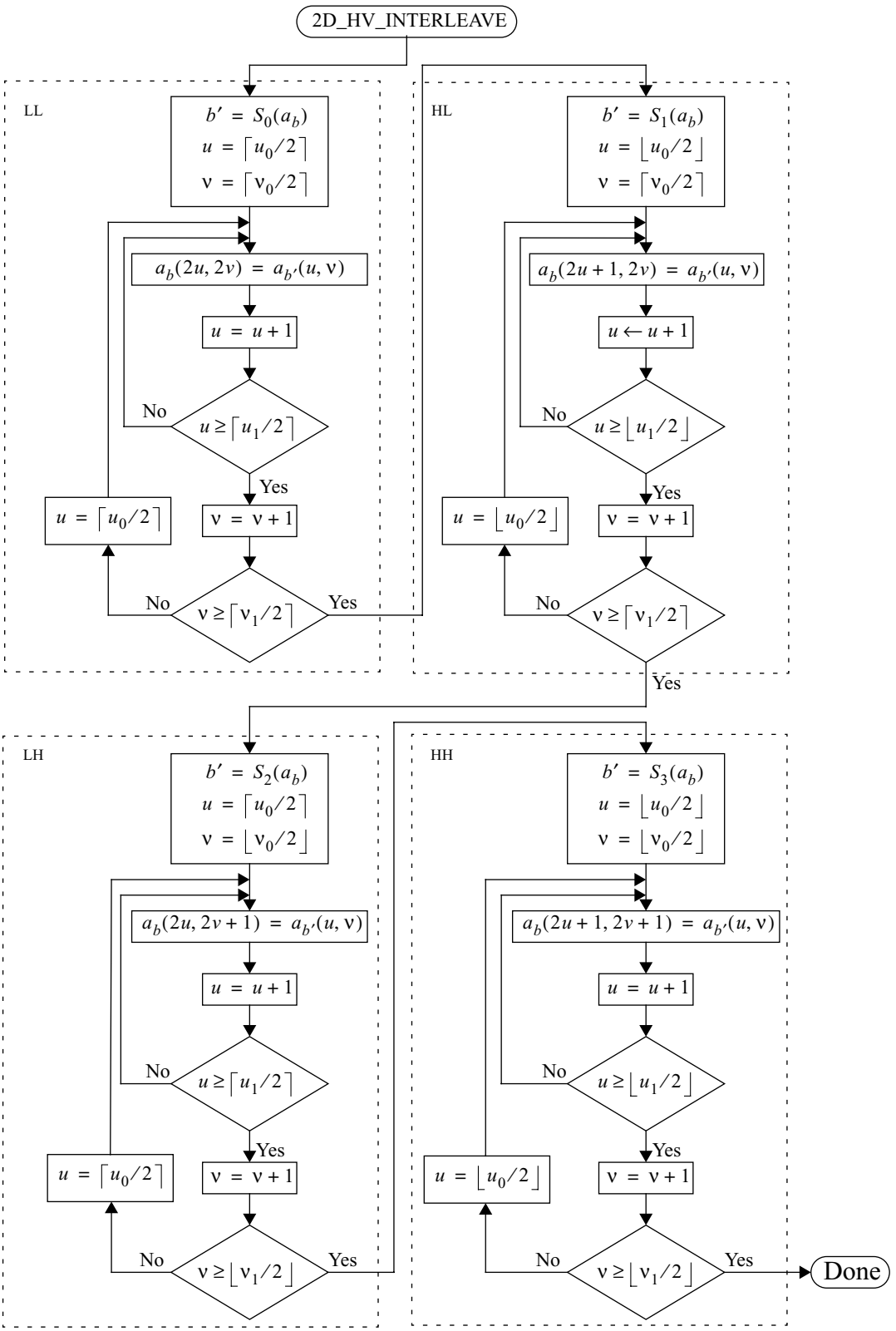

Figure F-22 The 2D_HV_INTERLEAVE procedure.

F.3.3.2 The 2D_H_INTERLEAVE procedure

The 2D_H_INTERLEAVE procedure defined in Figure F-23 and Figure 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-23 Parameters for the 2D_H_INTERLEAVE procedure.

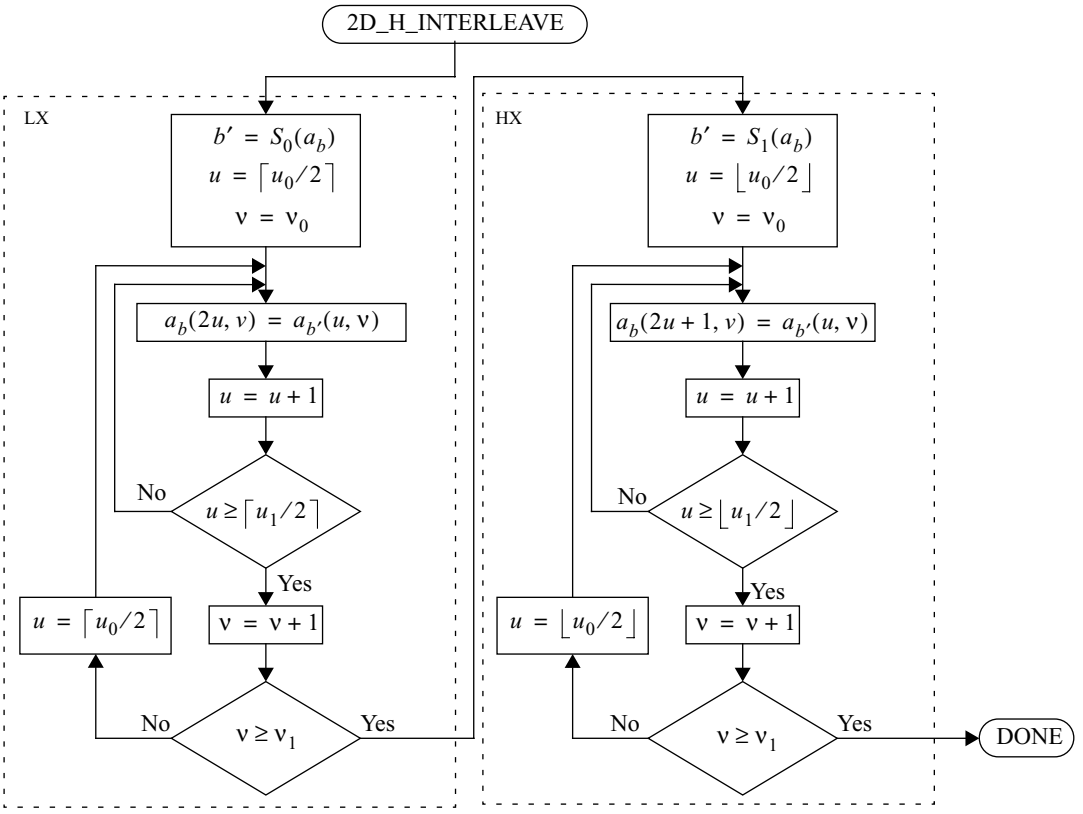

Figure F-24 The 2D_H_INTERLEAVE procedure.

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 Annex F.3.3.2. The procedure for this case is depicted in Figure F-25 and Figure F-26.

Figure F-25 Parameters for the 2D_V_INTERLEAVE procedure.

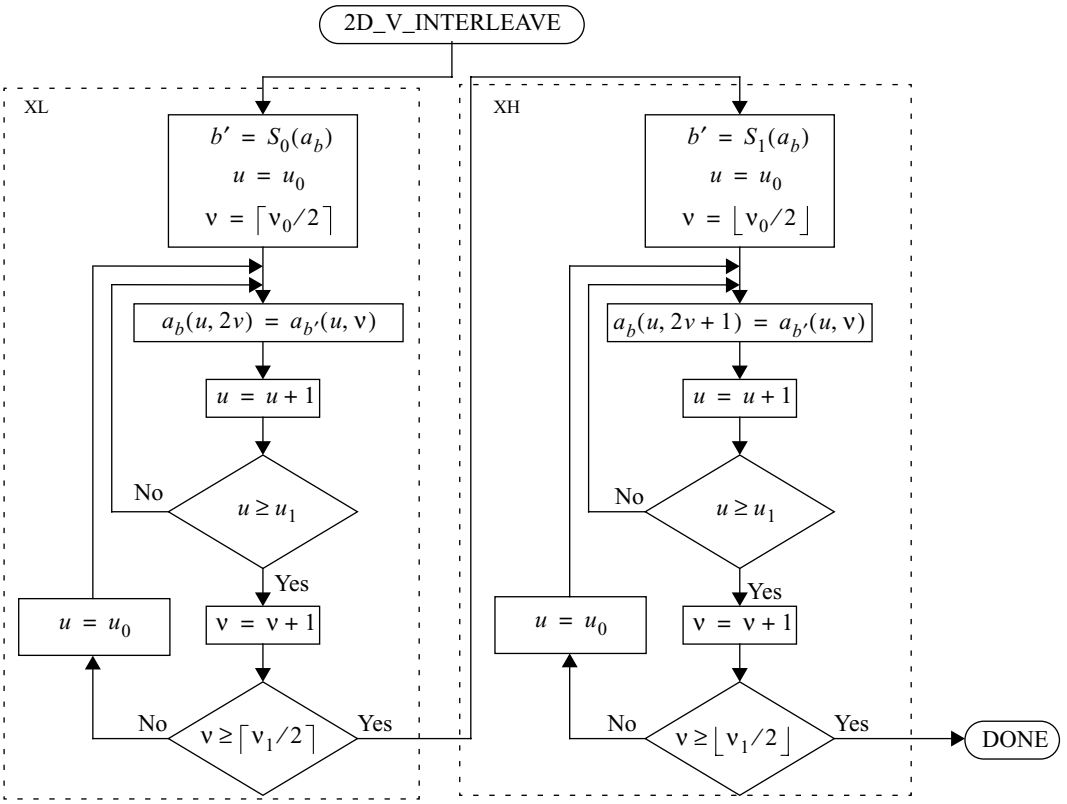

Figure F-26 The 2D_V_INTERLEAVE procedure.

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 ITU-T T.800 | IS 15444-1 procedures.

F.4.1 Modified FDWT procedure

Like the MOD_IDWT procedure, the FDWT remains much like that in ITU-T T.800 | IS 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.

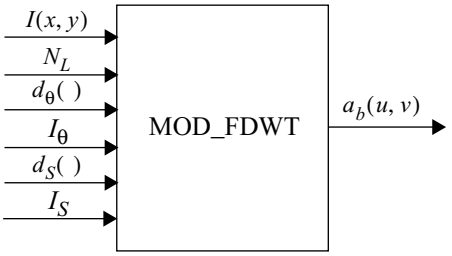

Figure F-27 Parameters for the MOD_FDWT procedure.

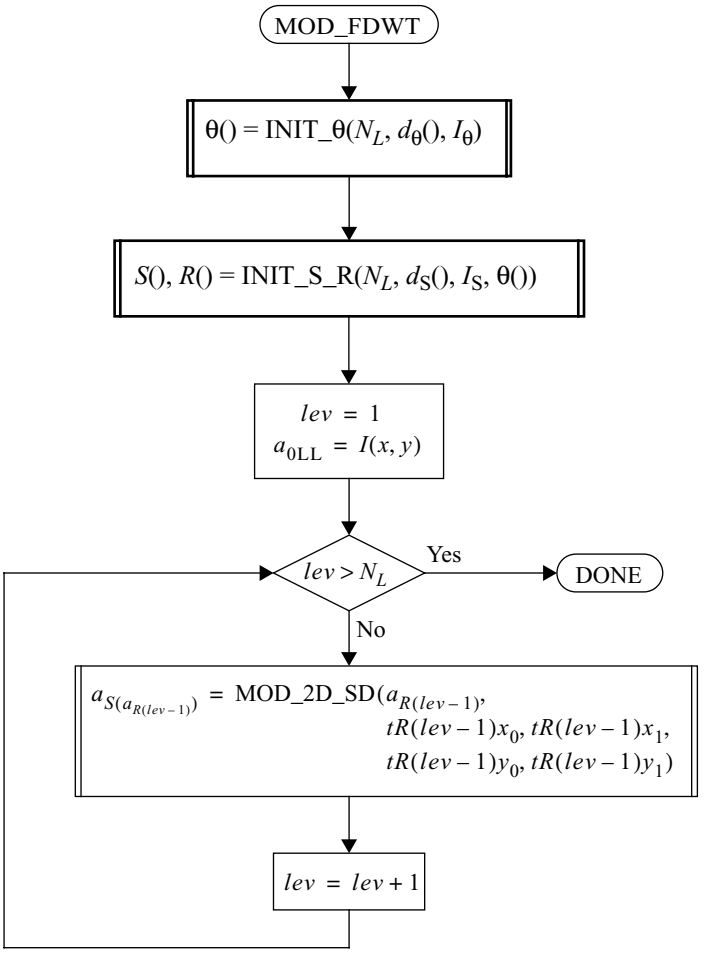

Figure F-28 The MOD_FDWT procedure.

F.4.2 Modified 2D_SD procedure

Major changes are required for 2D_SD from that in ITU-T T.800 | IS 15444-1. This procedure is composed of operations which split one subband into 2 or 4 resulting subbands. This procedure must 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.

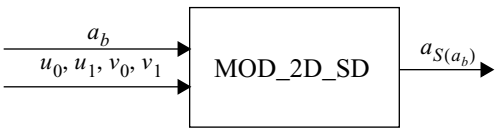

Figure F-29 Parameters for the MOD_2D_SD procedure.

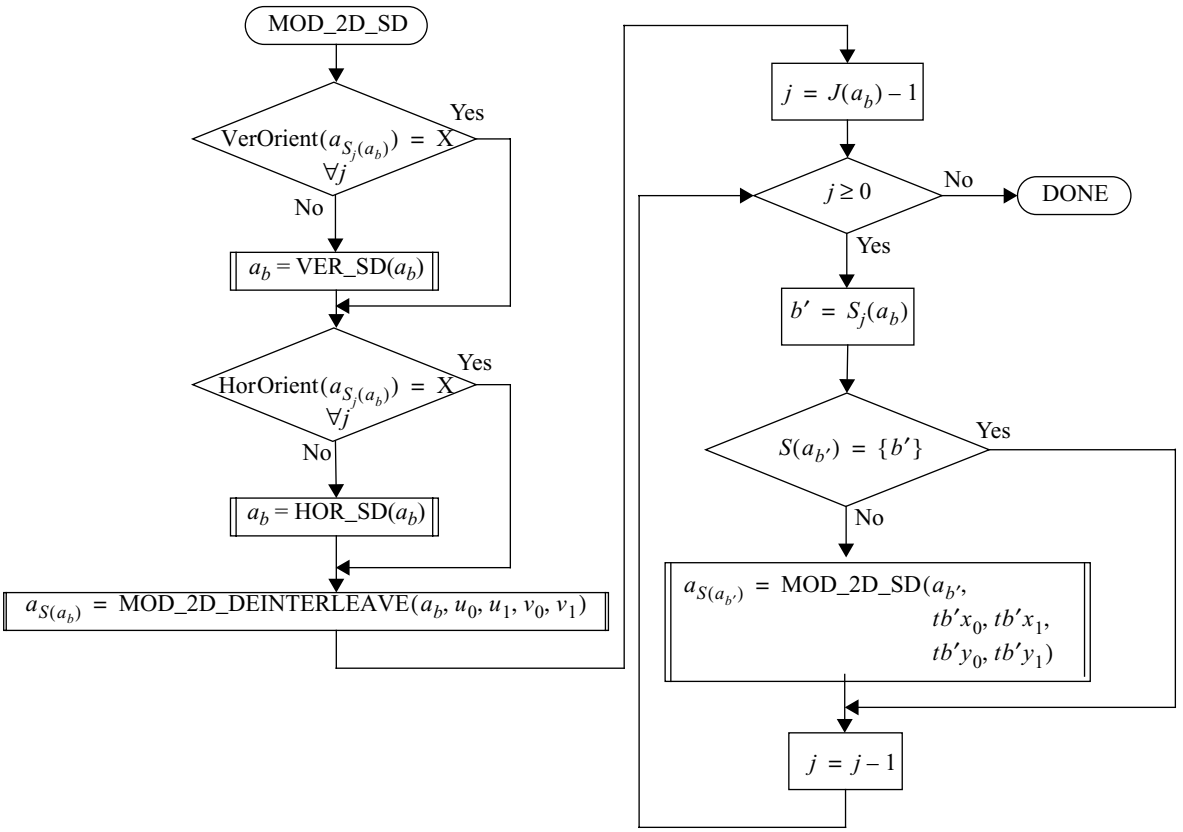

Figure F-30 The MOD_2D_SD procedure.

F.4.3 Modified 2D_DEINTERLEAVE procedure

Significant change were also necessary for the 2D_DEINTERLEAVE procedure. These changes are due to both sub-level processing and dis-joint horizontal/vertical subband splits. This procedure is shown in both Figure F-31 and Figure F-32. As shown in the later of these two figures, this routine now decides which of three lower level procedures must 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 Annex F.2.4 for the subband a_b which is being decomposed/deinterleaved.

Figure F-31 Parameters for the MOD_2D_DEINTERLEAVE procedure.

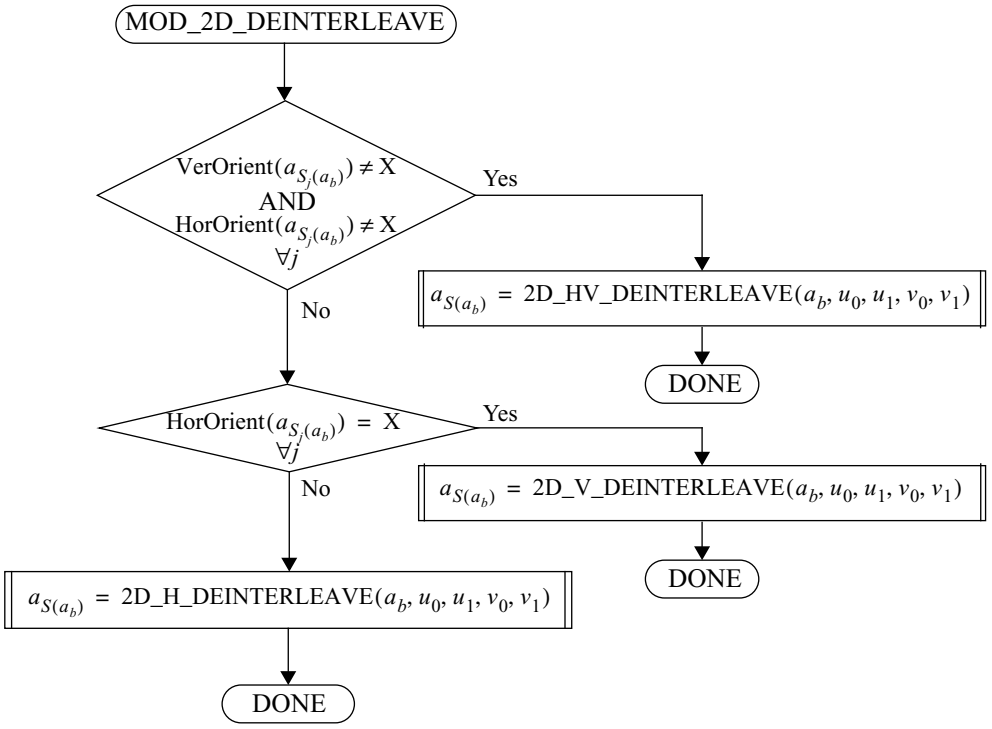

Figure F-32 The MOD_2D_DEINTERLEAVE procedure.

F.4.3.1 The 2D_HV_DEINTERLEAVE procedure

The 2D_HV_DEINTERLEAVE procedure is similar to the 2D_DEINTERLEAVE procedure from ITU-T T.800 | IS 15444-1. Difference come from only the sub-level processing necessary to interleave samples on a wavelet decomposition level. 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.

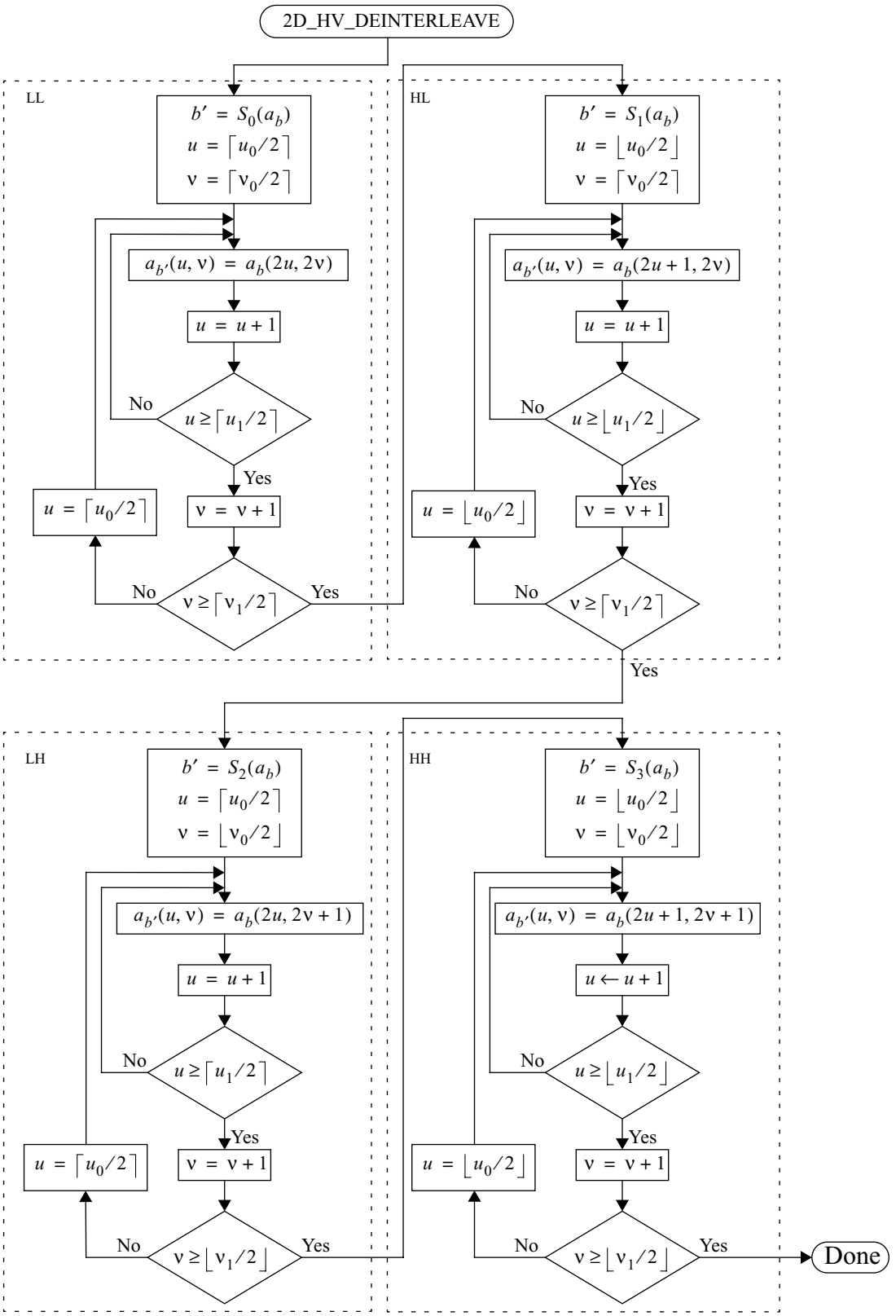

Figure F-34 The 2D_HV_DEINTERLEAVE procedure.

F.4.3.2 The 2D_H_DEINTERLEAVE procedure

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

Figure F-35 Parameters for the 2D_H_DEINTERLEAVE procedure.

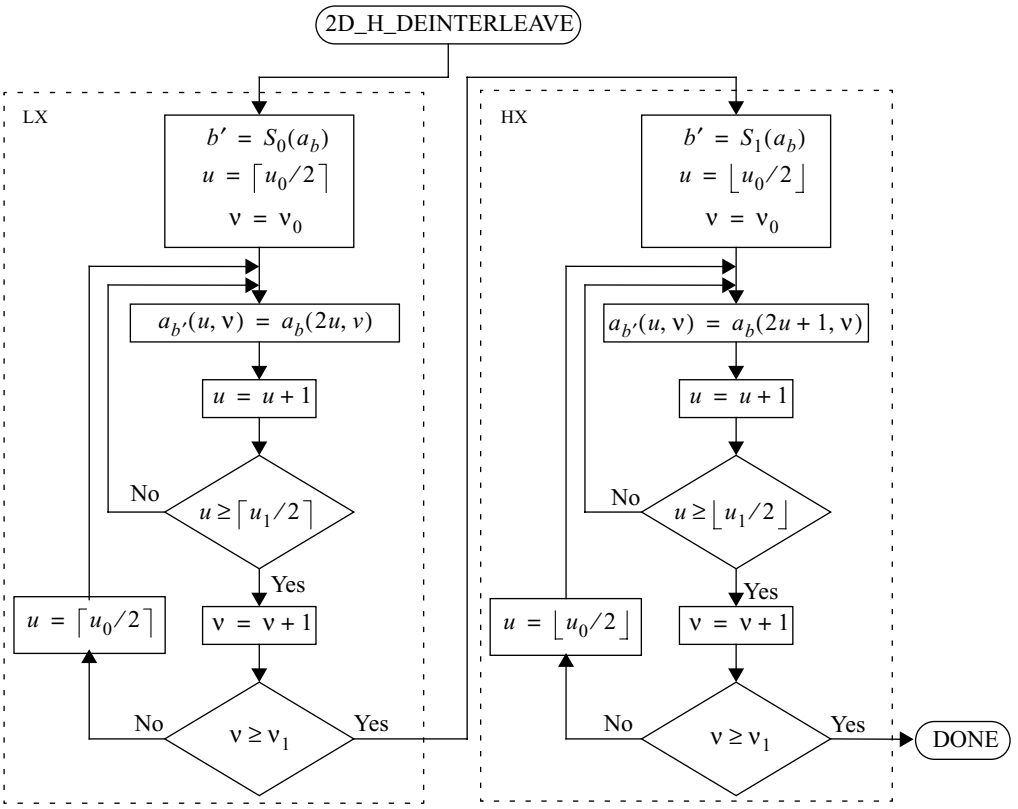

Figure F-36 The 2D_H_DEINTERLEAVE procedure.

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 Annex F.4.3.2. The procedure for this case is depicted inFigure F-37 and Figure F-38

Figure F-37 Parameters for the 2D_V_DEINTERLEAVE procedure.

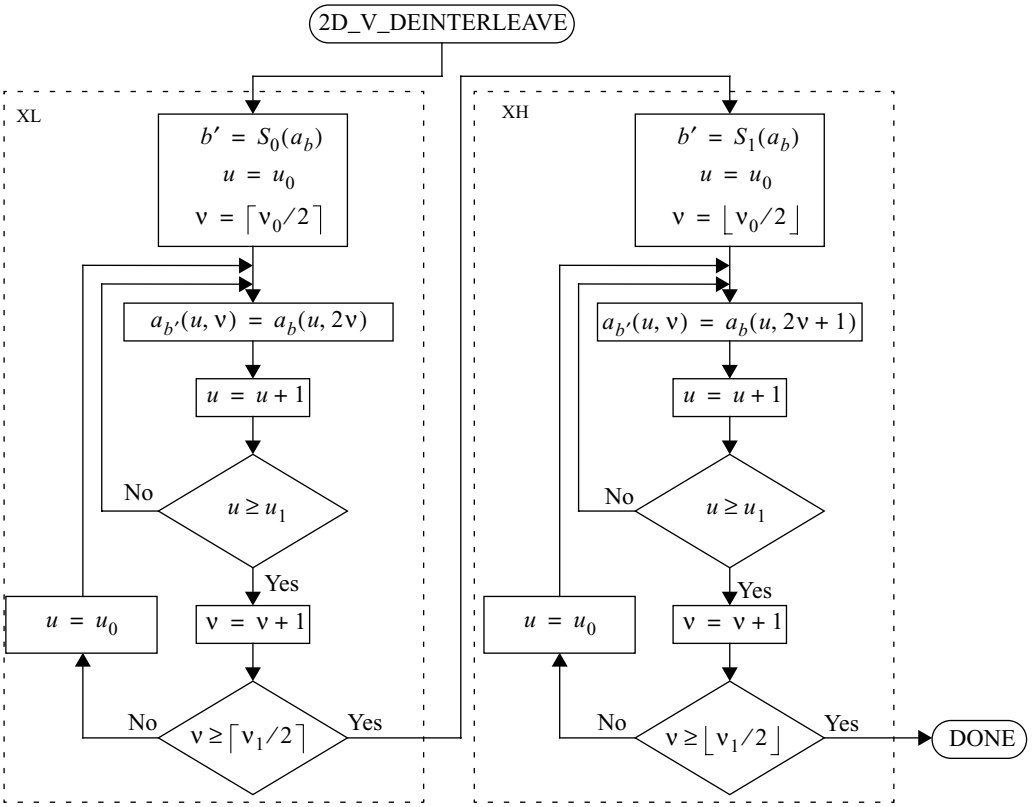

Figure F-38 The 2D_V_DEINTERLEAVE procedure.

Annex G

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.

G.1 One-dimensional wavelet transformation options

This Annex specifies one extension of the one-dimensional subband reconstruction procedure 1D_SR (see ITU-T T.800 | IS 15444-1): the 1D_SR_ARB procedure for two categories of wavelet transformations, whole-sample symmetric (WS) and half-sample symmetric (HS).

G.2 Signalling and interpretation of wavelet transformation parameters

Table G-1 lists the parameters for the wavelet filters used in the wavelet transformations and that are signaled in the codestream.

Table G-1 — Parameters for wavelet filters

Parameter tag	Meaning	Value(s)
Filt_Cat	Wavelet filter category	0 (arbitrary filters) 1 (WS filters) 2 (HS filters)
WT_Typ	Wavelet transformation type	0 (irreversible filters) 1 (reversible filters)
Coeff_Typ	Numerical type of lifting step coefficients	8 (8-bit signed integer) 16 (16-bit signed integer) 32 (32-bit signed integer) 64 (64-bit signed integer) 128 (128-bit signed integer)
N_{LS}	Number of lifting steps	positive integer
m_0	lowpass/highpass characteristic of lifting step number 0	0 (even-indexed subsequence) 1 (odd-indexed subsequence)
L_s	Number of lifting coefficients at lifting step s	positive integer: defined for $0 \le s < N_{LS}$
$\alpha_{s,k}$	k^{th} lifting coefficients for lifting step s	Coeff_Typ: defined for $0 \le s < N_{LS}$ and $0 \le k < L_s$
off_s	Offset for lifting step s	signed integer: defined for $0 \le s < N_{LS}$
K	Scaling factor **(irreversible transformations only)**	Coeff_Typ

Table G-1 — Parameters for wavelet filters

Parameter tag	Meaning	Value(s)
β_s	Additive residue for lifting step s (**reversible transformations only**)	nonnegative Coeff_Typ: defined for $0 \le s < N_{LS}$
ε_s	Base-2 scaling exponent for lifting step s (**reversible transformations only**)	nonnegative integers: defined for $0 \le s < N_{LS}$

G.3 Definitions and normalizations for lifted wavelet transform filter banks

G.3.1 Definition of the WS and HS filter bank categories

Define the following sequences:

$a_s(k)=\alpha_s(-k)$ for s=0,...,N_{LS-1} and all k

I(k)=1 for $k=0$ and I(k)=0 for all other k.

Define the following sequences in terms of the above sequences:

$Q^{(0)}{}_{00}=Q^{(0)}{}_{11}=I$

$Q^{(0)}{}_{01}=Q^{(0)}{}_{10}=0$

For s=0,...,N_{LS-1} define:

$A^{(s)}{}_{00}=A^{(s)}{}_{11}=I$

$A^{(s)}{}_{01}=a_s$ if $m_s=0$, $A^{(s)}{}_{01}=0$ otherwise

$A^{(s)}{}_{10}=a_s$ if $m_s=1$, $A^{(s)}{}_{10}=0$ otherwise

Define the following operation on sequences:

$$(a*b)(k) = \sum_n a(n)b(k-n)$$

Use the * operation to define the product $A^{(s-1)}*Q^{(s-1)}$ that appears in the flowchart as follows:

$Q^{(s)}{}_{ij} = A^{(s-1)}{}_{i0}*Q^{(s-1)}{}_{0j} + A^{(s-1)}{}_{i1}*Q^{(s-1)}{}_{1j}$ for i, j=0, 1 and s=1,...,N_{LS-1}.

With these definitions, the output $\{Q_{00}, Q_{01}, Q_{10}, Q_{11}\}$ of the following flowchart is well-defined:

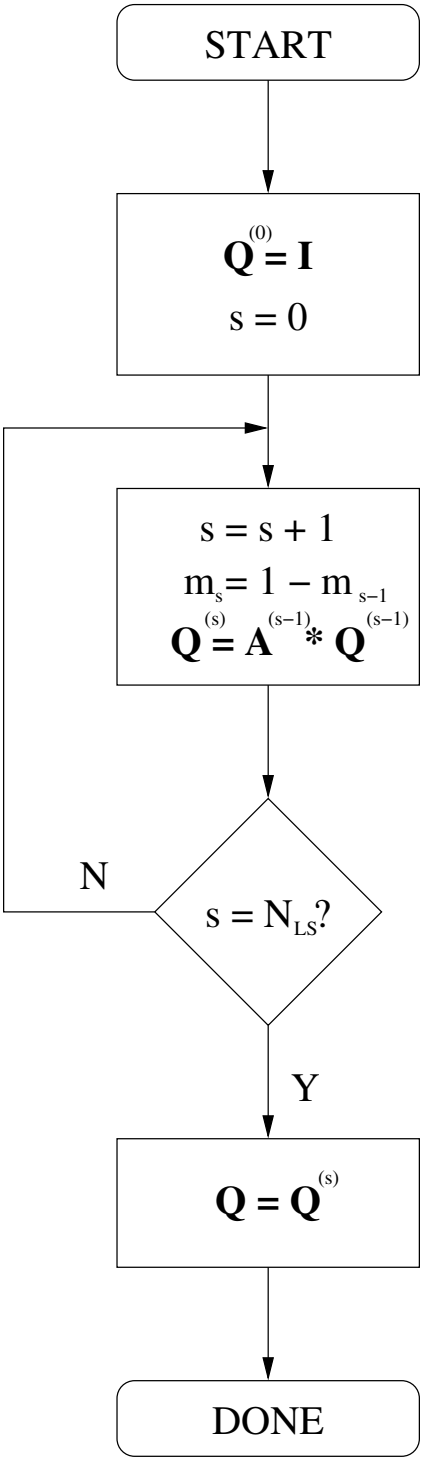

Figure G-1 — What is this?

The WS and HS categories are defined in terms of the flowchart output, $\{Q_{00}, Q_{01}, Q_{10}, Q_{11}\}$, as follows.

The filter bank defined by the parameters in Table G-1 is WS if and only if the following four equations are satisfied for all n:

$Q_{00}(n)=Q_{00}(-n)$

$Q_{01}(n)=Q_{01}(1-n)$

$Q_{10}(n)=Q_{10}(-1-n)$ and

$Q_{11}(n)=Q_{11}(-n)$.

Similarly, the filter bank defined by the parameters in Table G-1 is HS if and only if the following two equations are satisfied for all n:

$Q_{00}(n)=Q_{01}(-n)$ and

$Q_{10}(n)=-Q_{11}(-n)$.

G.3.2 Normalisation of reversible wavelet transformation filters

Define B_s recursively defined as

$$B_s = B_{s-2} + 2\left(\sum_{k=0}^{L_s-1} \alpha_{s,k}\right)\frac{B_{s-1}}{2^{\varepsilon_s}}, \qquad \text{G.1}$$

with the initial conditions being: $B_0 = 1$ and $B_1 = 1 + 2\left(\sum_{k=0}^{L_0-1} \alpha_{0,k}\right)/2^{\varepsilon_1}$.

The parameters $B_{N_{LS}-1}$ and $B_{N_{LS}}$ must satisfy $B_{N_{LS}-1} = 1$ and $B_{N_{LS}} = 2$ if m_0 and N_{LS} are of the same parity, and they must satisfy $B_{N_{LS}-1} = 2$ and $B_{N_{LS}} = 1$ if m_0 and N_{LS} are of the opposite parity.

G.3.3 Normalisation of irreversible wavelet transformation filters

Define B_s recursively defined as

$$B_s = B_{s-2} + 2\left(\sum_{k=0}^{L_s-1} \alpha_{s,k}\right)B_{s-1}, \qquad \text{G.2}$$

with the initial conditions being: $B_0 = 1$ and $B_1 = 1 + 2\left(\sum_{k=0}^{L_0-1} \alpha_{0,k}\right)$.

The parameters $B_{N_{LS}-1}$ and $B_{N_{LS}}$ must satisfy $B_{N_{LS}-1} = 1/K$ and $B_{N_{LS}} = 2K$ if m_0 and N_{LS} are of the same parity, and they must satisfy $B_{N_{LS}-1} = 2K$ and $B_{N_{LS}} = 1/K$ if m_0 and N_{LS} are of the opposite parity.

G.4 One-dimensional subband reconstruction procedure

The one-dimensional subband reconstruction filtering (1D_SR_ARB) 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.4.1 The 1D_SR_ARB procedure

As illustrated in Figure G-2, the 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*, the index i_1 of the sample following the last sample in array *Y*, and the filter category parameter Filt_cat. They produce as output an array, *X*, with the same indices (i_0, i_1).

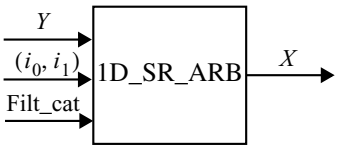

Figure G-2 — Parameters of the 1D_SR_ARB procedures

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 G-14, the 1D_SR_ARB procedure first uses the 1D_EXTR_ARB procedure to extend the signal *Y* beyond its left and right boundaries, resulting in the extended signal Y_{ext}. The 1D_SR_ARB procedure then applies the 1D_FILTR_ARB procedure to Y_{ext} to produce the reconstructed signal, *X*.

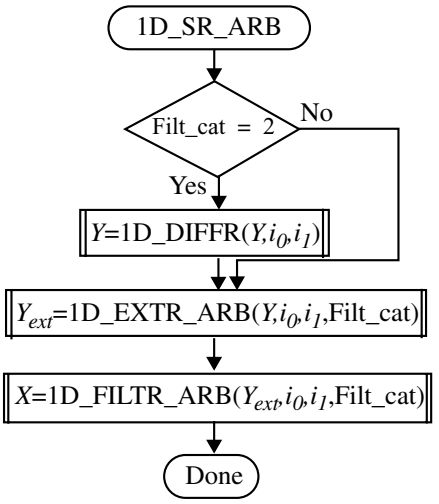

Figure G-3 — The 1D_SR_ARB **procedure**

G.4.2 The 1D_DIFFR procedure

The 1D_DIFFR procedure, shown in Figure G-4, takes as input *Y*, i_0, and produces as output a modified version of *Y* and an intermediate temporary value, L_0.

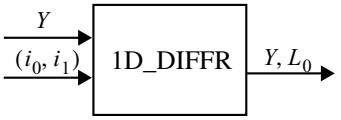

Figure G-4 — Parameters of the 1D_DIFFR procedure

The 1D_DIFFR procedure sets the value of L_0 to $L_0 = Y(i_0) + Y(i_0 + 2)$. Furthermore, if i_0 is odd, then $Y(i_0)$ is set to $Y(i_0) = 0$.

G.4.3 The 1D_EXTR procedure

The 1D_EXTR procedure calculates the values of $Y_{ext}(i)$ for values of i beyond the range $i_0 \le i < i_1$. Two versions of the procedure are defined: one for WS filter banks (parameter Filt_Cat=1) and another version for HS filter banks (parameter Filt_Cat=2).

G.4.3.1 Procedure 1D_EXTR for WS filter banks

When the parameter Filt_Cat is equal to one, the 1D_EXTR procedure for WS filter banks is identical to the 1D_EXTR procedure defined in ITU-T T.800 | IS 15444-1, Annex F.3.7, except for the specification of the minimum extension lengths, i_{left} and i_{right}.

G.4.3.2 Procedure 1D_EXTR for HS filter banks

When the parameter Filt_Cat is equal to 2, the 1D_EXTR procedure for HS filter banks is defined as follows. The extended input, $Y_{ext}(i)$, is defined as follows, depending on the parity of index i_0.

G.4.3.2.1 Case where i_0 is even

For values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0)) < i_1 - i_0$

$$Y_{ext}(i) = Y_{ext}(i_0 + \mathrm{mod}(i - i_0, 2(i_1 - i_0))) . \quad \text{G.3}$$

For even values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0)) \ge i_1 - i_0$,

$$Y_{ext}(i) = Y_{ext}(i_0 + \mathrm{mod}(2(i_1 - i_0 - 1) - (i - i_0), 2(i_1 - i_0))) . \quad \text{G.4}$$

For odd values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0)) \ge i_1 - i_0$,

$$Y_{ext}(i) = -Y_{ext}(i_0 + \mathrm{mod}(2(i_1 - i_0) - (i - i_0), 2(i_1 - i_0))) . \quad \text{G.5}$$

G.4.3.2.2 Case where i_0 is odd

For values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0 + 1)) < i_1 - i_0 + 1$

$$Y_{ext}(i) = Y_{ext}(i_0 + \mathrm{mod}(i - i_0, 2(i_1 - i_0 + 1))) . \quad \text{G.6}$$

For values of i such that $\mathrm{mod}((i - i_0, 2(i_1 - i_0 + 1)) = 2(i1 - i0))$

$$Y_{ext}(i) = L_0 \quad \text{G.7}$$

For even values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0 + 1)) \ge i_1 - i_0 + 1$ and $\mathrm{mod}((i - i_0, 2(i_1 - i_0 + 1)) < 2(i1 - i0))$,

$$Y_{ext}(i) = Y_{ext}(i_0 + \mathrm{mod}(2(i_1 - i_0 - 1) - (i - i_0), 2(i_1 - i_0))) . \quad \text{G.8}$$

For odd values of i such that $\mathrm{mod}(i - i_0, 2(i_1 - i_0)) \ge i_1 - i_0$,

$$Y_{ext}(i) = -Y_{ext}(i_0 + \mathrm{mod}(2(i_1 - i_0) - (i - i_0), 2(i_1 - i_0))) . \quad \text{G.9}$$

G.4.4 The one-dimensional reconstruction filtering (1D_FILTR_ARB) procedures

Two versions of the reconstruction filtering procedure (1D_FILTR_ARB) are specified, depending on whether the transformation is reversible or not.

G.4.4.1 The reversible one-dimensional reconstruction filtering (1D_FILTR_ARB) procedure

As shown in Figure G-5, the input parameters to procedure 1D_FILTR_ARB are $V_{ext}, i_0, i_1, off_s, \alpha_{s,k}, L_s, N_{LS}$.

Figure G-5 — Parameters of the 1D_FILTR_ARB procedure

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

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) - \left\lfloor \frac{\left(\sum_{k=off\bar{s}}^{offs+L_s-1} \alpha_{s,k} \cdot V_{ext}(2n+1-m_s+2k) \right) + \beta_s}{2^{e_s}} \right\rfloor, \qquad \text{G.10}$$

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.

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

$$W(k) = V_{ext}(k). \qquad \text{G.11}$$

G.4.4.2 The irreversible one-dimensional reconstruction filtering (1D_FILTR_ARB) procedure

As shown in Figure G-6, the input parameters to procedure 1D_FILTR_ARB are $V_{ext}, i_0, i_1, off_s, \alpha_{s,k}, L_s, N_{LS}, K$.

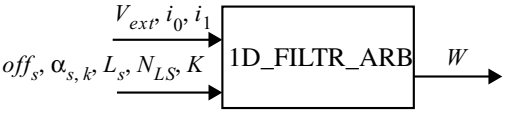

Figure G-6 — Parameters of the 1D_FILTR_ARB procedure

The 1D_FILTR procedure starts with two scaling steps:

$$V_{ext}(2n) = K \cdot V_{ext}(2n) \quad \text{for} \quad i_0 - i_{left} \le 2n < i_1 + i_{right}, \qquad \text{G.12}$$

$$\text{and } V_{ext}(2n+1) = (1/K) \cdot V_{ext}(2n+1) \text{ for } i_0 - i_{left} \le 2n+1 < i_1 + i_{right}. \qquad \text{G.13}$$

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

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) - \left(\sum_{k=offs}^{offs+L_s-1} \alpha_{s,k} \cdot V_{ext}(2n+1-m_s+2k) \right), \qquad \text{G.14}$$

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.

The values of $V_{ext}(k)$ such that $i_0 \le k < i_1$ form the output $W(k)$ of the 1D_FILTR_ARB procedure:

$$W(k) = V_{ext}(k) . \qquad \text{G.15}$$

G.5 One-dimensional subband decomposition procedures

The one-dimensional subband reconstruction filtering (1D_SD_ARB) 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.5.1 The 1D_SD_ARB procedure

As illustrated in Figure G-7, the 1D_SD_ARB 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).

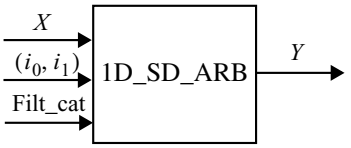

Figure G-7 — Parameters of the 1D_SD_ARB procedure

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 $Y(i_0) = X(i_0)/2$ 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-8, the 1D_SD_ARB procedure first uses the 1D_EXTD_ARB procedure to extend the signal X beyond its left and right boundaries, resulting in the

extended signal X_{ext}. The 1D_SD_ARB procedure then applies the 1D_FILTD_ARB procedure to X_{ext} to produce the decomposed signal, X.

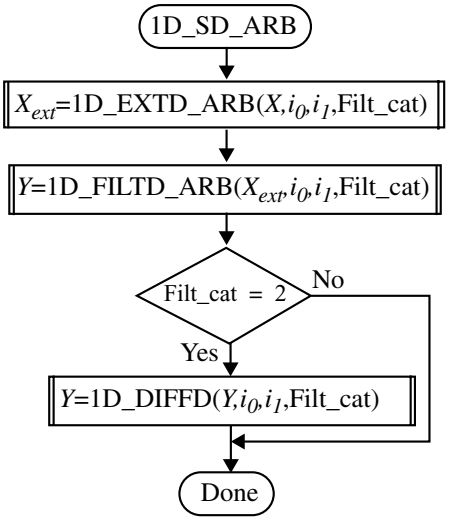

Figure G-8 — The 1D_SD_ARB **procedure**

G.5.2 The 1D_EXTD_ARB procedure

The 1D_EXTD_ARB procedure calculates the values of $X_{ext}(i)$ for values of i beyond the range $i_0 \le i < i_1$. Two versions of the procedure are defined: one for WS filter banks (parameter Filt_Cat=1) and another version for HS filter banks (parameter Filt_Cat=2).

G.5.2.1 Procedure for WS wavelet transformations

When parameter Filt_Cat=1, the 1D_EXTD_ARB procedure for WS filter banks is identical to the 1D_EXTD procedure defined in ITU-T T.800 | IS 15444-1, Annex F.3.7, except for the specification of the minimum extension lengths, i_{left} and i_{right}.

G.5.2.2 Procedure for HS wavelet transformations

When the parameter Filt_Cat=2, the 1D_EXTD_ARB procedure calculates the values of $X_{ext}(i)$ for values of i beyond the range $i_0 \le i < i_1$, as given in:

$$X_{ext}(i) = X(PSE_E(i, i_0, i_1)), \qquad \text{G.16}$$

where $PSE_E(i, i_0, i_1)$ is given by Equation G.17:

$$SE_E(i, i_0, i_1) = i_0 + \min(\mathrm{mod}((i - i_0, 2(i_1 - i_0)), 2(i_1 - i_0) - 1 - \mathrm{mod}(i - i_0, 2(i_1 - i_0))). \qquad \text{G.17}$$

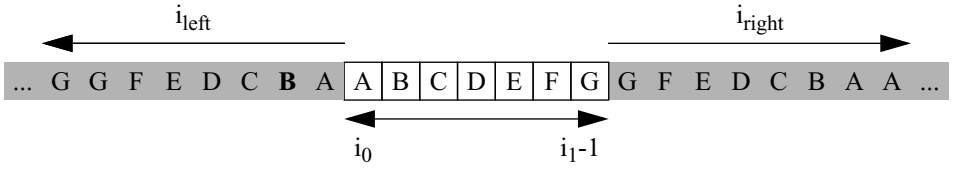

Figure G-9 Periodic symmetric extension of input signal for HS filter banks

NOTE $PSE_E(i, i_0, i_1) = i$, for $i_0 \le i < i_1$

Table G-2 gives an example of values of $PSE_E(i, i_0, i_1)$.

Table G-2 Example of $PSE_E(i, i_0, i_1)$

i	$i_0 - 3$	$i_0 - 2$	$i_0 - 1$	i_0	$i_0 + 1$	...	$i_1 - 1$	i_1	$i_1 + 1$	$i_1 + 2$	$i_1 + 3$
$PSE_E(i, i_0, i_1)$	$i_0 + 2$	$i_0 + 1$	i_0	i_0	$i_0 + 1$	...	$i_1 - 1$	$i_1 - 1$	$i_1 - 2$	$i_1 - 3$	$i_1 - 4$

G.5.3 The one-dimensional decomposition filtering (1D_FILTD_ARB) procedure

Two versions of the decomposition filtering procedure are specified, depending on whether the transformation is reversible or not.

G.5.3.1 The reversible one-dimensional decomposition filtering (1D_FILTD_ARB) procedure

As shown in Figure G-10, the input parameters to the reversible version of procedure 1D_FILTD_ARB are $V_{ext}, i_0, i_1, off_s, \alpha_{s,k}, L_s, N_{LS}$.

Figure G-10 — Parameters of the 1D_FILTD_ARB procedure

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

$$V_{ext}(2n + m_s) = V_{ext}(2n + m_s) + \left\lfloor \frac{\left(\sum_{k=offs}^{offs + L_s - 1} \alpha_{s,k} \cdot V_{ext}(2n + 1 - m_s + 2s) \right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor , \qquad \text{G.18}$$

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.

The values $W(k) = V_{ext}(k)$ form the output $W(k)$ of the 1D_FILTD_ARB procedure. When Filt_Cat=2 and i_0 is odd, the output values are the values in the range $i_0 - 1 \le k < i_1$.

Otherwise, the output values are the values in the range $i_0 \le k < i_1$.

G.5.3.2 The irreversible one-dimensional decomposition filtering (1D_FILTD_ARB) procedure

As shown in Figure G-11, the input parameters to the irreversible version of procedure 1D_FILTD_ARB are $V_{ext}, i_0, i_1, off_s, \alpha_{s,k}, L_s, N_{LS}, K$.

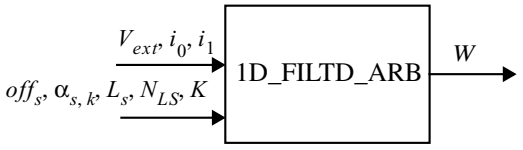

Figure G-11 — Parameters of the 1D_FILTD_ARB procedure

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

$$V_{ext}(2n + m_s) = V_{ext}(2n + m_s) - \left(\sum_{k = offs}^{offs + L_s - 1} \alpha_{s,k} \cdot V_{ext}(2n + 1 - m_s + 2k) \right), \quad \text{G.19}$$

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.

The 1D_FILTD_ARB procedure ends with two scaling steps:

$$V_{ext}(2n) = (1/K) \cdot V_{ext}(2n) \text{ for } i_0 - i_{left} \le 2n < i_1 + i_{right}, \quad \text{G.20}$$

$$\text{and } V_{ext}(2n+1) = K \cdot V_{ext}(2n+1) \text{ for } i_0 - i_{left} \le 2n + 1 < i_1 + i_{right}. \quad \text{G.21}$$

The values $W(k) = V_{ext}(k)$ form the output $W(k)$ of the 1D_FILTD_ARB procedure. When Filt_Cat=2 and i_0 is odd, the output values are the values in the range $i_0 - 1 \le k < i_1$.

Otherwise, the output values are the values in the range $i_0 \le k < i_1$.

G.5.3.3 The 1D_DIFFD procedure

The 1D_DIFFD procedure, shown in Figure G-12, takes as input *Y, i_0,* and Filt_Cat and produces as output a modified version of Y.

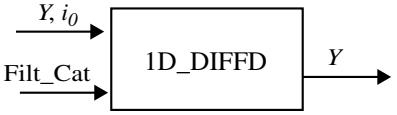

Figure G-12 — Parameters of the 1D_DIFFD procedure

If Filt_Cat=2 (i.e., if the filter bank is an HS filter bank) and i_0 is odd then $Y(i_0) = Y(i_0 - 1) + Y(i_0 + 1)$.

Otherwise, the 1D_DIFFD procedure returns *Y* unmodified.

G.6 Examples of structures (informative)

The following examples illustrate a choice of lifting parameters which guarantee that the wavelet transformations are either WS or HS.

G.6.1 WS filters

The following equations can be used at each lifting step s and are obtained by assuming L_s is even, $\alpha_{s,k} = \alpha_{s,-1-k}$ and $off_s = -1 - \frac{L_s}{2}$.

G.6.1.1 Lifting step *s* for forward reversible transformations

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) - \left\lfloor \frac{\left(\sum_{k=0}^{\frac{L_s}{2}-1} \alpha_{s,k} \cdot (V_{ext}(2n+1-m_s+2k) + V_{ext}(2n-1-m_s-2k)) \right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor \quad . \qquad \text{G.22}$$

G.6.1.2 Lifting step *s* for forward irreversible transformations

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) + \left(\sum_{k=0}^{\frac{L_s}{2}-1} \alpha_{s,k} \cdot (V_{ext}(2n+1-m_s+2k) + V_{ext}(2n-1-m_s-2k)) \right) \qquad \text{G.23}$$

G.6.2 HS filters

The lifting steps given in Figure G-13 guarantee that the filters implemented are HS. Note that the parameters $\alpha_{s,k}$ used in Figure G-13 are different from those used in all other sections. The function $R(x)$ is either the identity $R(x) = x$ (for irreversible transformations) or $R(x) = \left\lceil \frac{x + \beta_s}{2^{\varepsilon_s}} \right\rceil$ for reversible transformations.

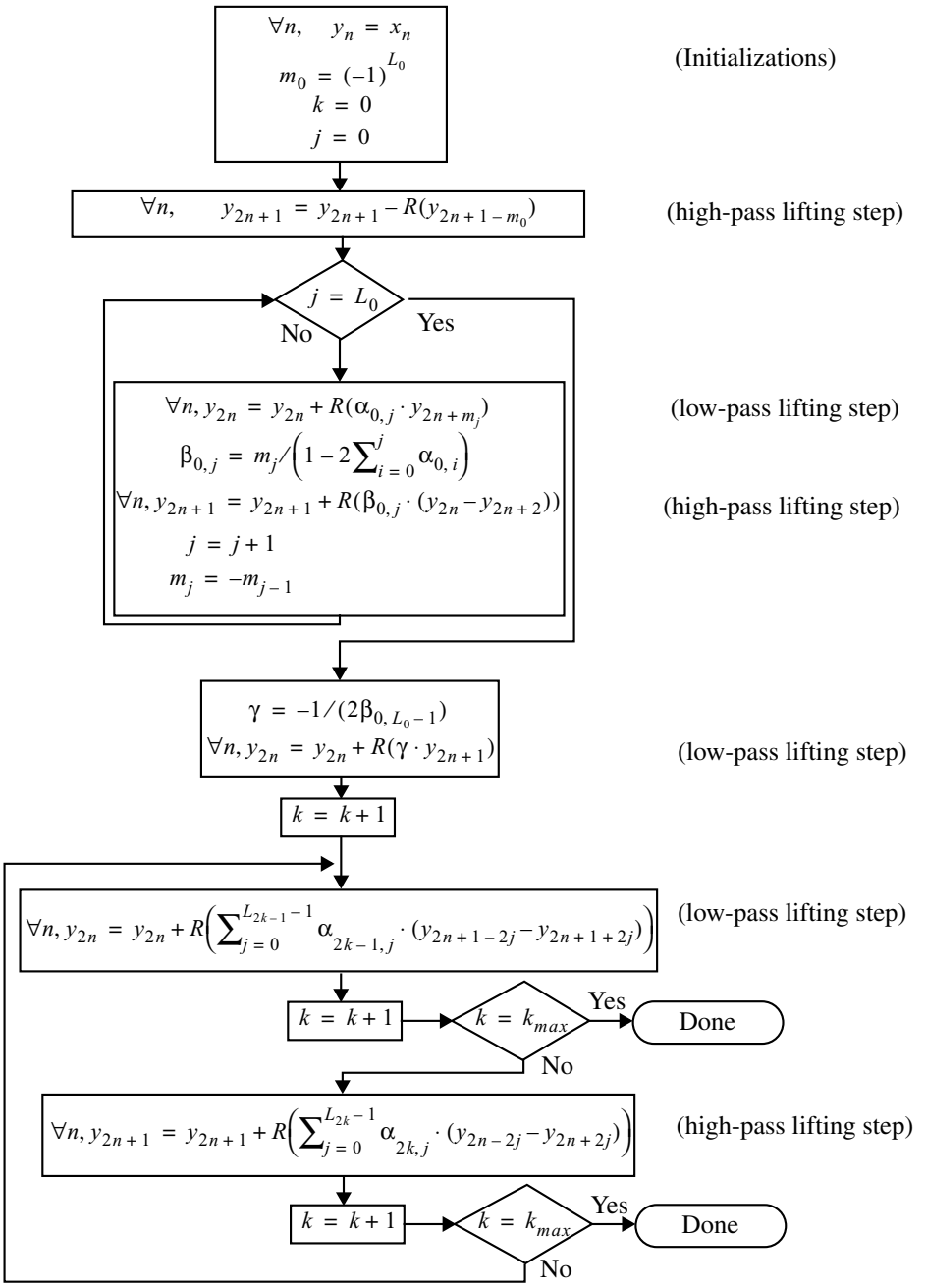

Figure G-13 — Lifting implementation for forward HS wavelet transformations

G.7 Examples of optional filter banks (informative)

G.7.1 WS Examples

G.7.1.1 The 1D_FILTR procedure for the CRF13-7 reversible wavelet transformation

For the CRF13-7 reversible wavelet transformation, the values of the filtering parameters $(\alpha_{0,0}, \alpha_{1,0}, \alpha_{0,1}, \alpha_{1,1}, \varepsilon_{R,0}, \varepsilon_{R,1})$ of the procedure specified in Annex G.5.3 are as follows:

$$\alpha_{0,0} = 5, \alpha_{1,0} = -1, \alpha_{0,1} = 9, \alpha_{1,1} = -1, \varepsilon_{0,0} = 4, \varepsilon_{R,1} = 4 \qquad \text{G.24}$$

G.7.2 HS Examples

G.7.2.1 The 1D_FILTR_ARB procedure for the Haar reversible wavelet transformation

The Haar reversible wavelet transformation requires three lifting steps, and no scaling step.

Equation G.25 specifies the first lifting step, which applies to all odd-indexed coefficients.

$$X(2n+1) = X(2n+1) - X(2n). \qquad \text{G.25}$$

Equation G.26 specifies the second lifting step, which applies to all even-indexed coefficients, and uses values calculated at the previous lifting step.

$$X(2n) = X(2n) + \left\lfloor \frac{X(2n+1)}{2} \right\rfloor. \qquad \text{G.26}$$

G.7.2.2 The 1D_FILTR_ARB procedure for the 2-10 reversible wavelet transformation

The 2-10 reversible wavelet transformation requires three lifting steps, and no scaling step.

Equation G.27 specifies the first lifting step, which applies to all odd-indexed coefficients.

$$X(2n+1) = X(2n+1) - X(2n). \qquad \text{G.27}$$

Equation G.28 specifies the second lifting step, which applies to all even-indexed coefficients, and uses values calculated at the previous lifting step.

$$X(2n) = X(2n) + \left\lfloor \frac{X(2n+1)}{2} \right\rfloor. \qquad \text{G.28}$$

Equation G.29 specifies the third lifting step, which applies to all odd-indexed coefficients, and uses values calculated at the previous lifting step

$$X_{ext}(2n+1) = X_{ext}(2n+1) - \left\lfloor \frac{\left(\sum_{s=0}^{1} \alpha_s \cdot (X_{ext}(2n+1-(2s+1)) + X_{ext}(2n+1+(2s+1))) \right) + \beta}{2^{\varepsilon}} \right\rfloor \quad , \qquad \text{G.29}$$

where $\alpha_0 = \quad , \alpha_1 = \quad , \alpha_{0,1} = \quad , \alpha_{1,1} = \quad , \varepsilon_{0,0} = \quad , \varepsilon_{R,1} = \quad , \beta$.

G.8 Overview of lifting step sequences (informative)

Filter banks are implemented as a sequence of lifting steps. A lifting step is a procedure that modifies either the even- or the odd-indexed samples in the input signal. A lifting step that modifies the even-indexed samples is referred to as a

lowpass lifting step, while a lifting step that modifies the odd-indexed samples is referred to as a *highpass* lifting step. The steps in a lifting sequence alternate between lowpass and highpass steps; the lowpass/highpass characteristic of step S_0 is signaled in the codestream [WHICH MARKER SEGMENT?!] by the parameter m_0 in Table G-1. The number of lifting steps is signaled in the parameter N_{LS}.

Two types of filter banks are defined in this standard: reversible and irreversible. This characteristic is signaled in the WT_Typ parameter in Table G-1. Reversible filter banks enable lossless compression whereas irreversible filter banks only support perfect reconstruction modulo finite precision effects (e.g., truncation or round off error).

A lifting step sequence for irreversible subband decomposition is shown in Figure G-14. The irreversible decomposition lifting sequence terminates with a gain scaling step, *G(K)*. The gain scaling step depends on a scaling parameter, *K*, which is signaled in the codestream; see Table G-1. The gain scaling step applies constant scaling gains to both the even- and odd-indexed subsequences.

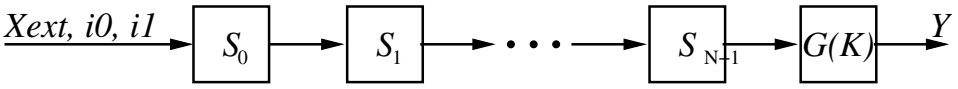

Figure G-14 — Lifting step sequence for irreversible subband decomposition.

A lifting step sequence for irreversible subband reconstruction is shown in Figure G-15. Note that the steps in the reconstruction sequence are mathematical inverses of the steps in the decomposition sequence, as indicated by the superscripts "-1". Also note that the order in which the reconstruction steps are applied is reversed from the order in which the decomposition lifting steps are applied. In particular, the irreversible reconstruction lifting sequence begins with the inverse gain scaling step, $G^{-1}(K)$.

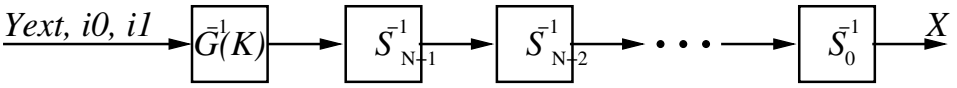

Figure G-15 — Lifting step sequence for irreversible subband reconstruction.

Decomposition and reconstruction lifting sequences for reversible filter banks are shown in Figure G-16 and Figure G-17. Note that *G(K)* and $G^{-1}(K)$ do not appear in these diagrams: lifting sequences for reversible filter banks do not have gain scaling steps.

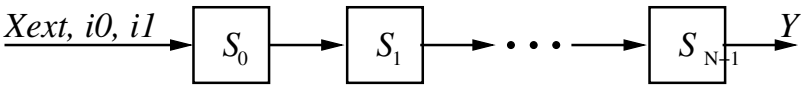

Figure G-16 — Lifting step sequence for reversible subband decomposition.

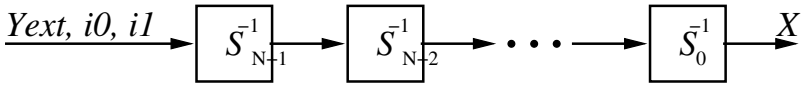

Figure G-17 — Lifting step sequence for reversible subband reconstruction.

More detail regarding lifting and scaling steps in an irreversible decomposition sequence is shown in Figure G-18. To clarify the description, the even- and odd-indexed samples in the input signal are depicted as parallel channels. This diagram depicts a lowpass lifting step followed by a highpass lifting step. The lowpass lifting step consists of applying the filter in the box labeled *alpha_s* to the odd-indexed subsequence and adding the result to the even-indexed

subsequence. The highpass lifting step consists of applying the filter in the box labeled alpha_{s+1} to the even-indexed subsequence and adding the result to the odd-indexed subsequence. The gain scaling step consists of multiplying the even-indexed subsequence by 1/K and multiplying the odd-indexed subsequence by K.

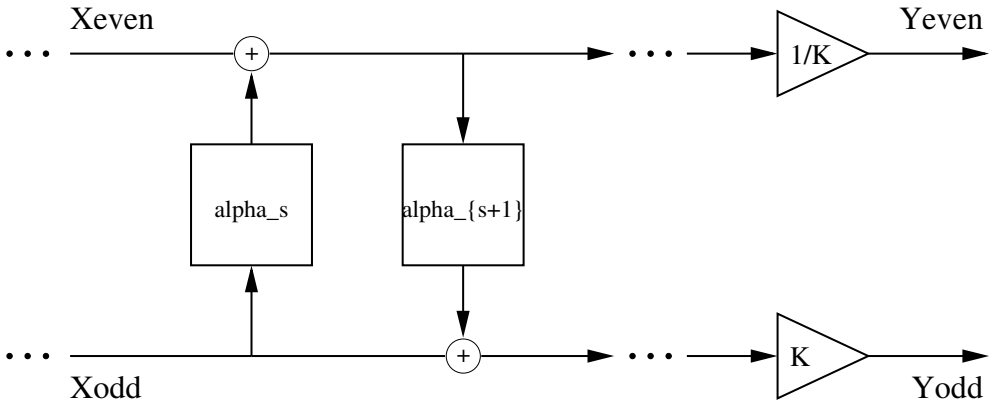

Figure G-18 — Detail of lowpass, highpass lifting steps and scaling step in an irreversible decomposition sequence.

A detail of the lifting and scaling steps in an irreversible reconstruction sequence is shown in Figure G-19. Note that the output of the filters alpha_s and alpha_{s+1} is *subtracted* from the even- and odd-indexed subsequences, and that the scaling gains are inverted.

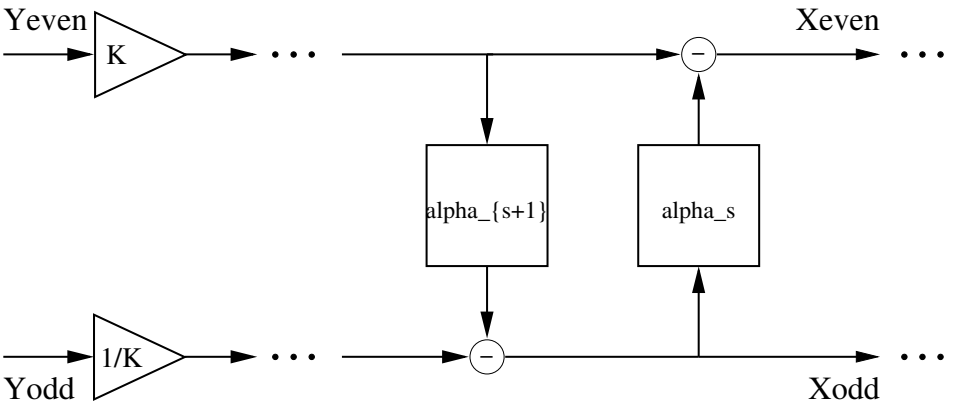

Figure G-19 — Detail of scaling step and lowpass, highpass lifting steps in an irreversible reconstruction

Annex H

Single sample overlap discrete wavelet transform, extensions

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1).

H.1 Single Sample Overlap Inverse Discrete Wavelet Transformation (SSO-IDWT)

The selection of the SSO option is signaled in the extended COD and COC markers (see Annex A). The SSO option is only applicable to wavelet transformations (reversible or irreversible) which use WS wavelet filters (i.e. Filt_cat=1).

H.1.1 Signalling

The additional parameters relevant to the selection of the SSO option are: XC, YC, z_x and z_y. The parameters XC, YC are signalled in the COD and COC extended marker segment (see Annex A.2.2). The parameters z_x and z_y are signaled in the Csiz marker parameter (see Annex A.2.1).

H.1.2 Modified procedures

The selection of the SSO option requires a modification of the 1D_FILTR filtering procedure described in Annex G.4.3 (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 ITU.T Rec. T.800 | IS 15444-1 Annex F.3.

H.1.3 The IDWT procedure

The IDWT procedure (illustrated in Figure H-1) 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 procedure (described in Annex H.1.4) is performed at every level *lev*, where the level *lev* decreases at each iteration, until N_L iterations are performed. The 2D_SR procedure is iterated over the *lev*LL subband produced at each iteration. Finally, the subband $a_{0LL}(u_{0LL}, v_{0LL})$ is the output array $I(x, y)$.

H.1.4 The 2D_SR Procedure

The 2D_SR procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.3.2, except for the addition of the parameters XC_{lev}, YC_{lev} (see Figure H-2), which are respectively used by the HOR_SR and VER_SR procedures (see Annex H.1.5 and Annex H.1.6).

H.1.5 The HOR_SR procedure

The HOR_SR procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.3.4, except for the addition of the parameter xC, which is used by the 1D_SR procedure (see Annex H.1.7).

H.1.6 The VER_SR procedure

The VER_SR procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.3.5, except for the addition of the parameter yC, which is used by the 1D_SR procedure (see Annex H.1.7).

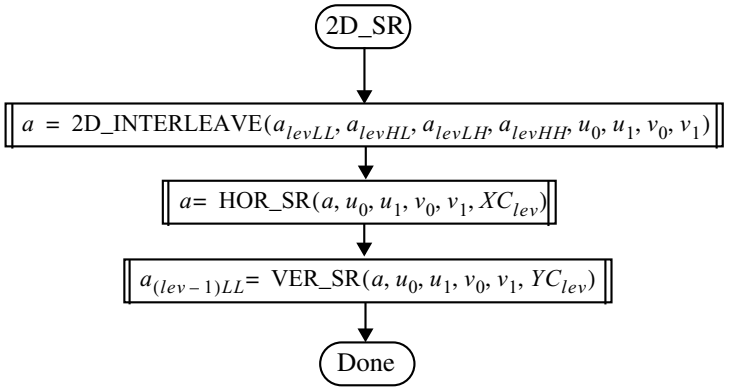

Figure H-2 — The 2D_SR procedure

H.1.7 The 1D_SR procedure

The 1D_SR procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex 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 Annex H.1.8). The parameter dC is either the parameter xC (if called by the HOR_SR procedure) or the parameter yC (if called by the VER_SR procedure).

H.1.8 The 1D_FILTR_SSO procedure

The 1D_FILTR_SSO procedure is a modification of the 1D_FILTR procedure described in Annex G.3.2. The input and output parameters of the 1D_FILTR_SSO procedure are given in Figure H-3.

Let k_0 be defined by

$$k_0 = \left\lfloor \frac{i_0}{dC} \right\rfloor \qquad \text{H.1}$$

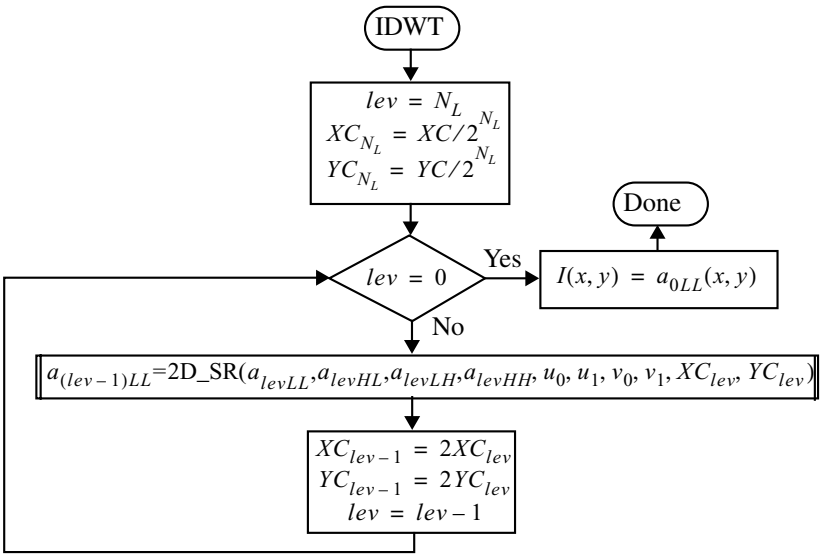

Figure H-1 — The IDWT Procedure

and N_I is defined by:

$$N_I = \left\lceil \frac{i_1}{dC} \right\rceil - \left\lfloor \frac{i_0}{dC} \right\rfloor \qquad \text{H.2}$$

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:

$$n_0 = i_0\,,\ n_{N_I} = i_1 - 1 \text{ and } n_p = (k_0 + p)dC \text{ for } p = 1, \ldots, N_I - 1\,, \qquad \text{H.3}$$

For an index $i \in I_p$, define the function $PSE_{O,p}(i)$ as

$$PSE_{O,p}(i) = PSE_O(i, n_p, n_{p+1} + 1)\,, \qquad \text{H.4}$$

where the function $PSE_O(i, i_0, i_1)$ is defined in ITU-T T.800 | IS 15444-1 Equation G.1.

H.1.8.1 Reversible transformations

This section specifies for reversible transformations the modifications of each lifting step as specified in Equation G.4. The modification of Equation G.4 ensures that each coefficient $V_{ext}(2n + m_s)$ 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 of $V_{ext}(n_p)$ for $p = 0, 1, \ldots, N_I$, if any, remain unmodified, while all other values $V_{ext}(2n + m_s)$ (i.e. for which $2n + m_s$ belongs to a single interval I_p) are modified according to Equation H.5:

$$V_{ext}(2n + m_s) = V_{ext}(2n + m_s) - \left\lfloor \frac{\left(\sum_{k = off_s}^{off_s + L_s - 1} \alpha_{s,k} \cdot V_{ext}(PSE_{O,p}(2n + 1 - m_s + 2k)) \right) + \beta_s}{2^{e_s}} \right\rfloor . \qquad \text{H.5}$$

H.1.8.2 Irreversible Transformations

This section specifies for irreversible transformations the modifications of each lifting step as specified in Equation G.14. The steps specified in Equation G.12 and Equation G.13 are not modified. The modification of Equation G.14 ensures that each coefficient $V_{ext}(2n + m_s)$ 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 of $V_{ext}(n_p)$ for $p = 0, 1, \ldots, N_I$, if any, are modified according to Equation H.6:

$$V_{ext}(n_p) = (1/B_s)V_{ext}(n_p)\,, \qquad \text{H.6}$$

where B_s is defined in Equation G.2, while all other values $V_{ext}(2n + m_s)$ (i.e. for which $2n + m_s$ belongs to a single interval I_p) are modified according to Equation H.7:

Figure H-3 Parameters of the 1D_FILTR_SSO procedure

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) - \left(\sum_{k=off_s}^{off_s+L_{s-1}} \alpha_{s,k} \cdot V_{ext}(PSE_{O,p}(2n+1-m_s+2k)) \right). \quad \text{H.7}$$

H.2 Single Sample Overlap Forward Discrete Wavelet Transformation (SSO-FDWT) (informative)

The selection of the SSO option is signaled in the extended COD and COC markers (see Annex A). The SSO option is only applicable to wavelet transformations (reversible or irreversible) which use WSS wavelet filters (i.e. Filt_cat=1).

H.2.1 Modified procedures

The selection of the SSO option requires a modification of the 1D_FILTD filtering procedure described in Annex G.4.3 (the 1D_FILTD_SSO procedure), as well as a modification of the FDWT, 2D_SD, HOR_SD, VER_SD and 1D_SD procedures described in ITU.T Rec. T.800 | IS 15444-1 Annex F.4.

H.2.2 The FDWT procedure

The FDWT procedure (illustrated in Figure H-4) 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. The 2D_SD procedure (described in Annex H.2.3) is performed at every level *lev*, where the level *lev* increases at each iteration, until N_L iterations are performed.

H.2.3 The 2D_SD Procedure

The 2D_SD procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.4.2, except for the addition of the parameters XC_{lev}, YC_{lev} (see Figure H-2), which are respectively used by the HOR_SD and VER_SD procedures (see Annex H.2.4 and Annex H.2.5).

H.2.4 The HOR_SD procedure

The HOR_SD procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.4.4, except for the addition of the parameter xC, which is used by the 1D_SD procedure (see Annex H.2.6).

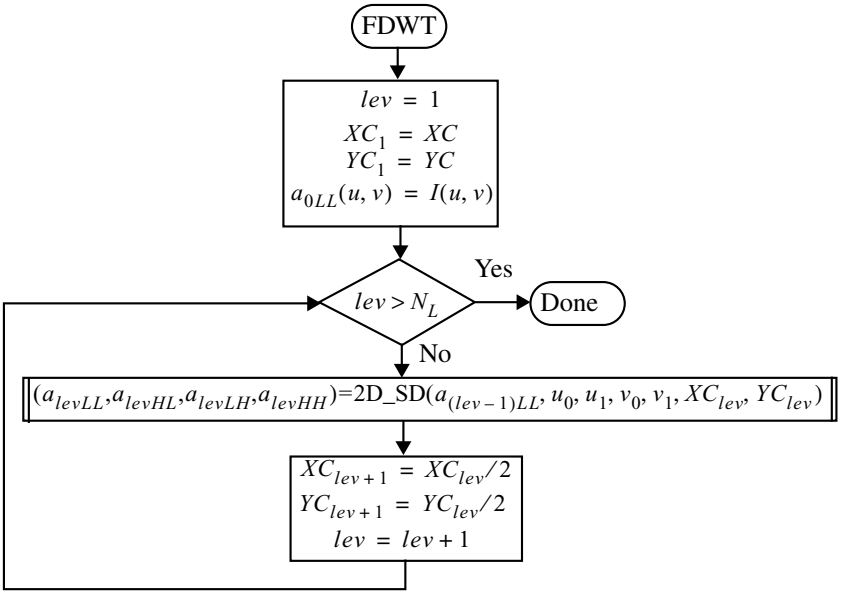

Figure H-4 — The FDWT procedure

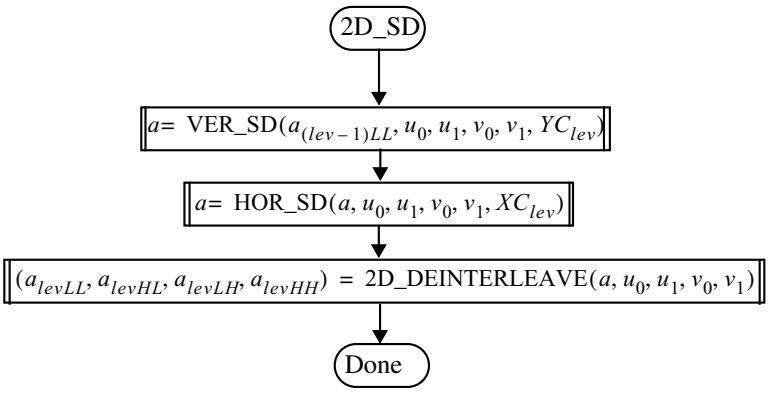

Figure H-5 — The 2D_SD procedure

H.2.5 The VER_SD procedure

The VER_SD procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex F.4.5, except for the addition of the parameter yC, which is used by the 1D_SD procedure (see Annex H.2.6).

H.2.6 The 1D_SD procedure

The 1D_SD procedure is identical to the one described in ITU.T Rec. T.800 | IS 15444-1 Annex 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 procedure by the 1D_FILTD_SSO procedure (see Annex H.1.8). The parameter dC is either the parameter xC (if called by the HOR_SD procedure) or the parameter yC (if called by the VER_SD procedure).

H.2.7 The 1D_FILTD_SSO procedure

The 1D_FILTD_SSO procedure is a modification of the 1D_FILTD procedure described in Annex G.3.2. The input and output parameters of the 1D_FILTD_SSO procedure are given in Figure H-6.

H.2.7.1 Reversible transformations

This section specifies for reversible transformations the modifications of each lifting step as specified in Equation G.18. The modification of Equation G.18 ensures that each coefficient $V_{ext}(2n+m_s)$ 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 of $V_{ext}(n_p)$ for $p = 1, \ldots, N_I - 1$, if any, remain unmodified, while all other values $V_{ext}(2n+m_s)$ (i.e. for which $2n+m_s$ belongs to a unique interval I_p) are modified according to Equation H.5:

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) + \left\lfloor \frac{\left(\sum_{k=off_s}^{off_s+L_s-1} \alpha_{s,k} \cdot V_{ext}(PSE_{O,p}(2n+1-m_s+2k)) \right) + \beta_s}{2^{\varepsilon_s}} \right\rfloor . \qquad \text{H.8}$$

Figure H-6 Parameters of the 1D_FILTD_SSO procedure

H.2.7.2 Irreversible Transformations

This section specifies for irreversible transformations the modifications of each lifting step as specified in Equation G.19. The modification of Equation G.19 ensures that each coefficient $V_{ext}(2n+m_s)$ 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 of $V_{ext}(n_p)$ for $p = 1, \ldots, N_I - 1$, if any, are modified according to Equation H.9:

$$V_{ext}(n_p) = B_s V_{ext}(n_p), \tag{H.9}$$

where B_s is defined in Equation G.2, while all other values $V_{ext}(2n+m_s)$ (i.e. for which $2n+m_s$ belongs to a single interval I_p) are modified according to Equation H.10:

$$V_{ext}(2n+m_s) = V_{ext}(2n+m_s) + \left(\sum_{k=off_s}^{off_s+L_{s-1}} \alpha_{s,k} \cdot V_{ext}(PSE_{O,p}(2n+1-m_s+2k)) \right). \tag{H.10}$$

H.3 Selection of Single Sample Overlap Parameters

The selection of the SSO option enables a low-memory block-based implementation of the wavelet transformations, both forward and inverse: the 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 option are: XC, YC, z_x and z_y.

H.3.1 Division of image tile components into overlapping SSO blocks

SSO blocks are of width XC+1 and height YC+1. The position of SSO blocks is independent of the position parameters z_x and z_y. 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-6). Two neighboring SSO blocks overlap by either one row of samples (vertical neighbors), one column of samples (horizontal neighbors), or just one sample (diagonal neighbors).

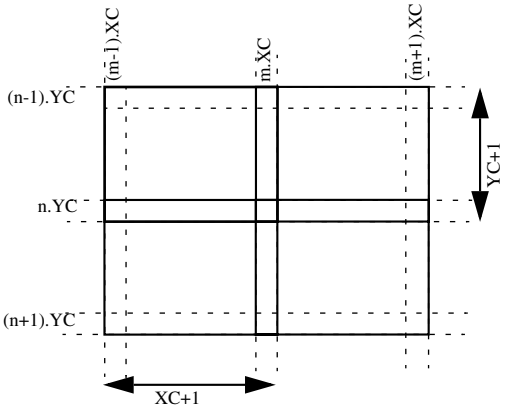

Figure H-7 — Position of SSO blocks

H.3.2 Selection of tile parameters

To maximize coding efficiency, the following selection of tile parameters is recommended: $XTOsiz = z_x$, $YTOsiz = z_y$, $\mathrm{mod}(XTsiz, XC) = 0$ and $\mathrm{mod}(YTsiz, YC) = 0$.

H.4 SSO for Tiles Inverse Discrete Wavelet Transformation for (TSSO-IDWT)

H.4.1 Signalling

The selection of the TSSO option is signaled in the extended COD and COC markers (see Annex A). The TSSO option is only applicable to wavelet transformations (reversible or irreversible) which use WS wavelet filters (i.e. Filt_cat=1).

The additional parameters relevant to the selection of the TSSO option are: $XTsiz$, $YTsiz$, z_x and z_y. The parameters XC, YC are signalled in the COD and COC extended marker segments (see Annex A.2.2). The parameters z_x and z_y are signaled in the Csiz marker (see Annex A.2.1).

H.4.2 Partitioning of the image into single-sample overlapping tiles

SSO tiles are of width $XTsiz$ and height $YTsiz$. The position of SSO tiles is independent of the parameters z_x and z_y. The first and last row of an SSO tile are always located at multiples of $YT = YTsiz - 1$, while the first and last column of an SSO tile are always located at multiples of $XT = XTsiz - 1$ (see Figure H-8). Two neighboring SSO tiles overlap by either one row of samples (vertical neighbors), one column of samples (horizontal neighbors), or just one sample (diagonal neighbors).

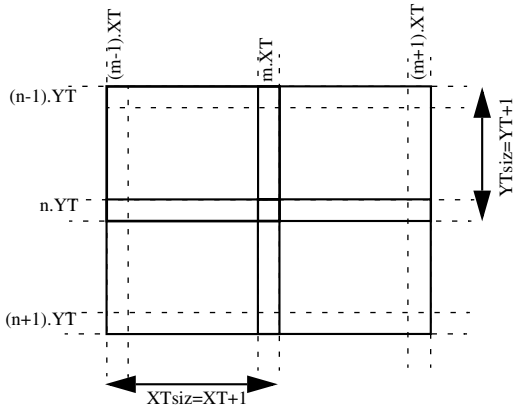

Figure H-8 — Position of SSO tiles

The TSSO-DWT uses a modified version of the 1D_FILTR procedure. The tile parameters $XTsiz$ and $YTsiz$ must be a power of two plus one (i.e. XT and YT must be a power of two), and the tile parameters $XTOsiz$ and $YTOsiz$ must satisfy: $XTOsiz = z_x$, $YTOsiz = z_y$.

H.4.3 Modified procedures

The selection of the SSO option requires a modification of the IDWT procedure described in ITU.T Rec. T.800 | IS 15444-1 Annex F.3 and uses the 2D_SR, HOR_SR, VER_SR and 1D_SR, and 1D_FILTR_SSO procedures described in Annex H.1.

H.4.4 The IDWT_TSSO procedure

The IDWT_TSSO procedure (illustrated in Figure H-1) is the modification of the IDWT procedure described in ITU.T Rec. T.800 | IS 15444-1 Annex F.3, and starts with the initialization of the variable lev (the current decomposition level) to N_L, of the variable XT_{N_L} to $XT/2^{N_L}$ and of the variable YT_{N_L} to $YT/2^{N_L}$. The 2D_SR procedure (described in Annex H.1.4) is performed at every level *lev*, where the level *lev* decreases at each iteration, until N_L iterations are

performed. The 2D_SR procedure is iterated over the *lev*LL subband produced at each iteration. Finally, the subband $a_{0LL}(u_{0LL}, v_{0LL})$ is the output array $I(x, y)$.

H.5 SSO for Tiles Forward Discrete Wavelet Transformation (TSSO-FDWT)

The selection of the TSSO option is signaled in the extended COD and COC markers (see Annex A). The SSO option is only applicable to wavelet transformations (reversible or irreversible) which use WSS wavelet filters (i.e. Filt_cat=1).

H.5.1 Modified procedures

The selection of the TSSO option requires a modification of the FDWT filtering procedure described in Annex G.4.3 (the FDWT_TSSO procedure), and uses the 2D_SD, HOR_SD, VER_SD, 1D_SD and 1D_FILTD_SSO procedures described in Annex H.1.

H.5.2 The FDWT_TSSO procedure

The FDWT_TSSO procedure (illustrated in Figure H-4) starts with the initialization of the variable lev (the current decomposition level) to 1, of the variable XT_1 to XT and of the variable YT_1 to YT. The 2D_SD procedure (described

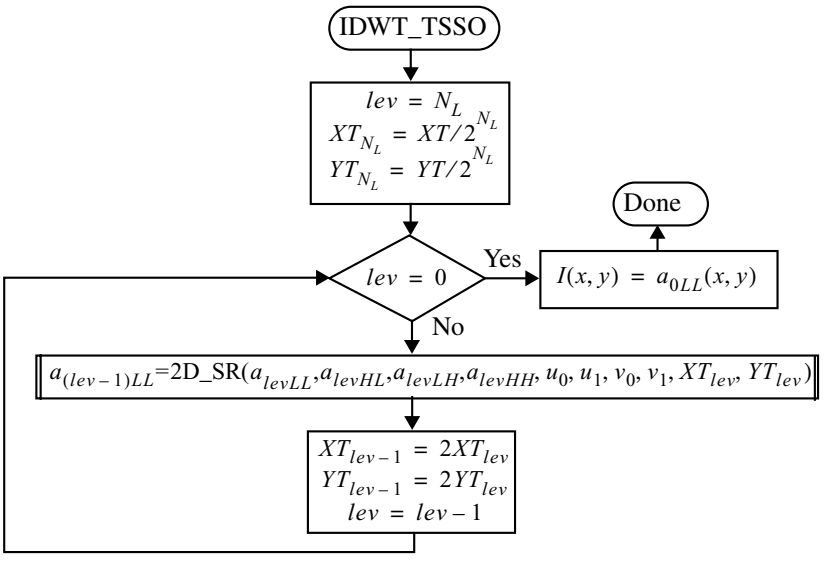

Figure H-9 — The IDWT_TSSO Procedure

in Annex H.2.3) is performed at every level *lev*, where the level *lev* increases at each iteration, until N_L iterations are performed.

H.5.3 Reconstruction of images samples from reconstructed tiles.

Since the reconstructed tiles overlap by one row and one column with neighboring tiles, some images samples will be reconstructed two or four times in different tiles. For any such sample, one must use the value of the sample reconstructed by the first tile, with respect to the tile ordering specified in ITU.T Rec. T.800 | IS 15444-1 Annex B.3.

H.6 SSO Examples (informative)

H.6.1 Illustration in the case of the 5-3 forward reversible transformation

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) - \left\lfloor \frac{V_{ext}(2n) + V_{ext}(2n+2)}{2} \right\rfloor \text{ for } i_0 < 2n+1 < i_1 - 1 \qquad \text{H.11}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n+2) \text{ for } 2n+1 = i_0\,, \qquad \text{H.12}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1\,. \qquad \text{H.13}$$

The second lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n-1) + V_{ext}(2n+1) + 2}{4} \right\rfloor \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0\,, \qquad \text{H.14}$$

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n+1) + 1}{2} \right\rfloor \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0\,, \qquad \text{H.15}$$

$$V_{ext}(2n) = V_{ext}(2n) + \left\lfloor \frac{V_{ext}(2n-1) + 1}{2} \right\rfloor \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0\,, \qquad \text{H.16}$$

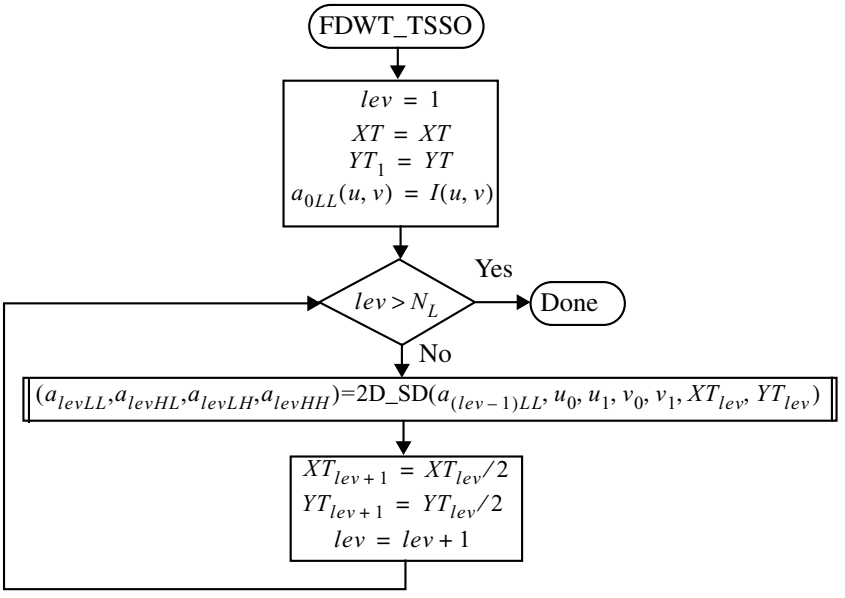

Figure H-10 — The FDWT_TSSO procedure

$$\text{and } V_{ext}(2n) = V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \ . \tag{H.17}$$

H.6.2 Illustration in the case of the 5-3 forward irreversible transformation

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) - \left(\frac{V_{ext}(2n) + V_{ext}(2n+2)}{2} \right) \text{ for } i_0 < 2n+1 < i_1 - 1 \ , \tag{H.18}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \ , \tag{H.19}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) - V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \ . \tag{H.20}$$

The second lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \left(\frac{V_{ext}(2n-1) + V_{ext}(2n+1)}{4} \right) \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.21}$$

$$V_{ext}(2n) = V_{ext}(2n) + \frac{V_{ext}(2n+1)}{2} \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.22}$$

$$V_{ext}(2n) = V_{ext}(2n) + \frac{V_{ext}(2n-1)}{2} \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.23}$$

$$\text{and } V_{ext}(2n) = V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \ . \tag{H.24}$$

H.6.3 Illustration in the case of the 9-7 forward irreversible transformation

The first lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) + \alpha(V_{ext}(2n) + V_{ext}(2n+2)) \text{ for } i_0 < 2n+1 < i_1 - 1 \ , \tag{H.25}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) + 2\alpha V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \ , \tag{H.26}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) + 2\alpha V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 \ . \tag{H.27}$$

The second lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \beta(V_{ext}(2n-1) + V_{ext}(2n+1)) \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.28}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\beta V_{ext}(2n+1) \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.29}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\beta V_{ext}(2n+1) \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 \ , \tag{H.30}$$

$$\text{and } V_{ext}(2n) = (1 + 2\beta) V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 \ . \tag{H.31}$$

The third lifting step is:

$$V_{ext}(2n+1) = V_{ext}(2n+1) + \gamma(V_{ext}(2n) + V_{ext}(2n+2)) \text{ for } i_0 < 2n+1 < i_1 - 1 \ , \tag{H.32}$$

$$V_{ext}(2n+1) = V_{ext}(2n+1) + 2\gamma V_{ext}(2n+2) \text{ for } 2n+1 = i_0 \ , \tag{H.33}$$

$$\text{and } V_{ext}(2n+1) = V_{ext}(2n+1) + 2\gamma V_{ext}(2n) \text{ for } 2n+1 = i_1 - 1 . \quad \text{H.34}$$

The fourth lifting step is:

$$V_{ext}(2n) = V_{ext}(2n) + \delta(V_{ext}(2n-1) + V_{ext}(2n+1)) \text{ for } i_0 < 2n < i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 , \quad \text{H.35}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\delta V_{ext}(2n+1) \text{ for } 2n = i_0 \text{ and } \mathrm{mod}(2n, dC) \neq 0 , \quad \text{H.36}$$

$$V_{ext}(2n) = V_{ext}(2n) + 2\delta V_{ext}(2n+1) \text{ for } 2n = i_1 - 1 \text{ and } \mathrm{mod}(2n, dC) \neq 0 , \quad \text{H.37}$$

$$\text{and } V_{ext}(2n) = (1 + 2\beta(1 + 2\alpha) + 2\delta(1 + 2\gamma(1 + 2\beta(1 + 2\alpha))))V_{ext}(2n) \text{ for } \mathrm{mod}(2n, dC) = 0 . \quad \text{H.38}$$

The scaling steps are the same for all coefficients.

Annex I

Multiple component transformations, extension

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1). The techniques presented in this chapter may not be used in conjunction with those found in Annex B.

This Annex specifies a multiple component transformations. The most common multi-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 six components 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 techniques presented in this Annex. The first is a matrix-based multiple component transform which forms linear combinations of components to reduce the correlation of each component. This transform structure permits simple component prediction transforms and includes more complex transforms such as the Karhunen-Lo ve Transform (KLT). The second decorrelation technique is a wavelet-based decorrelation transform.

I.1 Multiple component transformation

The use of the multiple component transformation is signalled in the Rsiz parameter (see Annex A.2.1).

I.1.1 Image component reconstruction

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 both inverse decorrelation transformation (e.g. KLT, etc.) and inverse dependency transformation (e.g. linear prediction, etc.). Figure I-1 illustrates the inverse multiple component transform processing steps and matrices needed to reconstruct image components from the codestream. The generic reconstruction information, which will be referred to as component reconstruction matrices, is shown in Figure I-2. Each multiple component sample is reconstructed by applying the reconstruction matrices included in the codestream. All matrix multiplications are applied from the left-hand side, thus a multiple component sample forms a column vector. It is the duty of the encoder to properly define the component reconstruction matrices via the MCT marker segments (see Annex A.3.7). The four component reconstruction matrices; decorrelation transform, decorrelation offset, dependency transform, and dependency offset, describe the appropriate *inverse* transformation process. For brevity s sake the word inverse has been omitted from their names.

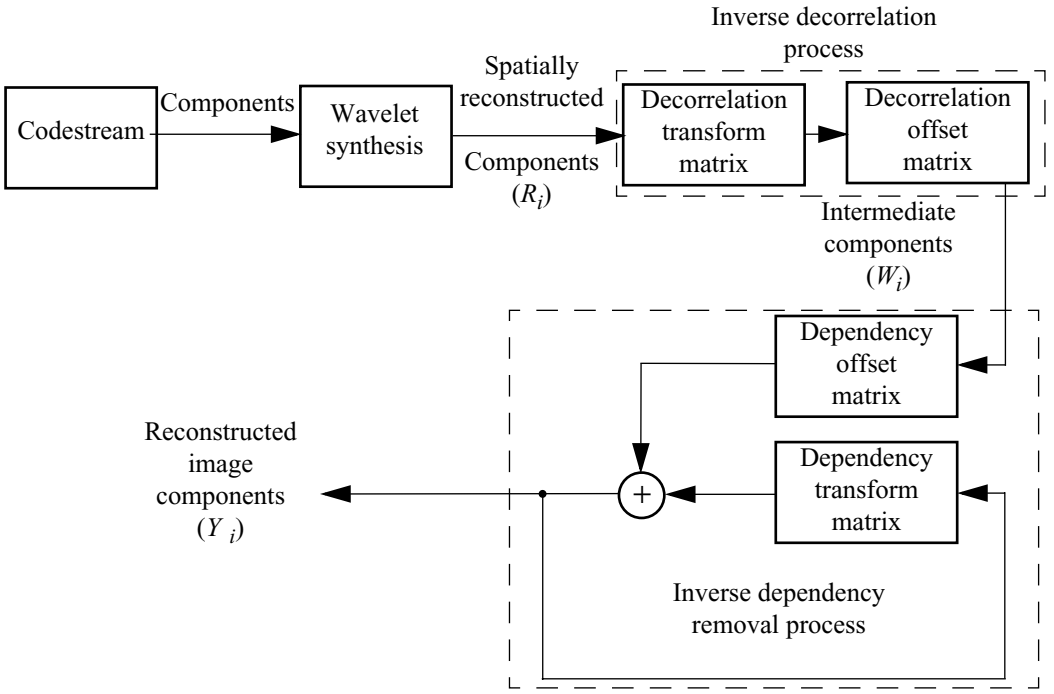

Figure I-1 Inverse multiple component transform processing

It is possible that not all of the component reconstruction matrices will be needed during the reconstruction process. For example, in the case that only a linear prediction has been applied to the image components, the decorrelation transform and decorrelation offset matrices would not be needed. The multiple component transform syntax allows for one or more of the component reconstruction matrices to be specified as null. If a matrix value is unspecified during the construction process, it is understood to be the identity matrix, in the case of the decorrelation transform, and the zero matrix in the case of the dependency transform, decorrelation offset and dependency offset matrices.

The decorrelation transform matrix contains weights to apply to the spatially reconstructed components, R_i, to reconstruct an intermediate component, W_i. The coefficients are subscripted as t_{ij}, where i is the row index and j is the column index. In addition, a decorrelation offset may be added to each spatially reconstructed component after application of the decorrelation transform. These coefficients are subscripted as m_i where i is a row index. Each row index indirectly specifies the intermediate component to be reconstructed; each column index indirectly specifies a particular spatially reconstructed component. Spatially reconstructed components are associated with the columns of the decorrelation transform matrix by the input component collection list in the MCC marker segment ($Cmcc^i$). Intermediate components are associated with the rows of the decorrelation transform matrix by the output component collection list in the MCC marker segment ($Wmcc^i$).

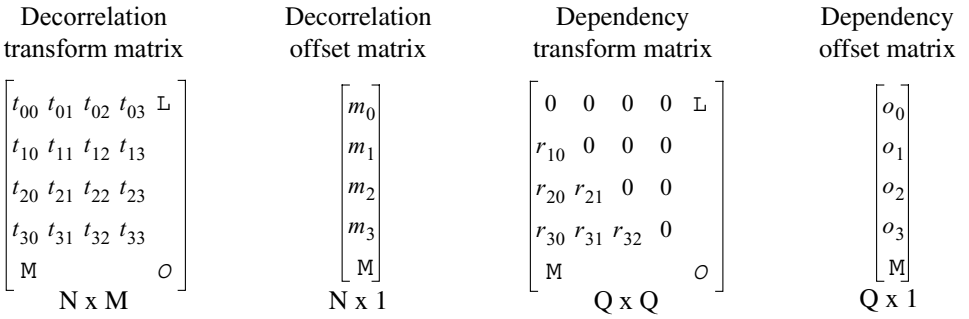

Figure I-2 Inverse multiple component transform matrices

Component reordering is allowed on both the input and output component collections in the MCC. The union of the input component collections within the MCC must include all components in the codestream. The application of the decorrelation transform and decorrelation offset matrices to the spatially reconstructed components is given by Equation I.1. For simplicity, no component reordering has been performed in this equation. A detailed example is presented in Annex I.1.3 where component reordering is employed. Equation I.1 has been rewritten in matrix form in Equation I.2. It is possible for the number of spatially reconstructed components used to differ from the number of intermediate components formed (e.g. dropped KLT components).

$$\begin{aligned} W_0 &= t_{00}R_0 + m_0 + t_{01}R_1 + m_1 + t_{02}R_2 + m_2 + t_{03}R_3 + m_3 + \mathtt{L} \\ W_1 &= t_{10}R_0 + m_0 + t_{11}R_1 + m_1 + t_{12}R_2 + m_2 + t_{13}R_3 + m_3 + \mathtt{L} \\ W_2 &= t_{20}R_0 + m_0 + t_{21}R_1 + m_1 + t_{22}R_2 + m_2 + t_{23}R_3 + m_3 + \mathtt{L} \\ W_3 &= t_{30}R_0 + m_0 + t_{31}R_1 + m_1 + t_{32}R_2 + m_2 + t_{33}R_3 + m_3 + \mathtt{L} \\ &\mathtt{M} \end{aligned} \qquad \text{I.1}$$

$$\begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \\ \mathtt{M} \end{bmatrix} = \begin{bmatrix} t_{00} & t_{01} & t_{02} & t_{03} & \mathtt{L} \\ t_{10} & t_{11} & t_{12} & t_{13} & \\ t_{20} & t_{21} & t_{22} & t_{23} & \\ t_{30} & t_{31} & t_{32} & t_{33} & \\ \mathtt{M} & & & & O \end{bmatrix} \begin{bmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \\ \mathtt{M} \end{bmatrix} + \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ \mathtt{M} \end{bmatrix} \qquad \text{I.2}$$

The dependency transform matrix contains weights to apply to each of the available intermediate components in order to reconstruct an image component, Y_i. Since this is a reconstruction process where initially at least one image component must have no spectral dependencies, and other components can be computed only after their dependants are fully computed, there are some restrictions on the form of this matrix as shown in Figure I-2. In particular, the dependency transform matrix must be square in dimension. Also, all non-zero values contained therein must appear only below the diagonal. This limit is imposed by the causality of the dependency transform process.

The coefficients of the matrix are subscripted as r_{ij}, where i is the row index and j is the column index. In addition, a dependency offset may be added to the image components during the reconstruction process. These coefficients are subscripted as o_i where i is a row index. Each row index indirectly specifies the image component to be reconstructed; each column index indirectly specifies the particular intermediate component required for reconstruction. Intermediate components are associated with columns of the dependency transform matrix by the input component collections lists in the MIC marker segment. Reconstructed image components are associated with the rows of the dependency transform and dependency offset matrices by the input component collection list in the MIC marker segment. The input and output component collections of the MCC and MIC marker segments jointly determine the final ordering of the reconstructed image components. The application of the dependency transform and dependency offset matrices to the intermediate

components is given by Equation I.3. The same equation is expressed in matrix form in Equation I.4. Again, for the sake of simplicity, these equations do not perform any component reordering. The example in Annex I.1.3 will illustrate this capability.

$$\begin{aligned} Y'_0 &= W_0 + o_0 \\ Y'_1 &= r_{10}Y'_0 + W_1 + o_1 \\ Y'_2 &= r_{20}Y'_0 + r_{21}Y'_1 + W_2 + o_2 \\ Y'_3 &= r_{30}Y'_0 + r_{31}Y'_1 + r_{32}Y'_2 + W_3 + o_3 \\ &\text{M} \end{aligned} \qquad \text{I.3}$$

$$\begin{bmatrix} Y'_0 \\ Y'_1 \\ Y'_2 \\ Y'_3 \\ \text{M} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & \text{L} \\ r_{10} & 0 & 0 & 0 & \\ r_{20} & r_{21} & 0 & 0 & \\ r_{30} & r_{31} & r_{32} & 0 & \\ \text{M} & & & & O \end{bmatrix} \begin{bmatrix} Y'_0 \\ Y'_1 \\ Y'_2 \\ Y'_3 \\ \text{M} \end{bmatrix} + \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \\ \text{M} \end{bmatrix} + \begin{bmatrix} o_0 \\ o_1 \\ o_2 \\ o_3 \\ \text{M} \end{bmatrix} \qquad \text{I.4}$$

Given the spectral reconstruction matrices described above and R_i, the spatially reconstructed components, Equation I.5 defines Y'_i, the spatially and spectrally reconstructed image components by combining Equation I.1 and Equation I.3:

$$\begin{aligned} Y'_0 &= t_{00}R_0 + m_0 + t_{01}R_1 + m_1 + t_{02}R_2 + m_2 + t_{03}R_3 + m_3 + \text{L} + o_0 \\ Y'_1 &= r_{10}Y'_0 + t_{10}R_0 + m_0 + t_{11}R_1 + m_1 + t_{12}R_2 + m_2 + t_{13}R_3 + m_3 + \text{L} + o_1 \\ Y'_2 &= r_{20}Y'_0 + r_{21}Y'_1 + t_{20}R_0 + m_0 + t_{21}R_1 + m_1 + t_{22}R_2 + m_2 + t_{23}R_3 + m_3 + \text{L} + o_2 \\ Y'_3 &= r_{30}Y'_0 + r_{31}Y'_1 + r_{32}Y'_2 + t_{30}R_0 + m_0 + t_{31}R_1 + m_1 + t_{32}R_2 + m_2 + t_{33}R_3 + m_3 + \text{L} + o_3 \\ &\text{M} \end{aligned} \qquad \text{I.5}$$

It is possible that an MCC or MIC marker segment refers to an input or output component whose value at that point is undefined. If either of these marker segments makes reference to an input component that does not exist (a spatially reconstructed component or intermediate component), that component should be treated like a null component (i.e. filled with zeros). The MCC marker segment may generate null output components as it reorders intermediate components. This is might be done to create space in the final reconstructed image component ordering for new components created by the inverse dependency transformation process. (The example in Annex I.1.3 illustrates this procedure.) An MIC marker segment should not generate null output components, although this standard does not expressly forbid it. The utility of null reconstructed image components is questionable and if an MIC marker segment does create such components, there is no requirement that a decoder produce them.

I.1.2 Bit depth and marker interpretation

This multiple component transform mechanism does not place restrictions on the bit depths of the reconstructed image components. Furthermore, it is possible for the number of components in the codestream to differ from the number of reconstructed image components. Therefore, when using the multiple component transformation mechanism, a CBD marker segment (see Annex A.3.6) shall be used. This marker segment indicates the total number of reconstructed image components and their respective bit depths after the inverse multiple component transform is applied.

When multiple component transformation processing is used, the SIZ marker segment (see ITU-T T.800 | IS 15444-1 Annex A.5.1 and Annex A.2.1, this document) shall indicate the number and bit depths of the components in the codestream. This interpretation of the SIZ marker segment is subtly different from its interpretation under other decoding processes in this standard and ITU-T T.800 | IS 15444-1, where it is used to indicate the number of reconstructed image components and their bit depths.

I.1.3 Matrix-based multiple component transform example (informative)

The example presented here exercises much of the flexibility and functionality on the multiple component transformation processes presented in this Annex and their supporting syntax in Annex A. We consider here not only the necessary codestream syntax needed by the decoder to properly interpret the codestream, but also the encoding decisions an encoder might face.

Our hypothetical multiple component image we wish to encode is a seven component multispectral image. Figure I-3 shows the components that are encoded in the codestream for this example. The encoder has analysed the multiple component image and decided that components 0, 1, 2, and 4 would be processed with a matrix-based decorrelation transform, components 5 and 3 with a dependency transform, and component 6 with no transform. Furthermore, the encoder decided to predict component 3 from component 5 in the dependency transform; only the residual prediction errors for component 3 are present in the codestream.

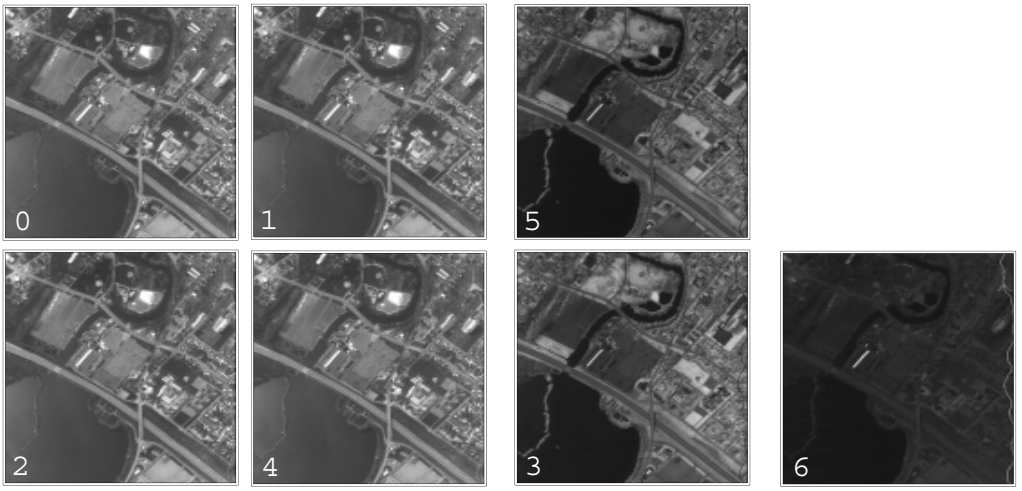

Figure I-3 Codestream components

To make the example more interesting, we will have the encoder place information for the creation of four additional components (a total of 11). These components include a one component panchromatic representation of the multispectral image and a three component false colour representation of the multispectral image. Although there are only seven components encoded in the codestream, the decoder will produce eleven. We will further require that these four additional components be the first four components in the reconstructed multiple component image. One might imagine that at a higher file format level, we could signal the significance of the first four components and state that decoding these components is optional. Such information would enable the decoder to select which representation of the image is desired; panchromatic, false colour, or full multispectral. However, inclusion of metadata to enable this functionality is beyond the scope of this standard.

Figure I-4 shows the portion of the inverse decorrelation transform due to the original seven component multispectral image. Since we do not yet know where the four additional components will be created (either in the inverse decorrelation transform or inverse dependency transform), the form of the decorrelation transform matrix is not yet certain. If one looks at the numbering of the intermediate components, W_i, we see that space has been made for the four additional components (W_0-W_3). The intermediate components generated by this inverse decorrelation transform are really reconstructed image components (Y'_i) since none of them will be further modified by the subsequent inverse dependency transform. On the right-hand side of Figure I-4, the MCC component collection parameters are given for the

inverse decorrelation transform. To complete the specification of the transform, MCT marker segments must be present in the main or appropriate first tile-part header and contain the decorrelation transform matrix and decorrelation offset matrix (see Annex A.3.7).

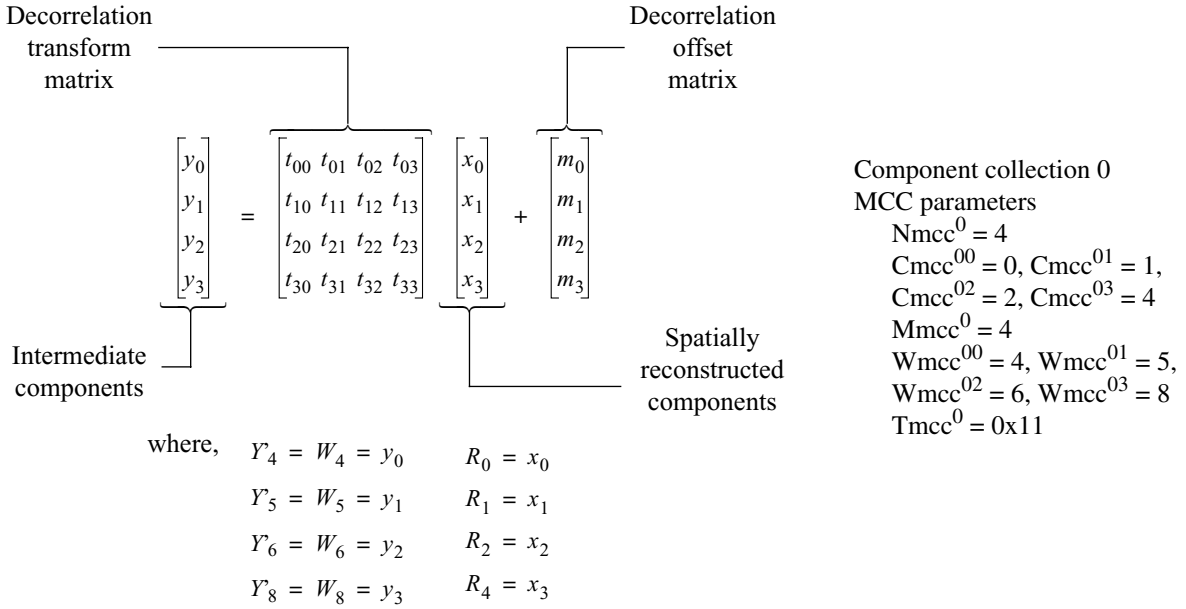

Figure I-4 Decorrelation transform matrix (MCC component collection 0 parameters)

The remaining components out of the original seven do not have any inverse decorrelation transform associated with them (ignoring any possible changes due to the four additional components). However, the MCC input component lists must include all seven original codestream components. Therefore, the second component collection is configured to pass the remaining components through without modification. Figure I-5 gives the MCC component collection 1 parameters needed to reorder the intermediate components.

Component collection 1 MCC parameters

$Nmcc^1 = 3$

$Cmcc^{10} = 3$, $Cmcc^{11} = 5$, $Cmcc^{12} = 6$

$Mmcc^1 = 3$

$Wmcc^{10} = 7$, $Wmcc^{11} = 9$,$Wmcc^{12} = 10$

$Tmcc^1 = 0x00$

Figure I-5 Reordering intermediate components (MCC component collection 1 parameters)

Figure I-6 illustrates the component reordering that we have implemented using the MCC marker segment in this example. The empty intermediate components have been created to hold the four additional components. Any subsequent inverse dependency transform processing that might reference one of the empty intermediate components will treat them as null components (i.e., identically zero). We will now consider the inverse dependency transformation for this example.

Again we will concentrate solely on the seven component multispectral image since we do not yet know where we will implement the four additional components.

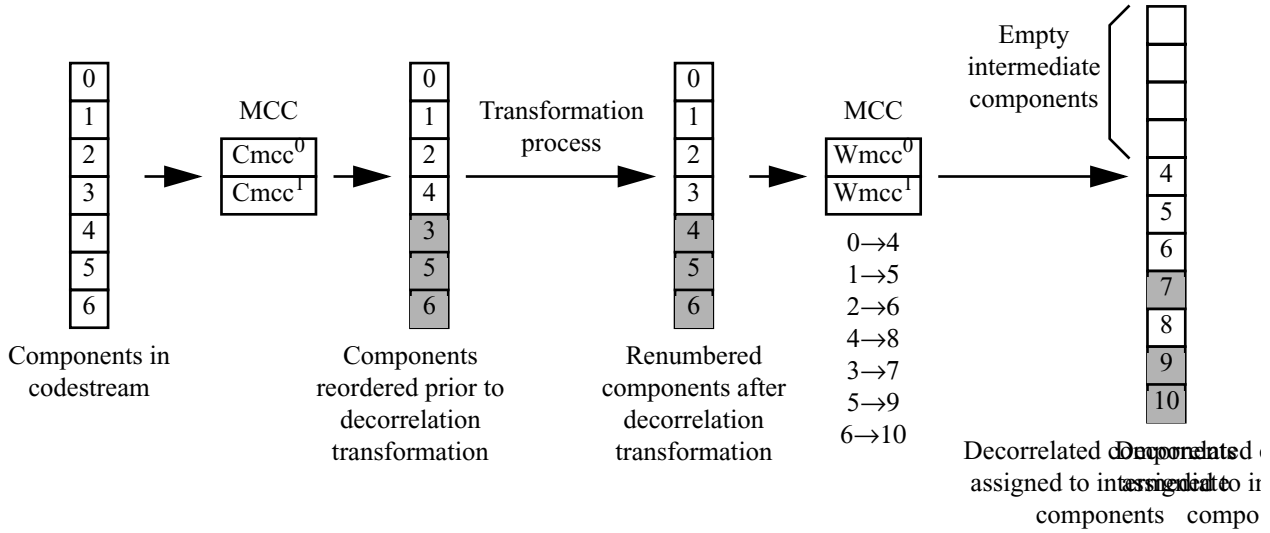

Figure I-6 Reordered intermediate components using MCC component collections

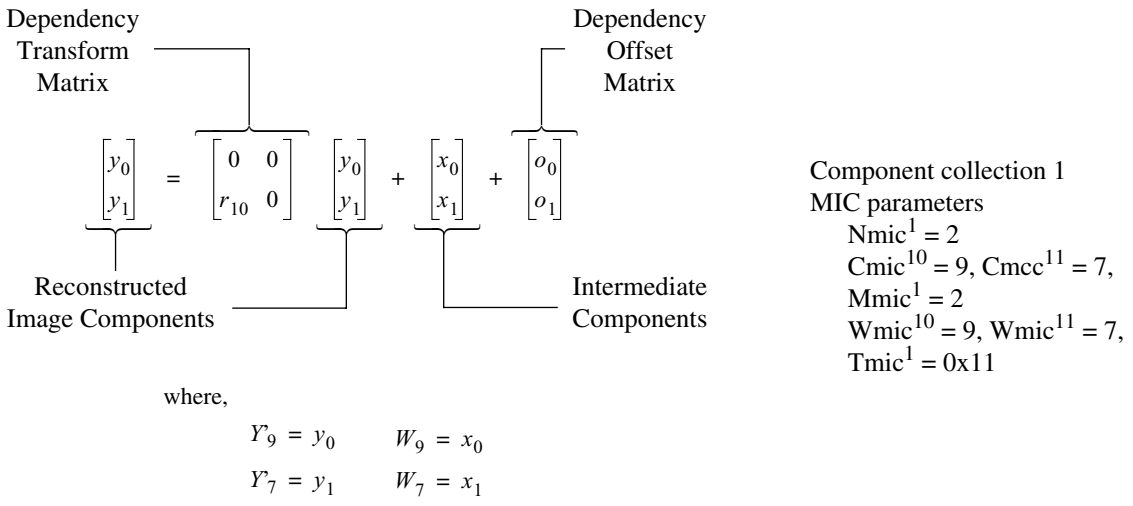

Figure I-7 Dependency transform matrix (MIC component collection 1 parameters)

Figure I-7 shows the inverse decorrelation transformation for reconstructed image components nine and seven. These components correspond to spatially reconstructed components five and three, respectively. We see that component seven is being predicted from component nine, analogous to the encoder predicting its original component three from component five. The inverse dependency transformation has been placed into component collection one in the MIC marker segment. Component collection zero for the MIC marker segment is given in Figure I-8. This component

collection does not have any inverse dependency transformation and all of the input intermediate components are simply passed through.

Component collection 0 MIC parameters
$Nmic^0 = 5$
$Cmcc^{00} = 4$, $Cmcc^{01} = 5$, $Cmcc^{02} = 6$,
$Cmcc^{03} = 8$, $Cmcc^{04} = 10$
$Mmcc^0 = 5$
$Wmcc^{00} = 4$, $Wmcc^{01} = 5$, $Wmcc^{02} = 6$,
$Wmcc^{03} = 8$, $Wmcc^{04} = 10$
$Tmcc^0 = 0x00$

Figure I-8 Passing through intermediate components (MIC component collection 0 parameters)

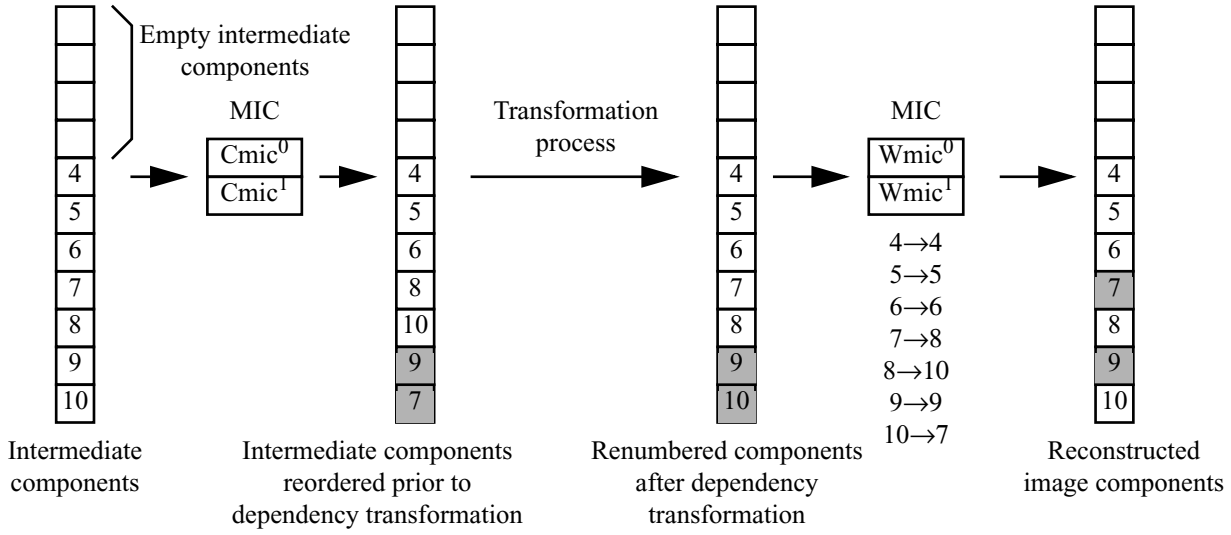

Figure I-9 Reordered reconstructed image components using MIC component collections

Figure I-9 illustrates the MIC component collections and reordering of the reconstructed image components. The component collections between the MCC and MIC marker segments are quite different. Although we might think of the seven component multispectral image as three distinct component collections (a decorrelation collection, a dependency collection, and a pass-through collection); from a codestream syntax point of view, it was handled with two sets of two component collections with different membership.

Up until this point we have largely neglected the fact that the encoder was going to include instructions for the decoder to create four additional reconstructed image components. We have used the MCC marker segment to make space for these components, but we have not modified the decorrelation transform matrix or the dependency transform matrix to tell the decoder how they are formed. Equation I.6 gives the form of the additional panchromatic component, Y_L^{1b}, and the three additional false colour components; Y_R^{3b}, Y_G^{3b}, and Y_B^{3b}. Careful examination of Equation I.6 reveals that one of the false colour components, Y_R^{3b}, is a function of Y'_7. This particular component is predicted from another reconstructed image component, Y'_7. We therefore know that Y_R^{3b} must be computed in the inverse dependency transformation. The other three additional components could be computed in the decorrelation transform matrix, but this would entail computing several intermediate components multiple times. We therefore choose to compute all additional components in the dependency transform matrix.

$$Y_L^{1b} = Y'_0 = \beta_0(Y'_4 - \mu_{Y'_4}) + \beta_1(Y'_5 - \mu_{Y'_5}) + \beta_2(Y'_6 - \mu_{Y'_6}) + \mu_{Y_L^{1b}}$$
$$= \beta_0 Y'_4 + \beta_1 Y'_5 + \beta_2 Y'_6 + \zeta$$

$$Y_B^{3b} = \alpha_0(Y'_4 - \mu_{Y'_4}) + \mu_{Y_B^{3b}} = \alpha_0 Y'_4 + \gamma_0 \qquad \text{I.6}$$
$$Y_G^{3b} = \alpha_1(Y'_5 - \mu_{Y'_5}) + \mu_{Y_G^{3b}} = \alpha_1 Y'_5 + \gamma_1$$
$$Y_R^{3b} = \alpha_2(Y'_7 - \mu_{Y'_7}) + \mu_{Y_R^{3b}} = \alpha_2 Y'_7 + \gamma_2$$

Combining Equation I.6 and the equation in Figure I-7, we may determine the new dependency transform matrix given in Equation I.7. We have retained the notation of Equation I.6 and Figure I-7 in forming Equation I.7 to make it clear where the various terms in the dependency transform and dependency offset matrices originated.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{10} & 0 & 0 & 0 & 0 & 0 \\ \beta_0 & \beta_1 & \beta_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_2 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} + \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ o_0 \\ o_1 \\ \zeta \\ \gamma_0 \\ \gamma_1 \\ \gamma_2 \end{bmatrix}$$

where, I.7

$$Y'_4 = y_0 \qquad W_4 = x_0$$
$$Y'_5 = y_1 \qquad W_5 = x_1$$
$$Y'_6 = y_2 \qquad W_6 = x_2$$
$$Y'_9 = y_3 \qquad W_9 = x_3$$
$$Y'_7 = y_4 \qquad W_7 = x_4$$
$$Y'_0 = Y_L^{1b} = y_5 \qquad W_0 = \varnothing = x_5$$
$$Y'_1 = Y_R^{3b} = y_6 \qquad W_1 = \varnothing = x_6$$
$$Y'_2 = Y_G^{3b} = y_7 \qquad W_2 = \varnothing = x_7$$
$$Y'_3 = Y_B^{3b} = y_8 \qquad W_3 = \varnothing = x_8$$

Figure I-10 gives the new MIC component collections for the eleven reconstructed image components. Collection zero in the MIC marker segment represents the new nine component dependency transform and component collection one is a two component pass through. Since we are not reordering any of the components in collection one, it could be omitted and the default behaviour of the inverse dependency transformation process would pass the components through untouched. It has been included for completeness. No net component reordering has been performed between the input and output component collections for this MIC marker segment. Component collection 0 in the MIC marker segment

points to two MCT marker segments with index one which contain the dependency transform matrix and the dependency offset matrix. The MCT marker segments must be present in either the main header or appropriate first tile-part header.

Component collection 0 MIC parameters
$Nmic^0 = 9$
$Cmcc^{00} = 4$, $Cmcc^{01} = 5$, $Cmcc^{02} = 6$,
$Cmcc^{03} = 9$, $Cmcc^{04} = 7$, $Cmcc^{05} = 0$,
$Cmcc^{06} = 1$, $Cmcc^{07} = 2$, $Cmcc^{08} = 3$
$Mmcc^0 = 9$
$Wmcc^{00} = 4$, $Wmcc^{01} = 5$, $Wmcc^{02} = 6$,
$Wmcc^{03} = 9$, $Wmcc^{04} = 7$, $Wmcc^{05} = 0$,
$Wmcc^{06} = 1$, $Wmcc^{07} = 2$, $Wmcc^{08} = 3$
$Tmcc^0 = 0x11$

Component collection 1 MIC parameters
$Nmic^1 = 2$
$Cmcc^{10} = 8$, $Cmcc^{11} = 10$
$Mmcc^1 = 2$
$Wmcc^{10} = 8$, $Wmcc^{11} = 10$
$Tmcc^1 = 0x00$

Figure I-10 Modified MIC component collections (11 components)

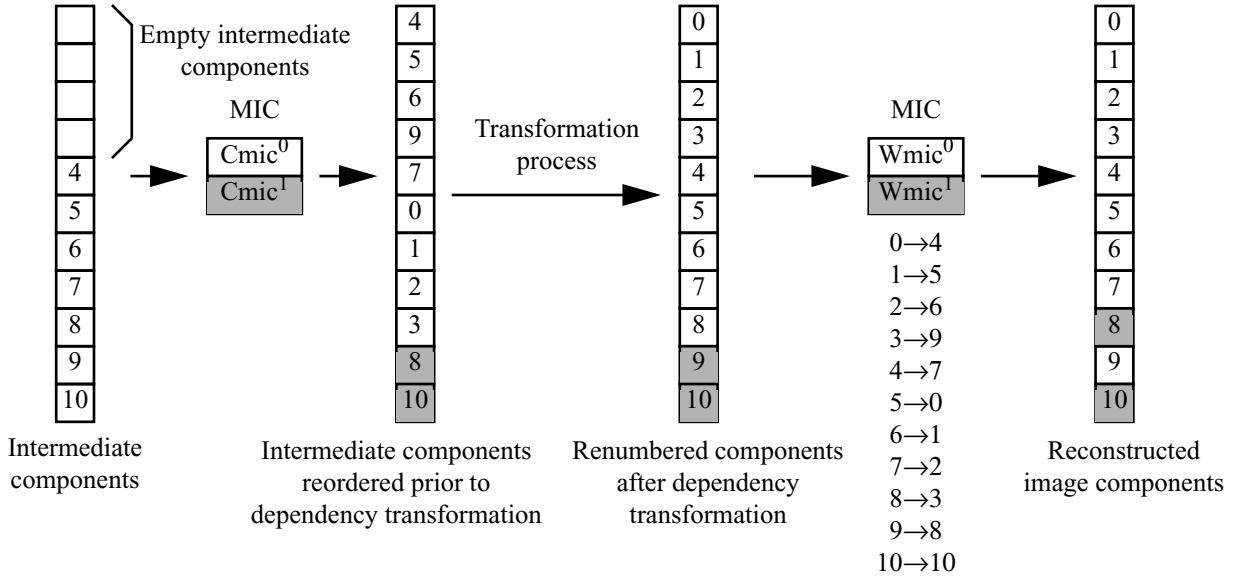

Figure I-11 Reordered reconstructed image components using MIC component collections

Figure I-4, Figure I-5, Figure I-6, Figure I-10, Figure I-11, and Equation I.7 represent the complete set of inverse decorrelation transformation matrices, inverse dependency transformation matrices, MCC marker segment parameters, and MIC marker segment parameters for the eleven reconstructed image components in this example. To properly apply an inverse dependency transformation, care must be taken with the processing order of components. Although we have written Equation I.7 in matrix form, it is not a typical matrix equation in the sense that we can not simply plug in the input column vectors and matrix multiply to get our results. The dependency transformation matrix must contain non-zero entries only below the matrix diagonal. Any causal dependency transform can be arranged into this form. By ordering the dependency matrix in this fashion, the dependency transform may be processed simply by reconstructing the image components row by row, starting with the first row in the matrix. The lower triangular nature of the dependency matrix

guarantees causality in the dependency transformation processing. It is true that there may be other processing orders possible for a given dependency transform matrix, depending upon its structure. In general, any processing order that ensures the causality of the reconstructed image component processing is acceptable. The method described here will work for any properly formed dependency transform matrix.

The following observations can be made regarding component collections:

The Wmcc ^{i} and Wmic^{i} parameters jointly determine the final ordering of the reconstructed image components. In the absence of an MIC marker segment, then Wmcc^{i} will control the final ordering

Wmcc ^{i} may be used to create space for new components that do not exist in the codestream. The new components themselves may be generated by the decorrelation and/or dependency transformation processes. However, since the dependency transform matrix is constrained to be square, the Wmic^{i} cannot be used to create space for new components.

Input components can be reused in different component collections. Output components should be kept distinct across all component collections in any given MCC or MIC marker segment.

If an MCC or MIC marker segment references an input or output component that does not exist, that component should be treated as a null component (i.e. full of zeros).

Processing order in the inverse dependency transformation is important. The structure of the dependency transform matrix must be examined to determine the causal processing order for reconstructed image components.

One can do more than decorrelation and simple prediction with the MCC and MIC marker syntax.

I.2 Multi-dimensional wavelet transform

In addition to providing matrix-based decorrelation transform processing, this Annex provides a wavelet-based decorrelation process. The MCC marker segment allows for the specification of a wavelet transform to be applied to a given input component collection (see Annex A.3.8). In fact, it is possible to use wavelet-based decorrelation on one component collection and matrix-based decorrelation on another within the same MCC marker segment. When wavelet-based decorrelation is used, the ATK marker segment (see Annex A.3.5) and ADS marker segment (see Annex A.3.4) are used to specify the wavelet transform kernel and arbitrary decomposition used in processing the component collection. Signalling in the MCC marker segment specifies which ATK and which ADS marker segments are to be used for the component collection.

I.2.1 Inverse multi-dimensional wavelet transform

The spatially reconstructed image components are processed, spatial location by spatial location, with an inverse multiple component wavelet transform when so indicated in the MCC marker segment. The inverse multiple component wavelet transform is a one-dimensional wavelet transform. However, it can be described using the two-dimensional wavelet transform mechanisms described in Annex F of ITU-T T.800 | IS 15444-1 with modifications and extensions as noted in Annex G of this standard.

The spatially reconstructed image components are denoted as $R_j(x,y)$, where j = 0, 1,..., Csiz-1. At each spatial location (x,y), a two-dimensional array $a_{N_L}(u,v)$ is formed. The entries in the i^{th} input component list, Cmcc^{ik}, k = 0, 1,..., Nmcc^{i}-1, from a MCC marker segment are used to order the samples in $a_{N_L}(u,v)$. Specifically, for k = 0, 1,..., Nmcc^{i}-1, $a_{N_L}(k,0) = R_{\text{Cmcc}^{ik}}(x,y)$. The array $a_{N_L}(u,v)$ is only one sample wide in the v direction, thus it is effectively a one-dimensional vector. However, it can be treated as two-dimensional for purposes of the inverse multiple component wavelet transform. The inverse transform procedure IDWT (from Annex F of ITU-T T.800 | IS 15444-1) is now applied as modified by extensions noted in Annex G of this standard. The subband coordinates tby_0 and tby_1 are 0 and 1, respectively, for all levels in the reconstruction process. The subband coordinates tbx_0 and tbx_1 are determined as in

Annex B of ITU-T T.800 | IS 15444-1, modified by extensions noted in this standard. For purposes of determining those coordinates, the tile size in the *x* direction is assumed to be Nmcc^{i}.

Using the notation of ITU-T T.800 | IS 15444-1, the output of the IDWT procedure is $I(x,y)$. To avoid confusion regarding the component spatial locations, denote the output of the IDWT procedure as $I(u,v)$. The intermediate components in the inverse multiple component transform are denoted as $W_j(x,y)$. The entries in the relevant output component list from the MCC marker segment, $\text{Wmcc}^{\text{i}k}$, k = 0, 1,..., Mmcc^{i}-1, are used to order the samples in $W_j(x,y)$. Specifically, $W_{\text{Wmcc}^{ik}}(x,y) = I(\text{Wmcc}^{ik},0)$ for k = 0, 1,..., Mmcc^{i}-1.

I.2.2 Forward multi-dimensional wavelet transform (informative)

The input image components are processed, spatial location by spatial location, with a multiple component wavelet transform. The presence of this transform is indicated in the MCC marker segment. The forward multiple component wavelet transform is a one-dimensional wavelet transform. However, it can be described using the two-dimensional wavelet transform mechanisms of Annex F of ITU-T T.800 | IS 15444-1 with modifications and extensions as noted in Annex G of this document.

The input image components are denoted as $I_j(x,y)$ where j is the image component number. At each spatial location (x,y), a two-dimensional array $T(u,v)$ is formed. Denote the set of image components to be transformed by $S_{in} = \{icc^0, icc^1, \ldots, icc^{N_{cc}-1}\}$. Then the array $T(u,v)$ is constructed as $T(k,0) = I_{\text{icc}^{\text{k}}}(x,y)$, k = 0, 1,..., Ncc-1 The array $T(u,v)$ is only one sample wide in the v direction. Thus it is effectively a one-dimensional vector. However, it can be treated as two-dimensional for purposes of the multiple component wavelet transform. The transform procedure FDWT (from Annex F of ITU-T T.800 | IS 15444-1) is now applied as modified by extensions noted in Annex G of this document. The input $I(x,y)$ for the FDWT procedure in Annex F of ITU-T T.800 | IS 15444-1 is set to $T(u,v)$. The subband coordinates tby_0 and tby_1 are 0 and 1, respectively, for all levels in the transform process. The subband coordinates tbx_0 and tbx_1 are determined as in Annex B of ITU-T T.800 | IS 15444-1, modified by extensions noted in this standard. For purposes of determining those coordinates, the tile size in the *x* direction is *Ncc*.

Using the notation of ITU-T T.800 | IS 15444-1, the output of the FDWT procedure is $a_{N_L}(u,v)$, where N_L is the number of decomposition levels applied. The components passed to the transform and coding engines (after application of multiple component transforms) are denoted as $R_j(x,y)$ where j = 0, 1,..., Csiz-1. Denote the set of components to be filled with the result of the multiple component wavelet transform by $S_{out} = \{occ^0, occ^1, \ldots, occ^{N_{cc}-1}\}$. Then the $R_j(x,y)$ are set as $R_{\text{occ}^{\text{i}}}(x,y) = a_{N_L}(i,0)$, i = 0, 1,..., Ncc-1. the parameters N_L, S_{in}, and S_{out} are all communicated to the decoder by setting appropriate field values in the MCC marker segment.

Annex J

Non-linear Transformation

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex A.2.1).

This Annex specifies two 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, gamma and look-up table (LUT) style non-linearities, may be employed by encoders prior to multiple component transformation and encoding to increase compression efficiency. A common usage of these transformations might be to perceptually flatten a scanner or sensor with a linear response, from 12 bits to 8 bits precision prior to compression.

J.1 Non-linear transforms

The use of the non-linear transform is signalled in the Rsiz parameter (see Annex A.2.1).

J.1.1 Decoded component reconstruction

The non-linear transforms specified in this Annex are point transforms. These transforms are applied to each sample (point) in a given component. The transforms do not span components like the multiple component transforms described in Annex I. They may be used, however, in conjunction with the multiple component transforms. Figure J-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.

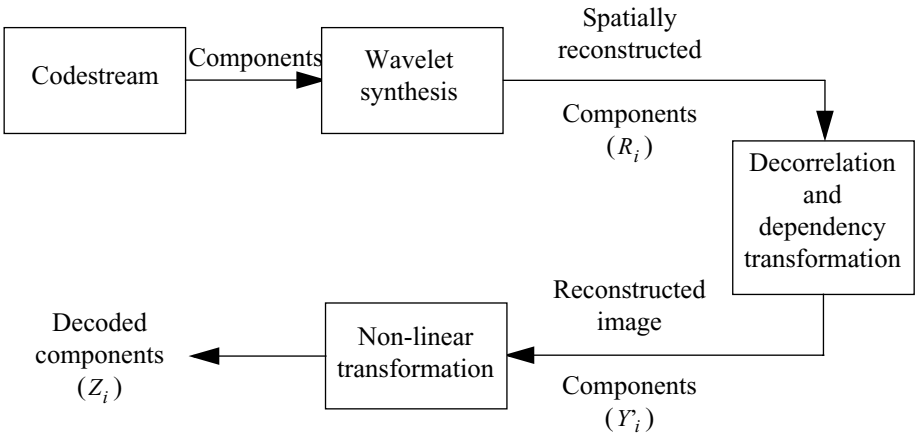

Figure J-1 Non-linear transformation application

J.1.2 Bit depth and marker interpretation

If a multiple component transformation is used in a codestream, the SIZ (see ITU-T T.800 | IS 15444-1 Annex A.5.1 and Annex A.2.1, this document) 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. Processing of the codestream components with the inverse decorrelation and dependency transforms 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 Annex 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 transforms, the CBD marker segment placed in the codestream shall determine the number of reconstructed image components and their bit depths prior to 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. An NLT marker segment (see Annex A.3.10) is placed in the codestream whenever a non-linear transform is used. The NLT marker indicates the bit depths of the decoded components that result from the application of a non-linear transform to a reconstructed image component. Those reconstructed image components that do no undergo a non-linear transform shall have no change in their bit depth.

J.2 Gamma-style non-linearity

The gamma-style non-linearity contains two functional segments: a linear region for small component values, and a power-law exponential region for large component values. Figure J-2 shows a forward gamma non-linearity transformation that might be used by an encoder. This particular gamma-style transformation is from Rec. 709 (HDTV). The relationship between an input value L and gamma-adjusted value L' is given by,

$$\begin{aligned} L' &= -\left(A|L|^{E} - B\right) && L < -\frac{T}{S} \\ L' &= SL && -\frac{T}{S} \le L \le \frac{T}{S} \\ L' &= AL^{E} - B && L > \frac{T}{S} \end{aligned} \qquad \text{J.1}$$

where S is the toe-slope, T is the toe-slope threshold, E is the gamma exponent, and A and B are continuity parameters.

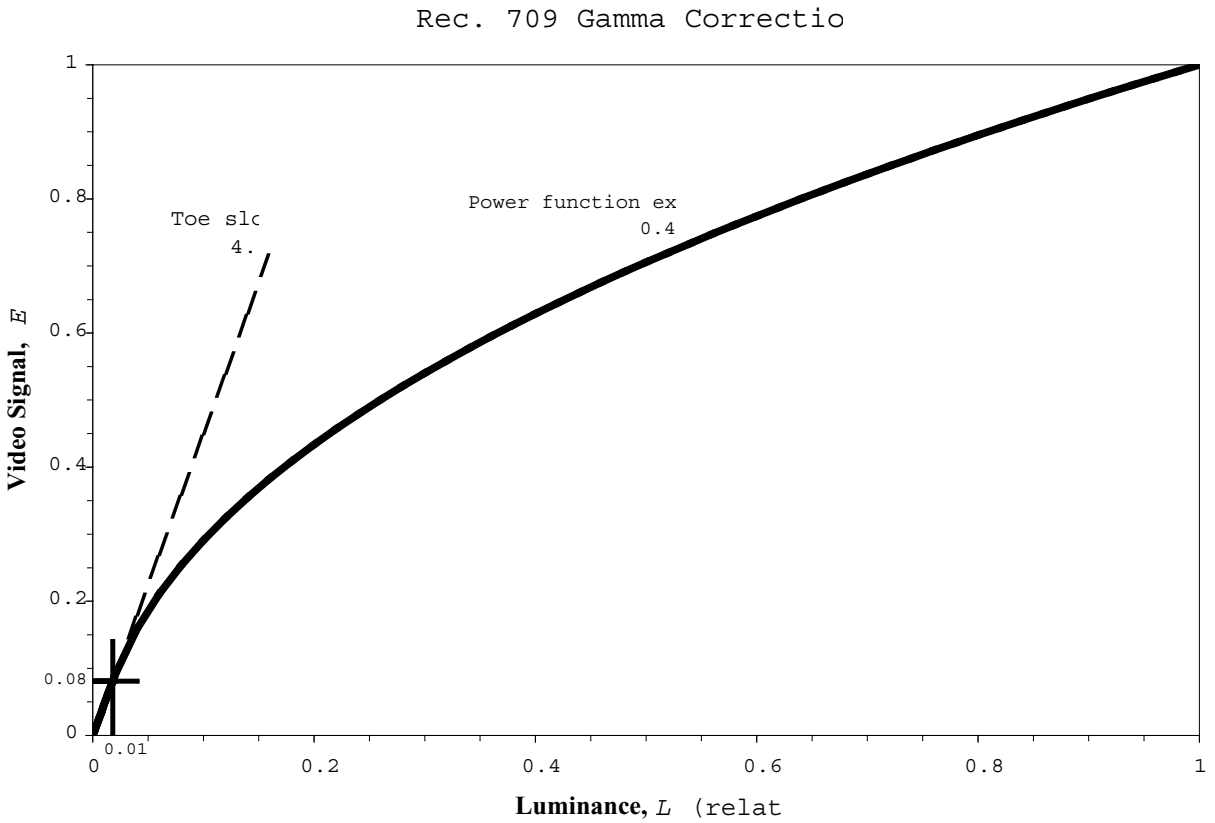

Figure J-2 Gamma-type forward non-linear transformation

A decoder inverts the gamma-type transformation by application of the following equation,

$$\begin{aligned} L &= -\left(\frac{|L'| + B}{A}\right)^{1/E} && L' < -T \\ L &= \frac{L'}{S} && -T \le L' \le T \\ L &= \left(\frac{L' + B}{A}\right)^{1/E} && L' > T \end{aligned} \qquad \text{J.2}$$

The parameters T, S, E, A, and B are all communicated in the NLT marker segment. These parameters must be appropriately scaled for the dynamic range of the data. For example, if we wished to apply the function in Figure J-2 to 8 bit data the T, A, and B parameters would have to be adjusted appropriately.

J.3 LUT-style non-linearity

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 LUT to perform the transform. Second, if the number of table values specified is equal to the number of possible decoded values, then a LUT is explicitly specified.

The LUT-style non-linearity requires that a list of table values be supplied. It is assumed that the possible number of decoded values for a component is $N = 2^{R_I}$, where R_I is the number of bits used to represent a given reconstructed image

component. It is also assumed that the minimum and maximum possible decoded values are D_{min} and D_{max}, respectively. A total of N_{points} table values are supplied, where $2 \le N_{points} \le N$. Table value T_i, $i = 0, 1, ..., N_{points} - 1$, is assigned to be the non-linearity output value when D_i is supplied as input. The D_i are implicitly determined by D_{min}, D_{max}, and N_{points} through the following equation,

$$D_i = D_{min} + i\Delta$$
$$\text{where,} \quad \Delta = \frac{D_{max} - D_{min}}{N_{points} - 1} \qquad \text{J.3}$$

For any value, x, that is supplied as input to the non-linearity and is not equal to any of the D_i, the corresponding output is determined by interpolation as follows. Let k be the largest integer such that $D_k < x$. Then the non-linearity output value, x', for input, x, is given by,

$$x' = T_k + \left(\frac{x - D_k}{\Delta}\right)\left(T_{k+1} - T_k\right). \qquad \text{J.4}$$

The parameters, N_{points}, D_{min}, and D_{max} are signalled in the NLT marker segment, along with the T_i values.

J.4 LUT-style non-linearity example (informative)

Table J-1 gives a LUT approximation to the Rec. 709 gamma curve. For this example we have $N_{points} = 9$, $D_{min} = 0$, and $D_{max} = 255$. The various T_i table values are given. Figure J-3 plots the LUT approximation and a scaled version of the Ref. 709 curve suitable for 8-bit data.

Table J-1 8-bit LUT approximation to Rec. 709

i	D_i	T_i
0	0.00	0.00
1	31.88	90.00
2	63.75	127.00
3	95.63	156.00
4	127.50	180.00
5	159.38	202.00
6	191.25	222.00
7	223.13	239.00
8	255.00	255.00

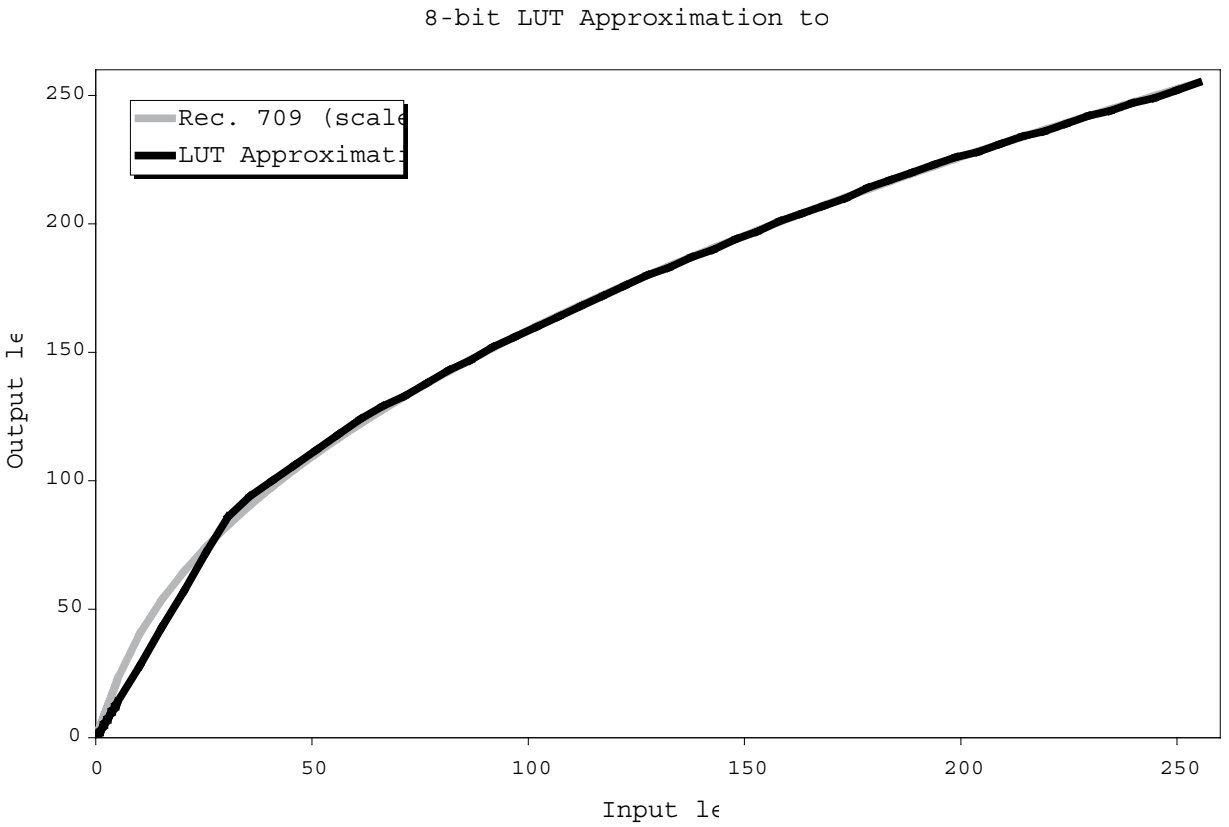

Figure J-3 LUT-type non-linear transformation

Annex K

Region-of-interest coding and extraction, extensions

(This Annex forms a normative and 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 ITU-T T.800 | IS 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 Annex 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.

K.1 Decoding of ROI

The procedure specified in this section is applied only in the case of the presence of an extended RGN marker segment, see Annex A.2.4 (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 ARN marker segment for each ROI. The following steps 2-6 are applied to each coefficient (u, v) of subband *b.*

2) Generate the ROI mask $\{M_i(u, v)\}$ for all ROI, see Annex K.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

$$s(u, v) = s_{Max} - max(s_i \cdot M_i(u, v)) \qquad \text{K.1}$$

where i = *0... Number of ROI-1*

5) For each coefficient (u, v) discard the first *s(u,v)* MSBs and shift the remaining MSBs *s(u,v)* places, as described in Equation K.2, for $i = 1, \ldots, M_b$

$$MSB_i(b, u, v) = \begin{cases} MSB_{i+s(u,v)}(b, u, v) & \text{if } i + s(u, v) \le N_b(u, v) \\ 0 & \text{if } i + s(u, v) > N_b(u, v) \end{cases} . \qquad \text{K.2}$$

6) Update the value of $N_b(u, v)$ as given in Equation K.3.

$$N_b(u, v) = \max(0, N_b(u, v) - s(u, v)) \qquad \text{K.3}$$

K.2 Description of the Scaling based method

This section 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.

K.2.1 Encoding with ROI (informative)

At the encoder side an ROI mask is created describing which quantized transform coefficients must be encoded with better quality (up to lossless). The ROI mask is a bit map describing these coefficients for one ROI. See Annex K.3 for details on how the mask is generated.

The quantized transform coefficients are scaled in such a manner that the relative significance of each transform coefficient is equal to the specified scaling value, s, of the ROI to which it applies. If a transform coefficient belongs to several ROI the largest *s* value is chosen. If a transform coefficient belongs to the Background, the scaling value *s* equals *0*. Before scaling the quantized transform coefficients of one code-block, the highest, s_{Max}, and lowest, s_{Min}, scaling value for the coding block are found.

Consider a quantized transform 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 then increase by $(s_{Max}-s_{Min})$.

Since the coding blocks are treated independently, quantized transform coefficients belonging to the same ROI might end up having different levels of significance in different coding blocks. This difference between coding blocks must be 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 would be 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 transform 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 Annex K.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 $s(u,v) = s_{Max}-s_{Min}$ LSBs to each coefficient $|q_b(u,v)|$. The number M'_b of magnitude bit-planes will then be

$$M'_b = M_b + s(u,v) \qquad \text{K.4}$$

where M_b is given by ITU-T T.800 | ISO15444-1 Equation E.2 and the new value of each coefficient is given by

$$|q_b(u,v)| = |q_b(u,v)| \cdot 2^{s(u,v)} \qquad \text{K.5}$$

4) Scale down all coefficients for which $s(u,v) \neq 0$ so that

$$|q_b(u, v)| = \frac{|q_b(u, v)|}{2^{s(u, v)}} \quad \text{K.6}$$

5) For each ROI write the scaling value, *s*, shape, and reference points into the codestream using the extended RGN marker segment as described in Annex A.2.1.

K.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 transform 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

$$M(u, v) = \begin{cases} 1 & \text{wavelet coefficient (u,v) is needed} \\ 0 & \text{accuracy on (u,v) can be sacrificed without affecting ROI} \end{cases} \quad \text{K.7}$$

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 transform will reproduce the coefficients of the previous mask.

For example, the last step of the inverse transform 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 transform 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.

K.3.1 Elliptic mask generation on the reference grid

The elliptic mask described in this section is generated on the reference grid. When generated on the reference grid the methods described in Annex K.3.2 and Annex K.3.3 are used for mask generation in the wavelet domain. An ellipse is described by four parameters, see Figure K-1, all signaled in the extended RGN marker, see Annex A.2.4. The

parameters are $(x_{offset}, y_{offset}, h, w)$ where x_{offset} and y_{offset} are the x offset and y offset of the center of the ellipse from the reference grid origin, respectively; h is the height of the ellipse; w is the width of the ellipse.

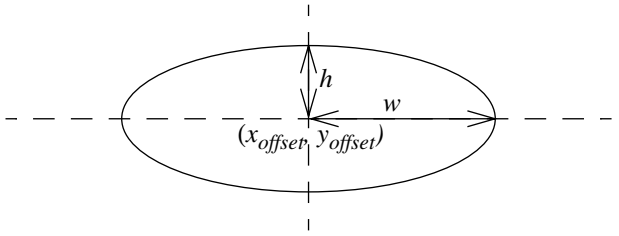

Figure K-1 — Elliptic mask on the reference grid

The correct mask for the reference grid is given by Equation K.8.

$$h^2 \cdot (x - x_{offset})^2 + w^2 \cdot (y - y_{offset})^2 \leq w^2 \cdot h^2 \qquad \text{K.8}$$

Thus, a coordinate on the reference grid belongs to the ROI if and only if Equation K.8 is true.

K.3.2 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. This ROI mask is defined on the reference grid, and is a two-dimensional binary mask with the same dimensions as the corresponding tile-component. The ROI mask has non-zero values where the samples of the tile-component belong to the ROI and zero values otherwise.

Instead of calculating the wavelet coefficients using the lifting steps, as described in Annex G, 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.

This is done by replacing equations Equation G.8 and Equation G.10 by

For each lifting step s where s goes from number of lifting steps, N_{LS} to 1,

a)

$$\begin{aligned} &\text{for all n :} \\ &\text{if } (R_{ext}(2n + m_s) == 1) : \\ &R'_{ext}(2n + 1 - m_s - 2off_s) = 1, \ldots, \\ &R'_{ext}(2n + 1 - m_s + 2(off_s + L_s - 1)) = 1 \end{aligned} \qquad \text{K.9}$$

where R_{ext} are the samples in the ROI mask, $M_i(u,v)$ corresponding to V_{ext} in Equation G.8 and Equation G.10. 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, and where off_s is the offset for lifting step s.

b)

$$R_{ext} = R'_{ext} \quad \text{K.10}$$

After doing this for all the lifting steps, the ROI mask samples are separated into subbands the same way as the wavelet coefficients are separated in using the deinterleave procedure described in Annex F.4.5 ITU-T Rec. T.800 | ISO/IEC 15444-1.

This ensures that each coefficient that has affected a coefficient in the ROI during the inverse wavelet transform will have a '1' in the corresponding place in the ROI mask.

K.3.3 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), and the bottom right corner, (x_2, y_2), on the reference grid will be given in the ARN marker segment as XAarn, YAarn, XBarn, and YBarn, respectively (see Annex A.3.8).

In each level of decomposition, the steps described in the previous section 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 be the lowest and highest indices of non-zero samples in R_{ext}.

1) For each lifting step s where s goes from number of lifting steps N_{LS} to 1,

a) Find the lowest sample index $2n + m_s \geq x_1)$ that is in the mask

$$x'_1 = 2n + m - (2off_s + 1) \quad \text{K.11}$$

$$\begin{aligned} &\text{if } (x'_1 > x_1): \\ &\quad x'_1 = x_1 \end{aligned} \quad \text{K.12}$$

b) Find the highest sample index $2n + m_s \leq x_2$

$$x'_2 = 2n + m_s + (2(off_s + L_s) + 1) \quad \text{K.13}$$

$$\begin{aligned} &\text{if } (x'_2 < x_2): \\ &\quad x'_2 = x_2 \end{aligned} \quad \text{K.14}$$

c) Set $\begin{matrix} x_1 = x'_1 \\ x_2 = x'_2 \end{matrix}$,

where $m_s = 1 - m_{s-1}$ indicates whether the *s*th 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*.

Let all samples between x1 and x2 be non-zero and then separate the ROI mask samples into subbands the same way as the wavelet coefficients are separated in using the deinterleave procedure described in Annex F.4.5 ITU-T Rec. T.800 | ISO/IEC 15444-1.

K.4 Remarks on region of interest coding

This section contains a remark for the case of multi-components and a remark on implementation precision.

K.4.1 Usage together with ITU-T T.800 | ISO/IEC 15444-1 RGN marker segment

The ARN marker segment should not be used together with the RGN marker segment described in ITU-T T.800 | ISO/IEC 15444-1.

K.4.2 Multi-component remark (informative)

For the case of color images, the method applies separately in each color component. If some of the color 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.

K.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 down scaling 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 L

JPX extended file format syntax

(This Annex forms a normative and 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 ITU-T T.800 | IS 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard.

L.1 File format scope

This annex of this Recommendation | International Standard 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 ITU-T T.800 | IS 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 of this Recommendation | International Standard.

This annex of this Recommendation | International Standard

- 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.

L.2 Introduction to JPX

As defined in ITU-T T.800 | IS 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, a 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 must 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.

L.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 .jpx (readers shall also recognize files with the extension .JPX). 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 for that file. This will maximize the interoperability of that file without sacrificing file indication (as the Brand field in the File Type box for all JPX files shall always be jpx\040 .

L.2.2 File organization

As in JP2 files, a JPX file represents a collection of boxes. Some of those boxes are independent, and some of those boxes contain other 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, but other Recommendations | International Standards may define other boxes, which may be found within a JPX file. 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 (ITU-T T.800 | IS 15444-1 Annex I.6).

L.2.3 Greyscale/Colour/multi-component specification

The JP2 file format allowed the colourspace of the image to be specified in multiple ways (Enumerated or Restricted ICC methods). The JPX file format expands on this by:

Defining additional colourspaces for the Enumerated method

Defining a method for vendors and other standards bodies to register additional Enumerated colourspaces.

Defining a new method to allow the use of any input ICC profile (the Any ICC method)

Defining a new method to allow vendors to define unique codes for colourspaces (the Vendor Colour method)

Allowing the use of extensions of the multiple component transform and non-linearity point transformation extensions within the codestream (see Annex I and J of this Recommendation | International Standard)

L.2.4 Specification of opacity information

The JPX file format specifies two extensions with respect to 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.

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.

L.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, for example those defined by ISO/IEC 15938 (MPEG—7), may be embedded in a JPX file within XML boxes.

Furthermore, the MPEG—7 binary box (as defined in Annex L.9.19) may be used to store MPEG—7 binary (BiM) format metadata as defined in Section L.11.18.

L.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.

L.2.7 Combining multiple codestreams

In addition to specifying a 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.

L.2.8 Conformance

L.2.8.1 Interpretation of JPX data structures

All conforming files shall contain all boxes required by this Recommendation | International Standard as shown in Table L-12, and those boxes shall be as defined in this Recommendation | International Standard. Also, all conforming readers shall correctly interpret all required boxes defined in this Recommendation | International Standard.

L.2.8.2 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, the following baseline set of features is defined. Files that are extended 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.

L.2.8.2.1 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.

L.2.8.2.2 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 (signaled 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.

L.2.8.2.3 Codestreams

The codestream specified by the first compositing layer shall not require support for extensions other than the multiple component transform (Annex I.1) and non-linearity extensions (Annex J). 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 be require support for other extensions in order to be decoded.

In addition, the JPX file format provides for the storage of codestream that are compressed by other compression types. This is indicated through the C field in the Image Header box in either the JP2 Header box or a Codestream Header box.

L.2.8.2.4 Color specification

The first compositing layer shall contain at least one Colour Specification box from the following list:

- Enumerated method and EnumCS values indicating either sRGB, sRGB—grey, ROMM—RGB, sRGB related luminance-chrominance, or e-sRGB.
- Enumerated method, EnumCS value of either CIELab or CIEJab, and no EP values (default values are used)
- Restricted ICC method
- Any ICC method

A baseline JPX file may contain additional colorspace specifications, such as other enumerated values or vendor defined colorspace specifications. However, the file shall contain at least one color 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 colorspace of the image).

L.2.8.2.5 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.

L.2.8.2.6 Cross-reference boxes

All Cross-Reference boxes that must 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.

L.2.8.2.7 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.

L.2.8.2.8 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.

L.2.8.2.9 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 behaviors and functions associated with that extended data.

L.2.9 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 L-1 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 L-1, 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.

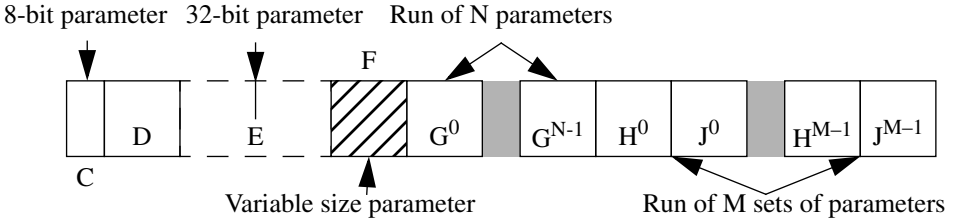

Figure L-1 — Example of the box description figures

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.

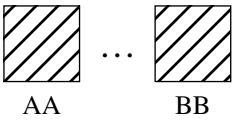

Figure L-2 — Example of the superbox description figures

For example, the superbox shown in Figure L-2 must contain an AA box and a BB box, and the BB box must follow the AA box. However, there may be other boxes found between boxes AA and BB. Dealing with unknown boxes is discussed in Annex L.10

L.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 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.

L.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, one 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 and optimization. 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 though it can use the more complex profile.

L.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 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 must 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 Color specification box, then all additional parameters must be specified for that Colour Specification box.

L.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.

L.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 UUID s. 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 the file 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).

L.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.

L.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.

L.3.7 Interactions with the decorrelating multiple component transform

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, MIC, and NLT markers specified in Annex A.3.7, Annex A.3.8, Annex A.3.9, and Annex 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.

L.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 filename/URL data (the Data Reference box) in a table. Each fragment pointer 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 must 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 Annex L.9.8 in this Recommendation | International Standard. 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 Annex L.9.9).

The following figure 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:

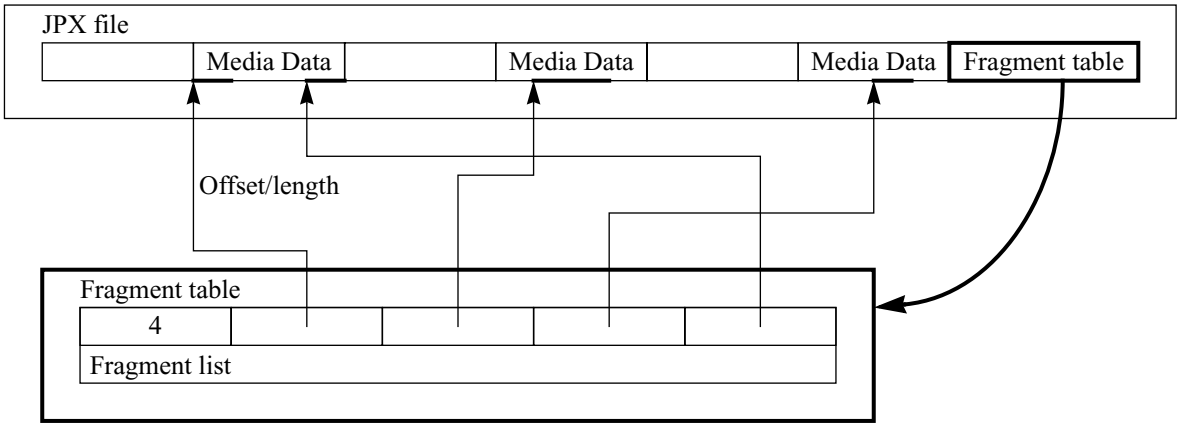

Figure L-3 Example fragmented JPX file where all fragments are in the same file

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 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.

To extract the complete codestream from the file, an application must locate the fragment table for that codestream in the file, and then parse 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.

The following figure 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

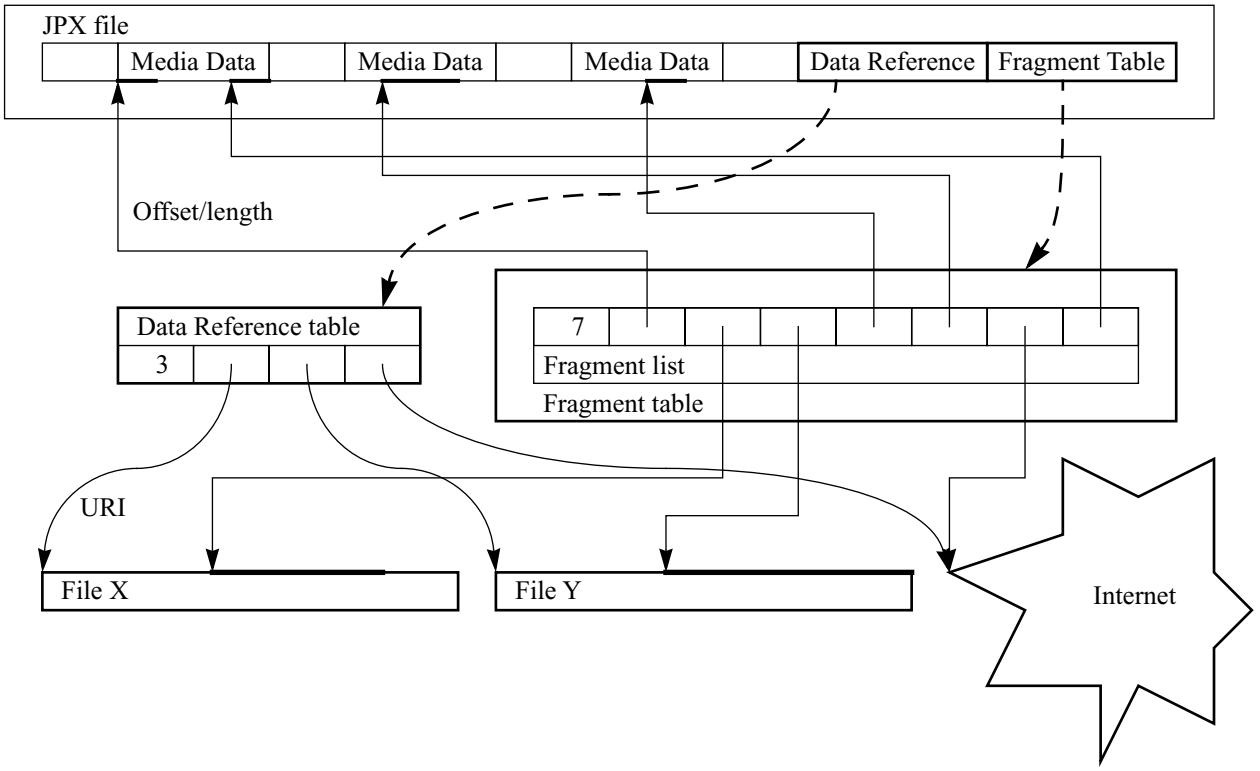

Figure L-4 Example fragmented JPX file where all fragments are in the same file

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.

L.5 Combining multiple codestream

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.

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. The JPX file format includes syntax for specifying how codestreams are combined to form compositing layers including how the codestreams should be 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.

L.5.1 Mapping codestreams to compositing layers

To facilitate the mapping of multiple codestreams to single compositing layers, JPX separates header fields defined for JP2 into two logical groups: those specific to a single codestream are grouped into a Codestream Header box (Annex L.9.3) and those specific to a compositing layer are grouped into a Compositing Layer Header box (Section Annex L.9.4). The process of mapping codestream components to channels is exemplified in Figure L-5. Multiple codestreams are combined via a Codestream Registration box (Annex L.9.4.7) to provide the complete set of components for a compositing layer. A Component Mapping box (ITU-T T.800 | IS 15444-1 Annex 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 (Annex L.9.4.5) or an Opacity box (Annex L.9.4.6). 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.

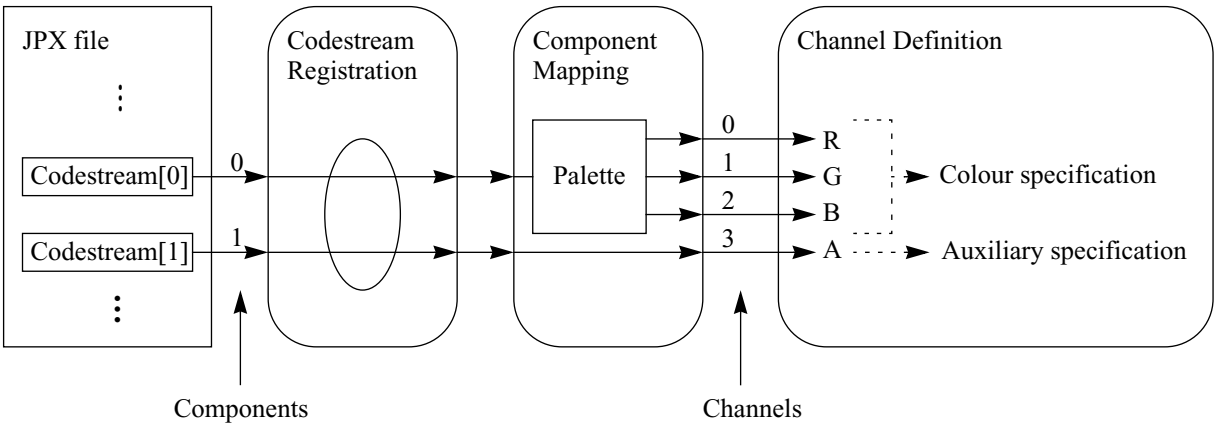

Figure L-5 Example combination of two codestreams into a single compositing layer

L.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 any Component Registration boxes are present then there shall be one Component Registration box in each Compositing Layer Header box in the file. In this case, the order of compositing layers is given by the sequence order of compositing layer header boxes in the file.

L.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.

L.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 three mechanisms to share information: default headers, cross-references and label associations.

L.5.2.1 Default headers and metadata

Where 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 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 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.

L.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.

L.5.2.3 Labelling and association

Need text

L.5.3 Composition

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

L.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 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:

$$\begin{aligned} A_c &= 1-(1-A_t)\times(1-A_b) \\ s &= \frac{A_t}{A_c} \\ t &= \frac{(1-A_t)\times A_b}{A_c} \\ R_c &= sR_t + tR_b \\ G_c &= sG_t + tG_b \\ B_c &= sB_t + tB_b \end{aligned} \qquad \text{L.1}$$

In the case where the bottom sample is fully opaque, this simplifies to:

$$\begin{aligned} R_c &= A_tR_t + (1-A_t)R_b \\ G_c &= A_tG_t + (1-A_t)G_b \\ B_c &= A_tB_t + (1-A_t)B_b \end{aligned} \qquad \text{L.2}$$

The above equations require access to the background pixel however, 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 standard.

L.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 must save the background prior to execution and use this saved background when executing the next instruction.

L.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 PERSIST 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. This mode permits multi-sprite animation.

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.

L.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 must 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 compositing layer defined in the file in sequence order shall be used. A single compositing layer may be reused any number of times in any given animation.

L.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 upon the caching strategy used by the reader as many will not wish to free any compositing layer once decoded.

L.6 Using Reader Requirements Masks to Determine How a File Can be Used

JPX defines a file architecture rather than a file type. The JPX 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. Parameterised CIE Lab).

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 with in the head 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 something - 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 reader).

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 an 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 - information it can determine from 1.

All JPEG 2000 family files are not necessarily interoperable. For this reason, these expressions describe each aspect of the file, and the combination of features that must be supported to interpret the file correctly.

L.6.1 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.

L.6.2 Display Contents

A second expression is used to describe the functionality required to display the contents of the file correctly. Files may contain several representations of a single image, so that the expression to display the contents correctly may include several options.

L.6.3 Fallback

In the event that a file cannot be displayed correctly, a fallback method for displaying the file is defined. The fallback methods are intended to be ultimately interoperable, as such, they may not generate exactly correct 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 must 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 IPU-T T.800 | IS 15444-1 Annex I.6, can read the file.

This Recommendation | International Standard defines one additional fallback positions, JPX baseline, as defined in Annex L.2.8. Also, other organizations may register other fallback positions by following the process defined in Annex L.7. 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. Where 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.

L.6.4 Expression Representation

The expressions of the requirements to fully understand all aspects, and to display the file correctly are stored in the Reader Requirements box, which is a mandatory feature of a Reader Requirements box. 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 reader. 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.

L.6.4.1 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

$$(A \,\&\, B \,\&\, C \,\&\, E) \mid (D \,\&\, E), \tag{L.3}$$

is factored into the expression:

$$(A \mid D) \,\&\, (B \mid D) \,\&\, (C \mid D) \,\&\, E. \tag{L.4}$$

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 L-1 Example expression

A	B	C	D	E
1	0	0	1	0

The full expression is written in table form:

Table L-2 Expanded expression

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 L.3 is Display Contents and if the expression for Fully Understand Aspects is:

$$((A \mid D) \,\&\, (B \mid D) \,\&\, (C \mid D) \,\&\, E) \,\&\, (F \mid G), \qquad \text{L.5}$$

then, noting that the expression in Equation L.4 is a common factor, here, the resulting table is:

Table L-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

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 subexpressions 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.

L.6.4.2 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 (encoded using the JPX standard numbering scheme, REF?) 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.

L.6.4.3 Examples

If an image processing program produces a JPX file containing a single codestream 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:

$$\text{sRBG} \;\&\; (\text{single codestream} \mid (\text{multiple codestream} \;\&\; \text{compositing})) \qquad \text{L.6}$$

and to fully understand the file requires:

$$\text{sRGB} \;\&\; (\text{single codestream} \mid (\text{multiple codestream} \;\&\; \text{compositing})) \;\&\; \text{metadata} \qquad \text{L.7}$$

Equation L.6 factors as:

$$\text{sRGB} \;\&\; (\text{single codestream} \mid \text{multiple codestream}) \;\&\; (\text{single codestream} \mid \text{compositing}) \qquad \text{L.8}$$

Equation L.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 below:

Table L-4 Example of a Reader Requirements expressions for Equation L.6 and L.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

The bitmasks indicate which subexpressions are required for each degree of functionality. Thus the expression for `Display Contents` is:

$$(\text{Sub-expr a})\&(\text{Sub-expr b})\&(\text{Sub-expr d}). \qquad \text{L.9}$$

Thus, the above table is stored in the file as:

Table L-5 Example of a Reader Requirements box for Equation L.6 and L.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:

$$(\text{sRGB} \,\&\, \text{single codestream}) \mid (\text{CMYK} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions}) \qquad \text{L.10}$$

while the expression to Understand All Aspects is:

$$((\text{sRGB} \,\&\, \text{single codestream}) \mid (\text{CMYK} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions})) \,\&\, \text{metadata} \,\&\, \text{ACME print metadata} \qquad \text{L.11}$$

Factorising these into sub-expressions gives:

$$\text{single codestream} \,\&\, (\text{sRGB} \mid \text{CMYK}) \,\&\, (\text{sRGB} \mid \text{ACME extensions}) \qquad \text{L.12}$$

and

$$\text{single codestream} \,\&\, (\text{sRGB} \mid \text{CMYK}) \,\&\, (\text{sRGB} \mid \text{ACME extensions}) \,\&\, \text{metadata} \,\&\, \text{ACME print metadata} \qquad \text{L.13}$$

respectively.

The resulting file Reader Requirements table is:

Table L-6 Reader requirements table for Equation L.10 and L.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. Note that 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:

Table L-7 Reader Requirements box data for Equation L.10 and L.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					

Table L-7 Reader Requirements box data for Equation L.10 and L.11

Vendor Feature Compatibility list	ACME extensions UUID					
	0	0	0	1	0	
	ACME print metadata UUID					
	0	0	0	0	1	

L.6.5 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. Note that if a flag is set in the bitmask, then this function is an option in the subexpression 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 L-8 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 (RCM) is `1110`, and the `Fully Understand Aspects` (FUA) mask is `1111`, the reader can perform (FUA & !RCM) 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.

L.7 Extensions to the JPX file format and the registration of extensions

Registration is the process of adding extensions to the capabilities to this standard after the standard has been published. In this standard, many capabilities may be extended through registration. Other items may be extended, but do not require the intervention of a third party to prevent extension conflict. This section identifies those items which may be extended by registration and the process by which capabilities may be registered, as well as identifying those items which may be extended independent of registration and the process by which the Registration Authority will publish the those extensions.

L.7.1 Registration Elements

The registration process is composed of the following elements.

Table L-9 — Registration elements

Element	Identification
Registration Authority	WG1
Submitter	Entity creating the extension to this standard
Review Board	WG1 file format committee
Submission/Item	The proposed extension
Review Board chair	File Format editor
Test	Varies by item

Registration Authority. The organizational entity responsible for reviewing, maintaining, distributing, and acting as a point of contact for all activities related to the registration

Submitter. The submitter is the organization or person who requests that the item be registered.

Review board. The review board is the organizational entity that approves the registration of a proposed item. It is composed of an ad hoc committee appointed by the Review Board Chair

Review board chair. The review board chair is responsible for seeing that each candidate item is considered. He communicates with the submitter through the Registration Authority.

Test. Rational that the Review Board should use to determine if submission/item should be registered

Submission/Item. This is the proposal for registration. Each proposal shall include the name of the item to be extended, the proposed tag/identity for the extension, and a rational/purpose for the extension.

L.7.2 Differentiation between publication and registration

In the JPX CD, several features of the file format may be extended independent of a registration process. 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.

However, to promote interoperability, it is useful to gather the definitions indicated by these UUIDs in one place. In this case, registration of the UUID is not needed to eliminate possible conflict with other vendors, and says nothing about the preferred status of any particular vendor solution.

As such, this proposal clearly differentiates between solutions that require the intervention of a registration committee from those solutions that can be created solely by the individual vendor. This proposal also allows the Registration Authority to label particular proposed element definitions as preferred solutions.

L.7.2.1 Published

Published items are those elements of a JPX file that can be safely extended, generally through the use of URLs or UUIDs, without risk of conflict with other vendors. Values can be assigned for published items without the help of a third-party. However, it is useful to involve a third-party as a single publisher of the definitions of the extended elements from all vendors.

For example, a Vendor Colourspace value is a published item; the value is indicated through the use of a UUID. To promote interoperability, the Registration Authority shall publish a database of all known vendor colourspaces and the colourimetric definitions associated with each UUID.

L.7.2.2 Registered

Registered 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 things if there is not a 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 shall be considered as the specification of a preferred solution.

For example, an Enumerated Colourspace is a registered item; the value is indicated through the use of a 4-byte integer. The Review Board shall evaluate all proposed enumerated colourspaces in terms of preferred technologies. Proposed solutions that are considered preferred solutions shall be allocated a value by the Registration Authority. Proposers of solutions that are not preferred shall be referred to the Vendor Colourspace method as an alternate solution to the proposed problem.

L.7.2.3 Preferred published solutions

In some cases, such as the use of the UUID or XML boxes to embed metadata within a JPX file, there is not a corresponding registered item which can be used for preferred solutions. As such, the Registration Authority, upon recommendation from the Review Board, may choose to label a particular value of a published item as a preferred solution.

L.7.3 Items which can be extended by registration

The following items may be extended by registration. Only items that are listed here may be extended by registration.

Table L-10 — Items which can be extended by registration

Item	Purpose
Enumerated colourspaces	Define additional standard colorspaces
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

L.7.3.1 Enumerated colorspace

New values of the EnumCS field in the Color Specification Box shall be registrable. A proposal to register a new enumerated colorspace must contain a complete colorimetric definition of that colorspace, instructions on how to use images in that colorspace, any required enumerated parameters (for the EP field in the Colour Specification box) and any default values of those parameters.

However, when evaluating proposed enumerated colorspaces, the Review Board shall limit the allocation of enumerated values to international and defacto standards, in addition to determining appropriateness of the proposed solution. Non-standard colorspaces shall be specified through the use of the Vendor Colorspace method.

L.7.3.2 Desired reproduction boxes

New box types for Desired Reproduction information (like the Graphics Technology Standard Output box in the JPX format) shall be registrable. A proposal to register a new desired reproduction must contain a complete definition of the reproduction scenario, including the binary structure of the reproduction data as well as when an application should use the reproduction data.

The Review Board shall evaluate proposed reproductions based on the following criteria:

Does it meet a need not already met by other defined reproductions?

Is the binary format of the reproduction data sufficiently defined?

Is it a general case or a vendor specific case (i.e. output on a typical CRT vs. output on a particular CRT model from a particular vendor)?

The Review Board shall restrict the allocation of Desired Reproduction boxes to general cases that meet needs not already met by other defined reproductions. Other proposed reproductions shall be specified by embedding the data in a UUID box and placing that UUID box within the Desired Reproduction superbox.

L.7.3.3 Compatibility modes

New compatibility modes for the File Type box (values for the CL^i fields) shall be registrable. A proposal to register a new compatibility mode must contain a complete definition of the JPX reader requirements for that compatibility mode, as well as the definition of the 4-byte CL^i field for this mode.

The Review Board shall evaluate proposed compatibility modes based on the following criteria:

Does it meet a need not already met by other compatibility modes?

Is it expected that a wide range of applications will desire to implement support for the particular set of features required by this compatibility mode, or is this mode specific to a particular vendor or application?

Are readers that support this compatibility mode required to support the entire JPX baseline feature set?

Will the creation of this compatibility mode negatively affect interoperability in the target application area?

The Review Board shall restrict the allocation of compatibility modes to cases that meet the needs of a wide range of applications, that are not already met by other modes, and that do not negatively affect interoperability in the target application area. The allocation of modes to feature sets that do not require support for the baseline feature set will be denied in cases where the baseline features are appropriate for the target application. In addition, the

L.7.4 Published items

The following items may be extended, and the Registration Authority shall publish the specifications of those extensions. Only items that are listed here will be published. The text of extension to be published shall be evaluated by the Review

Board before publication by the Registration Authority. In addition, the Review Board may choose to label particular published solutions as preferred, as described below.

Table L-11 — Items which can be extended by registration

Item	Purpose
Vendor feature codes	Define additional vendor specific features
Vendor colourspaces	Define additional vendor-specific colourspaces
Binary filter algorithms	Define additional algorithms for use in the Binary Filter box
UUID metadata	Define additional metadata for use within UUID boxes
XML metadata	Define additional metadata for use within XML boxes

L.7.4.1 Vendor feature codes

The Review Board shall publish the definition of submitted vendor feature codes (values of the VF^i field in the Application Profile box). All submissions must include a complete definition of the feature, including defined data structure, interactions with other data structures, and instructions on how to implement a decoder that supports that feature.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be return it to the submitter for clarification.

L.7.4.2 Vendor colorspaces

The Review Board shall publish the definition of submitted vendor colorspace codes (values of the VCLR field in the METHDAT field for Color Specification boxes that use the Vendor Color method). All submissions must include a complete colorimetric definition of that colorspace, instructions on how to use images in that colorspace, any required vendor parameters (for the VP field in the Color Specification box) and any default values of those parameters.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Review Board may choose to label a particular vendor colorspace as a preferred solution. The committee shall make this decision using the same criteria as would be used when evaluating a proposal for allocation of an enumerated colorspace value (as specified in Annex L.7.3.1).

L.7.4.3 Binary filter algorithms

The Review Board shall publish the definition of submitted binary filter type values (values of the F field in the Binary Filter box). All submissions must include a complete definition of the algorithm and the format of the DATA field in the binary filter box.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Board may choose to label a particular binary filter as a preferred solution. The Board shall reserve this label for international or defacto standards, based on the desired use of the binary filter. For example, encryption technology can be used for both encrypting data and for creating digital signatures. While a particular binary filter may be a preferred solution for encrypting metadata, it may not be preferred for digital signatures.

L.7.4.4 UUID metadata

The Review Board shall publish the definition of submitted UUID s used in UUID boxes. All submissions must include a complete definition of the DATA field in the UUID box and instructions on using that data.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Board may choose to label a particular metadata specification as a preferred solution. The committee shall reserve this label for international or defacto standards, based on the target application for the metadata.

L.7.4.5 XML metadata

The Review Board shall publish the definition of submitted Document Type Definitions (DTD s) and XML Schema s used in XML boxes. All submissions must include a complete definition of the information contained in XML instance documents (found in XML boxes) that use that DTD or schema, as well as instructions on using that data.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the committee may choose to label a particular metadata specification as a preferred solution. The committee shall reserve this label for international or defacto standards, based on the target application for the metadata.

L.7.5 Registration Process

The following is the registration process.

1. A submitter creates a candidate item for registration.
2. The candidate item is submitted to the Registration Authority.
3. The Registration Authority passes the candidate item to the Review Board Chair.
4. The Review Board Chair distributes the candidate item to the Review Board and schedules meetings, phone calls, etc. as appropriate for consideration of the item.

5a. If approved the Chair passes the approval to the Registration Authority who notifies ISO and the submitter, and makes the registered or published item available.

5b. If declined, the Chair prepares a response document indicating why the item was declined and passes this to the Registration Authority who notifies the submitter.

L.7.6 Timeframes for the registration process

L.7.6.1 Requests for registration

The Review Board shall respond to all requests for registration within five months from the date of submission. Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 to evaluate the proposal, make a decision, and draft the response.

L.7.6.2 Requests for publication

The Review Board shall respond to all request for publication within two months from the data of submission. Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 or use e-mail discussions or conference calls to evaluate the proposal, make a decision, and draft the response.

L.7.6.3 Requests for preferred status for published solutions

The Review Board shall respond to all requests for preferred status for published solutions within five months from the date of submission. A request for preferred status may be made at the same time that the request for publication is made.

Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 to evaluate the proposal, make a decision, and draft the response.

L.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 ITU-T T.800 | IS 15444-1 Annex I.6. However, many new boxes are defined, and the structures of several boxes are extended as follows:

The Brand field in the File Type box shall be jpx\040 for files that are completely defined by this Recommendation | International Standard.

Additional forms of the Colour Specification box are defined (Annex L.9.4.2).

The JPEG 2000 compressed codestream may contain extensions as defined in Annex A of this Recommendation | International Standard.

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 Annex L.9.2 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 standard.

L.9 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 standard shall be used for all JPX files.

Figure L-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.

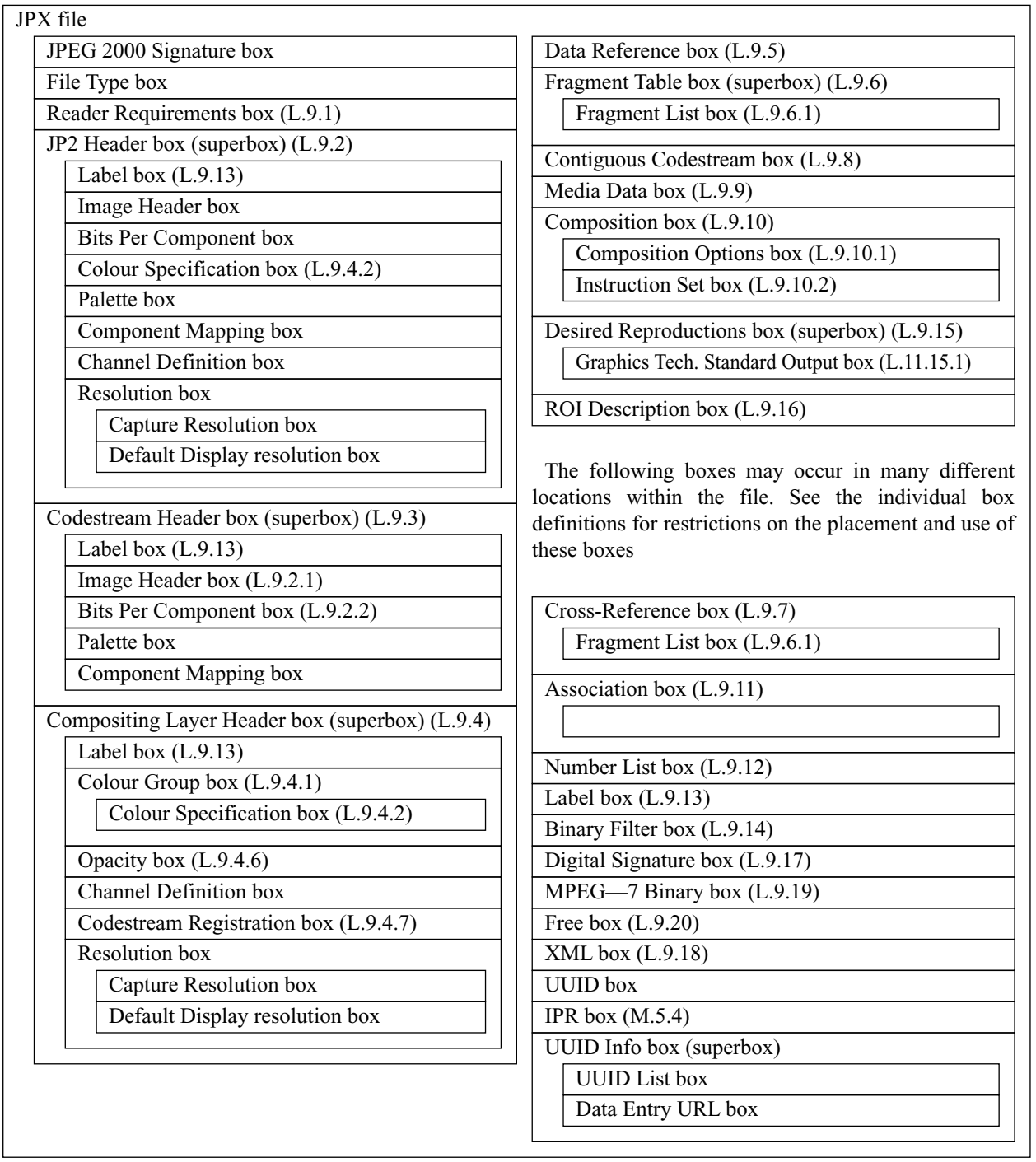

Figure L-6 Boxes defined within a JPX file

Table L-12 lists all boxes defined as part of this Recommendation | International Standard. Boxes defined as part of the JP2 file format are not listed.

Table L-12 — Boxes defined within this Recommendation | International Standard

Box name	Type	Required?	Comments
Reader Requirements box (L.9.1)	rreq (0x7272 6571)	Yes	This box specifies the different modes in which this file may be processed.
JP2 Header box (superbox) (L.9.2)	jp2h (0x6A70 3268)	No	This box specifies JP2 compatibility and default header information for the codestreams and compositing layers.
Image Header box (L.9.2.1)	'ihdr' (0x6968 6472)	Yes	This box specifies the size of the image and other related fields.
Bits Per Component box (L.9.2.2)	'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) (L.9.3)	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) (L.9.4)	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 (L.9.4.1)	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 (L.9.4.2)	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 (L.9.4.6)	opct (0x6F70 6374)	No	This box specifies how opacity information is contained within a set of channels.
Codestream Registration box (L.9.4.7)	creg (0x6372 6567)	No	This box specifies the alignment between the set of codestreams that make up one compositing layer.
Data Reference box (L.9.5)	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) (L.9.6)	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 (L.9.6.1)	flst (0x666C 7374)	No	This box specifies a list of fragments that make up one particular codestream within this JPX file.
Cross-Reference box (L.9.7)	cref (0x6372 6566)	No	This box serves 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 (L.9.8)	jp2c (0x6A70 3263).	No	This box contains one codestream from the JPX file, stored contiguously in a single box.

Table L-12 — Boxes defined within this Recommendation | International Standard

Box name	Type	Required?	Comments
Media Data box (L.9.9)	mdat (0x6D64 6174).	No	This box contains generic media data., which is referenced through the Fragment Table box.
Composition box (L.9.10)	comp (0x636F 6D70)	No	This box specifies how a set of compositing layers shall be combined to create the rendered result.
Composition Options box (L.9.10.1)	copt (0x636F 7074)	No	This box specifies generic options for the composition of multiple compositing layers.
Instruction Set box (L.9.10.2)	inst (0x696E 7374)	No	This box specifies the specific instructions for combining multiple compositing layers to create the rendered result.
Association box (L.9.11)	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 (L.9.12)	nlst (0x6E6C 7374)	No	This box specifies what entities are associated with the data contained within an Association box
Label box (L.9.13)	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 (L.9.14)	bfil (0x6266 696C)	No	This box contains data that has been transformed as part of the storage process (such as compressed or encrypted).
Desired Reproductions box (superbox) (L.9.15)	drep (0x6472 6570)	No	This box specifies a set of transformations that must be applied to the image to guarantee a specific desired reproduction on a set of specific output devices.
Graphics Technology Standard Output box (L.9.15.1)	gtso (0x6774 736F)	No	This box specifies the desired reproduction of the rendered result for commercial printing and proofing systems.
ROI Description box (L.9.16)	roid (0x726F 6964)	No	This box specifies information about specific regions of interest in the image.
Digital Signature box (L.9.17)	chck (0x6368 636B)	No	This box contains a checksum or digital signature for a portion of the JPX file.
MPEG—7 Binary box (L.9.19)	mp7b (0x6D70 3762)	No	This box contains metadata in MPEG-7 binary format (BiM) as defined by ISO/IEC [CD] 15938
Free box (L.9.20)	free (0x6672 6565)	No	This box contains data that is no longer used and may be overwritten when the file is updated.

L.9.1 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 must be supported by a reader in order to fully use the file. The Reader Requirements box must immediately follow the File Type box, and there shall be one and only one Reader Requirements box in the file.

The type of an Reader Requirements box shall be ‘rreq’ (0x7272 6571’). The contents of the Reader Requirements box is as follows:

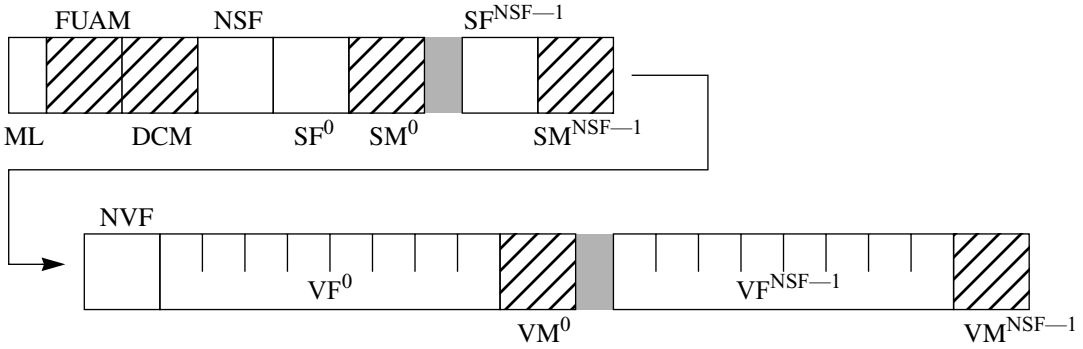

Figure L-7 Organization of the contents of the Reader Requirements box

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: Decode Completely 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 SF^i 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 as follows:

Table L-13 — Legal values of the SF^i field

Value	Meaning
	Codestream contains no extensions
	Contains multiple composition layers
	Codestream is compressed using JPEG 2000 and requires at least a Level 0 decoder
	Codestream is compressed using JPEG 2000 and requires at least a Level 1 decoder
	Codestream is compressed using JPEG 2000 and requires at least a Level 2 decoder
	Codestream is compressed using DCT
	Does not contain opacity

Table L-13 — Legal values of the SF^i field

Value	Meaning
	Compositing layer includes opacity channel (non-premultiplied)
	Compositing layer includes premultiplied channel opacity
	Compositing layer specifies opacity using a chroma-key value
	Codestream is contiguous
	Codestream is fragmented such that fragments are all in file and in order
	Codestream is fragmented such that fragments are all in file but out of order
	Codestream is fragmented such that fragments are in multiple local files
	Codestream is fragmented such that fragments are across the internet
	Rendered result created using compositing
	Support for compositing layers is not required (reader can load a single, discrete compositing layer)
	Contains multiple, discrete layers that should not be combined through either animation or compositing
	Compositing layers each contain only a single codestream
	Compositing layers contain multiple codestreams
	All compositing layers are in the same colourspace
	Compositing layers are in multiple colorspaces
	Rendered result created without using animation
	Animated, but first layer covers entire area and is opaque
	Animated, but first layer does not cover the entire rendered result area
	Animated, and no layer is reused
	Animated, but layers are reused
	Animated with persistent frames only
	Animated with non-persistent frames
	Rendered result created without using scaling
	Rendered result involves scaling within a layer
	Rendered result involves scaling between layers
	Contains ROI metadata
	Contains IPR metadata
	Contains Content metadata
	Contains History metadata
	Contains Creation metadata
	Portion of file is digitally signed in a secure method

Table L-13 — Legal values of the SF^i field

Value	Meaning
	Portion of file is checksummed
	Desired Graphic Arts reproduction specified
	Compositing layer uses palettized color
	Compositing layer uses Restricted ICC profile
	Compositing layer uses Any ICC profile
	Compositing layer uses sRGB enumerated colorspace
	Compositing layer uses sRGB-grey enumerated colorspace
	Compositing layer uses BiLevel 1 enumerated colorspace
	Compositing layer uses BiLevel 2 enumerated colorspace
	Compositing layer uses YCbCr 1 enumerated colorspace
	Compositing layer uses YCbCr 2 enumerated colorspace
	Compositing layer uses YCbCr 3 enumerated colorspace
	Compositing layer uses PhotoYCC enumerated colorspace
	Compositing layer uses YCCK enumerated colorspace
	Compositing layer uses CMY enumerated colorspace
	Compositing layer uses CMYK enumerated colorspace
	Compositing layer uses CIELab enumerated colorspace with default parameters
	Compositing layer uses CIELab enumerated colorspace with parameters
	Compositing layer uses CIEJab enumerated colorspace with default parameters
	Compositing layer uses CIEJab enumerated colorspace with parameters
	Compositing layer uses e-sRGB enumerated colorspace
	Compositing layer uses ROMM—RGB enumerated colorspace
	Compositing layers have non-square samples
	Compositing layers have labels
	Codestreams have labels
	Compositing layers have different color spaces
	Compositing layers have different metadata

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 L-14 — Format of the contents of the Reader Requirements box

Field name	Size (bits)	Value
ML	8	1,2,4 or 8
EM	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

L.9.2 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 codestream specified by the JP2 Header box, 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 Annex L.9.2.1 and Annex L.9.2.2 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 application profiles may restrict the placement of this box.

L.9.2.1 Image Header box

The format and structure of the Image Header box is identical to that defined in Annex I.5.3.1 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:

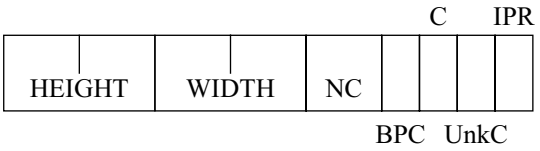

Figure L-8 — Organization of the contents of an Image Header box

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 specifies the bit depth of the components in the codestream, minus 1, and is stored as a 1-byte field.

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) must contain a Bits Per Component box defining the bit depth of each component (as defined in Annex I.5.3.2 in the JP2 file format).

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: Compression type. This parameter specifies the compression algorithm used to compress the image data. Legal values of this field are as follows:

Table L-15 — 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 8, sample values shall be packed into bytes so that no bits are unused between samples. However, each sample shall begin on a byte boundary and padding bits having value zero shall be inserted after the last sample of a scan line as necessary to fill out the last byte of the scan line. Simple values appear in component-interleaved order. When multiple sample values are packed into a byte, the first sample shall appear in the most significant bits of the byte. When a sample is larger than a byte, its most significant bit shall appear in earlier bytes.
1	**Recommendation T.4, the basic algorithm known as MH (Modified Huffman).** This value is only permitted for bi-level images.
2	**Recommendation T.4, commonly known as MR (Modified READ).** This value is only permitted for bi-level images.

Table L-15 — Legal C values

Value	Meaning
3	**Recommendation T.6, commonly known as MMR (Modified Modified READ).** This value is only permitted for bi-level images.
4	**ITU—T rec. T.82 | ISO/IEC 11544**Commonly known as JBIG. This value is only permitted for bi-level images.
5	**CCITT Rec. T.81 | ISO/IEC 10918—1 or ITU—T Rec.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.**
other values	Reserved for ISO use

UnkC: Colourspace Unknown. The value of this field is identical to that defined for the JP2 file format.

IPR: Intellectual Property.The value of this field is identical to that defined for the JP2 file format.

Table L-16 — BPC values

Values (bits) MSB LSB	Component 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 values
1xxx xxxx	Components are signed values
1111 1111	Components vary in bit depth
	All other values reserved for ISO use.

Table L-17 — 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 L-16
C	8	7
Unk	8	0 — 1
IPR	8	0 — 1

L.9.2.2 Bits Per Component box

The Bits Per Component box specifies the bit depth of each component. The structure of this box is identical to that defined in Annex I.5.3.2 in the JP2 file format. However, if the compression type of the codestream corresponding to this Bits Per Component box is not JPEG 2000, then the value of the field in this box shall match the respective bits per component data in the respective codestream format specification.

L.9.3 Codestream Header box (superbox)

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

Both codestreams and Codestream Header boxes are numbered separately, starting with 0, by their order in the file. Codestream Header box *i* shall be applied to codestream *i*. There shall either be one Codestream Header box in the file for each codestream, or there shall be zero Codestream Header boxes in the file. In the event that there are zero Codestream Header boxes, then the header information for all of the codestreams shall be taken to be the default header information contained within the JP2 Header box.

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

The type of a Codestream Header box shall be ‘jpch’ (0x6A70 6368). The contents of a Codestream Header box is as follows:

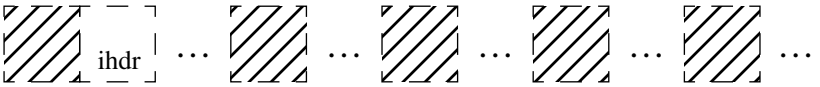

Figure L-9 — Organization of the contents of a Codestream Header box

label: Label box. This box specifies a label for this codestream. Its structure is specified in Annex L.9.13.

ihdr: Image Header box. This box specifies information about this codestream, such as its height and width. Its structure is specified in Annex L.9.2.1. 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 Annex L.9.2.2.

pclr: Palette box. This box defines the palette to use to create multiple components from a single component. Its structure is specified in Annex I.5.3.4 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 Annex I.5.3.5 of the JP2 file format.

roid: ROI Description box. This box describes regions of interest within this codestream. These ROI’s may or may not be directly associated with coded ROI’s in the codestream. Its structure is defined in Annex L.9.16.

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.

L.9.4 Compositing Layer Header box (superbox)

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

The type of a Compositing Layer Header box shall be 'jplh' (0x6A70 6C68). The contents of a Compositing Layer Header box is as follows:

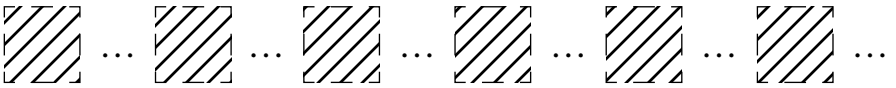

Figure L-10 — Organization of the contents of a Compositing Layer Header box

label: Label box. This box specifies a label for this compositing layer. Its structure is specified in Annex L.9.13.

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 Annex L.9.4.1. 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 Annex L.9.4.6. 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 Annex I.5.3.6 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 Annex L.9.4.7.

res: Resolution box. This box specifies the capture and default display resolutions of the image. Its structure is specified in Annex I.5.3.7 of the JP2 file format.

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.

L.9.4.1 Colour Group box

A Colour group box defines a set of 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.

A Colour Group box (or the JP2 Header box) shall not contain multiple Colour Specifications boxes with a METH value of 1 (Enumerated method), multiple boxes with a METH value of 1 (Restricted ICC method). A single colour group may contain multiple Colour Specification boxes with a METH value of 2 (Any ICC method) or 3 (Vendor Colour method). Multiple ICC profiles (or 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.

If the file is JP2 compatible (as indicated in the File Type box), then the JPX file may contain zero Colour Group boxes, indicating 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:

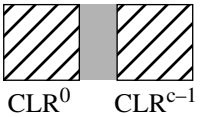

Figure L-11 — Organization of the contents of a Colour Group box

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 Annex L.9.4.2

L.9.4.2 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 image data after it has been decompressed and after any reverse decorrelating component transform and non-linearity point transform has been applied to the decompressed image data.

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:

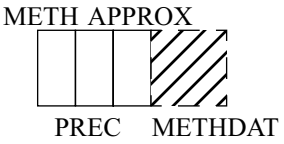

Figure L-12 — Organization of the contents of a Colour Specification box

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 follows:

Table L-18 — 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 Annex L.9.4.3.1, as well as additional parameter for some enumerated colourspaces as specified in Annex L.9.4.4.
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 Annex I.5.3.3.
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 ICC-1. The binary format of the METHDAT field is specified in Annex L.9.4.3.2.
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 Annex L.9.4.3.3.
other values	Reserved for other ISO use. 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 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. Examples of quantization 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:

Table L-19 — 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
other values	Reserved for other ISO use.

Contrary to the APPROX field in a JP2 file, a value of 0 in the APPROX field is illegal in a JPX file. JPX writers are required to properly indicate the degree of approximation of the colour specification to the correct definition of the colourspace. If the actual colourspace is unknown, then the value of the Unk 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 mask the specification of a separate colourspace definition within a single Colour Group box (or within the JP2 Header box).

Table L-20 — 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

L.9.4.3 METHDAT field specifications in the Colour Specification box

The following sections define the fields and values that make up the METHDAT field for each defined colour specification method.

L.9.4.3.1 METHDAT values for the Enumerated method

The contents of the METHDAT field for Colour Specification boxes using the Enumerated method is defined as follows:

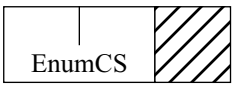

Figure L-13 — Organization of the contents of the METHDAT field for the Enumerated method

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 must 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 L-21):

Table L-21 — 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	**YCbCr(1).** This is a format often used for data that originated from a video signal. The colourspace is based on Recommendation 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. Recommendation ITU—R BT.601—5 specifies these ranges as well as defines a 3x3 matrix transformation that can be used to convert these samples into RGB.
3	**YCbCr(2).** This is the most commonly used format for image data that was originally captured in RGB (uncalibrated format). The colourspace is based on Recommendation 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 Recommendation ITU-R BT.601—5. Recommendation ITU-R BT.601—5 specifies a 3x3 matrix transform that can be used to convert these samples into RGB.
4	**YCbCr(3).** This is a format often used for data that originated from a video signal. The colourspace is based on Recommendation 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. Recommendation ITU-R BT.601—5 specifies these ranges as well as defines a 3x3 matrix transform 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 Recommendation ITU-R BT.709 reference primaries. Recommendation ITU-R BT.709 linear RGB image signals are transformed to non-linear R’G’B’ values to YCC corresponding to Recommendation 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.

Table L-21 — Additional legal EnumCS values

Value	Meaning
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 transform specified for $YC_bC_r(2)$ above, and then recombining the result with the unmodified K-sample. This transform 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 Annex L.9.4.4.1
18	**Bi-level(2)**.This value shall be used to indicate bi-level images. Each image sample is one bit: 1 = white, 0 = black.
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 Annex L.9.4.4.2
20	**e-sRGB.**As defined by PIMA 7667
21	**ROMM—RGB**As defined by PIMA 7666
22	sRGB based YCbCr
23	**YPbPr(1125/60).** This is the well known color space and value definition for the HDTV (1125/60/2:1) system for production and international program exchange specified by Recommendation ITU-R BT.709-3. The Recommendation specifies the color 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.
24	**YPbPr(1250/50).** This is the well known color space and value definition for the HDTV (1250/50/2:1) system for production and international program exchange specified by Recommendation ITU-R BT.709-3. The Recommendation specifies the color 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.
other values	Reserved for other ISO uses

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 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 Annex L.9.4.4. 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 L-22 — 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

L.9.4.3.2 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:

PROFILE

Figure L-14 — 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 L-23 — Format of the contents of the METHDAT field for the Any ICC method

Field name	Size (bits)	Value
PROFILE	Variable	Variable

L.9.4.3.3 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:

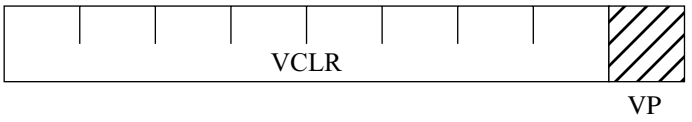

Figure L-15 — Organization of the contents of the METHDAT field for the Vendor Colour method

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 must 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 standard.

VP: Vendor parameters. This field 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 L-24 — Format of the contents of the METHDAT field for the Vendor Colour method

Field name	Size (bits)	Value
VCLR	128	Variable
VP	Variable	Variable

L.9.4.4 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.

L.9.4.4.1 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:

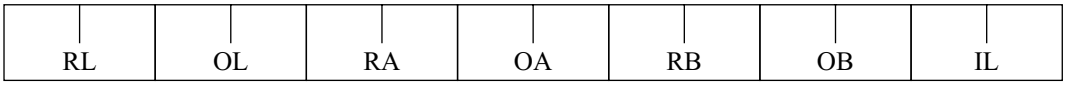

Figure L-16 — Organization of the contents of the EP field for the CIELab (EnumCS = 14)

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 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. According to ITU—T Rec. T.42, the calculations from real values L*a*b* to $n_a n_b$ bit integers, which are expressed by $N_L N_a N_b$, are made as follows:

$$N_L = \frac{2^{n_L} - 1}{RL} \times L^* + OL$$

$$N_a = \frac{2^{n_a} - 1}{RA} \times a^* + OA \qquad \text{L.14}$$

$$N_b = \frac{2^{n_b} - 1}{RB} \times b^* + OB$$

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 L.14. It is encoded as a 4-byte big endian unsigned integer.

OL: Offset for L*. This field specifies the *OL* value from Equation L.14. It is encoded as a 4-byte big endian unsigned integer.

RA: Range for a*. This field specifies the *RA* value from Equation L.14. It is encoded as a 4-byte big endian unsigned integer.

OA: Offset for a*. This field specifies the *OA* value from Equation L.14. It is encoded as a 4-byte big endian unsigned integer.

RB: Range for b*. This field specifies the *RB* value from Equation L.14. It is encoded as a 4-byte big endian unsigned integer.

OB: Offset for b*. This field specifies the *OB* value from Equation L.14. It is encoded as a 4-byte big endian unsigned integer

IL: Illuminant: This field specifies the illuminant data used in calculating the CIELAB values. Rather than specify the XYZ values of the normalizing illuminant, which are used in calculating CIELAB, the specification of the illuminant data follows ITU-T Rec. T-4 Annex E. The illuminant data consists of 4

bytes, identifying the illuminant. In the case of a standard illuminant, the 4 bytes are one of the following:

Table L-25 — 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 color 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 is omitted for the CIELab colourspace, then the following default values shall be used. The offset parameters are dependent on the number of bits per component and are defined in Ref3 for values for the 8 and 12 bit cases. For 8 bits, the L*, a* and b* offset parameter are 0, 128 and 96 respectively. For 12 bits, the value are 0, 248 and 1536, respectively. The L*, a*, and b* Range default values are 100, 255 and 255 respectively; the Range parameters are not dependent on the bit depth of the component. The default value of the IL field is 0x0044 3530, specifying CIE Illuminant D50.

Table L-26 — 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

L.9.4.4.2 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:

Figure L-17 — 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 $N_JN_aN_b$ bit integers, which are expressed by $N_JN_aN_b$, are made as follows:

$$N_J = \frac{2^{n_J} - 1}{RJ} \times J + OJ$$
$$N_a = \frac{2^{n_a} - 1}{RA} \times a + OA \qquad \text{L.15}$$
$$N_b = \frac{2^{n_b} - 1}{RB} \times b + OB$$

RJ: Range for J. This field specifies the *RJ* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 100 for RJ shall be used.

OJ: Offset for J. This field specifies the *OJ* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value 0 shall be used for OJ.

RA: Range for a. This field specifies the *RA* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 255 for RA shall be used.

OA: Offset for a. This field specifies the *OA* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value $2^{b—1}$ shall be used for OJ, where *b* is the number of bits per sample for the 'a' channel.

RB: Range for b. This field specifies the *RB* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 255 for RB shall be used.

OB: Offset for b. This field specifies the *OB* value from Equation L.15. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value $2^{b—1}$ shall be used for OJ, where *b* is the number of bits per sample for the 'b' channel.

Table L-27 — 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)

Table L-27 — Format of the contents of the EP field for CIEJab (EnumCS = 19)

Field name	Size (bits)	Value
OA	32	$0 — (2^{32}–1)$
RB	32	$0 — (2^{32}–1)$
OB	32	$0 — (2^{32}–1)$

L.9.4.5 Channel Definition box

The format and meaning of the Channel Definition box is identical to that defined in ITU-T T.800 | IS 15444-1 Annex I.5.3.6. However, the following additional value of the $Asoc^i$ field are normatively defined:

Table L-28 — 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

L.9.4.6 Opacity box

The Opacity box provides a minimal-overhead mechanism for specifying opacity through a chroma-key or specifying that this 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 alpha is a form of paletization and as such images using chroma-keyed alpha must 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 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, is that only the samples of the image that are of the chroma-key value must 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.

The type of the Opacity box shall be ‘opct’ (0x6F70 6374). The contents of this box shall be as follows:

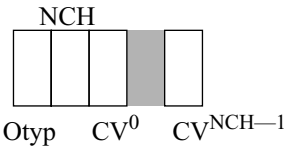

Figure L-18 — Organization of the contents of an Opacity box

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 L-29 — 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^1 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^1 fields.
3 — 255	Reserved for ISO use

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 is padded to a multiple of 8 bits 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 10 bits, then the CV^i value shall be stored in the low 10 bits of a 16 bit field.

Table L-30 — 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

L.9.4.7 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. 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:

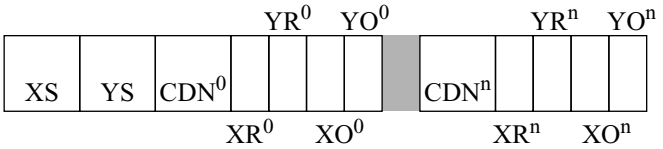

Figure L-19 — Organization of the contents of a Codestream Registration box

XS: Horizontal grid size. This field specifies the number of horizontal grid points on the compositing layer registration grid 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 specifies the number of vertical grid points on the compositing layer registration grid 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 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 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.

XO^i: Horizontal offset. This field specifies the horizontal distance from center of the top left point on the reference grid of the codestream specified by the CDN^i parameter to the center of the top left point on the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

YO^i: Vertical offset. This field specifies the vertical distance from center of the top left point on the reference grid of the codestream specified by the CDN^i parameter to the center of the top left point on the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

Table L-31 — 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

L.9.5 Data Reference box

The Data Reference box contains an array of URLs which are referenced by this file in a repeating pattern. 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. There shall be at most one Data Reference box within a JPX file. 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.

Also, while this box does contain other boxes, it is not a superbox; the Data Reference box shall not generally contain any box.

The type of the Data Reference box shall be dtbl (0x6474 626C), and its contents shall be as follows:

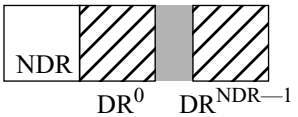

Figure L-20 Organization of the contents of a Data Reference box

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 ITU-T T.800 | IS 15444-1 Annex 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.

Table L-32 — Format of the contents of the Data Reference box

Field name	Size (bits)	Value
NDR	16	0 (2^{16}—1)
DR^i	Variable	Variable

L.9.6 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; they shall not be found within a superbox.

The type of the Fragment Table box shall be ftbl (0x6674 626C), and its contents shall be as follows:

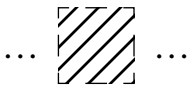

Figure L-21 Organization of the contents of a Fragment Table box

FLst: Fragment List.This field contains a Fragment List box as specified in Annex L.9.6.1.

L.9.6.1 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.

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, than all offsets in all Fragment list boxes shall be interpreted with respect to the stored length of the Binary Filter boxes, 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:

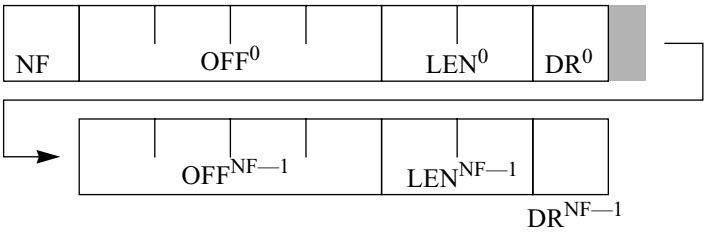

Figure L-22 Organization of the contents of a Fragment List box

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 file. The offset is relative to the first byte of the file (the first byte of the length field of the JPEG 2000 signature box). This field is encoded as a 64-bit unsigned integer. Only the first fragment in a Fragment List contained within a Cross-Reference box shall point to the first byte of a box header.

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 Data Reference Table box. This field is encoded as a 16-bit unsigned integer.

Table L-33 Format of the contents of the Fragment List box

Parameter	Size (bits)	Value
NF	16	0 (2^{16}—1)
OFF^i	64	12 (2^{64}—1)

Table L-33 Format of the contents of the Fragment List box

Parameter	Size (bits)	Value
LEN^i	32	0 (2^{32}—1)
DR^i	16	0 (2^{16}—1)

L.9.7 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:

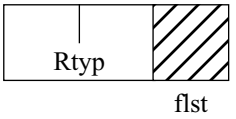

Figure L-23 Organization of the contents of a Fragment table box

Rtyp: Referenced box type. This field specifies the actual type of the box referenced by this Cross-Reference box.

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. The format of the Fragment List box is specified in Annex L.9.6.1.

Table L-34 Format of the contents of the Cross-Reference box

Parameter	Size (bits)	Value
Rtyp	32	0 (2^{32}—1)
flst	Variable	Variable

L.9.8 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 ITU-T T.800 | IS 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.

L.9.9 Media Data box

The Media Data box contains fragments of the JPEG 2000 codestream or other media data. Applications should not access Media Data boxes directly, but instead use the fragment table to determine what parts of what 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.

L.9.10 Composition box

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 Annex L.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:

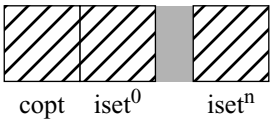

Figure L-24 Organization of the contents of a Composition box

copt: Composition Options box. This box specifies parameters that apply to the composition or animation as a whole. It is defined in Annex L.9.10.1.

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 Annex L.9.10.2

Table L-35 Format of the contents of the Composition box

Parameter	Size (bits)	Value
copt	Variable	Variable
iset^{i}	Variable	Variable

L.9.10.1 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:

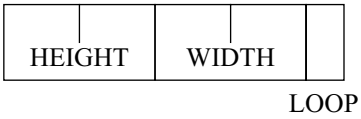

Figure L-25 — Organization of the contents of an Image Header box

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.

LOOP: Looping count. This field specifies the number of times to fully execute the display instructions. 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.

Table L-36 Format of the contents of the Composition Options box

Parameter	Size (bits)	Value
HEIGHT	32	1 °2 ^{32}—1
WIDTH	32	1 °2 ^{32}—1
LOOP	8	0 255

L.9.10.2 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 in the Composition box. Instruction Set boxes shall be found only within a Composition box; they shall not be found in any other locations in the file.

The type of the Instruction Set box shall be ‘inst’ (0x696E 7374) and contents of the box shall have the following format:

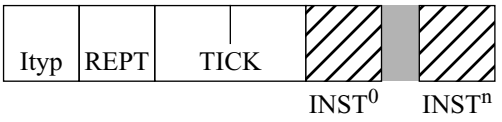

Figure L-26 — Organization of the contents of an Instruction Set box

Ityp: Instruction type. This field the type of this instructions, 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 L-37 — 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.
All other bit flags	Reserved for ISO use

REPT: Repetition. This field specifies the number of times to repeat this particular instruction. 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 Annex L.9.10.2.1.

Table L-38 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)

Table L-38 Format of the contents of the Instruction Set box

Parameter	Size (bits)	Value
$INST^i$	Variable	Variable

L.9.10.2.1 Instruction parameter

The following illustration shows the contents of each individual INST field (a single compositing instruction) within an Instruction Set box:

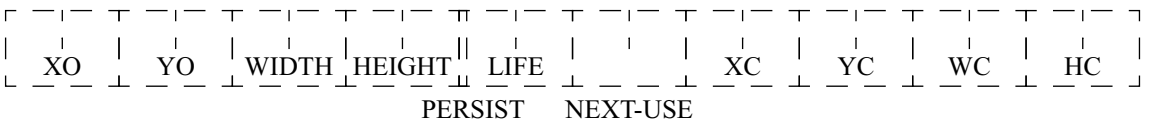

Figure L-27 — Organization of the contents of an INST field within an Instruction Set box

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 its 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 the 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 LOOP parameter in the Composition Options box. The composition layer passed on for reuse in this manor must 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 width. 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.

Table L-39 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^{a} 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

a. References to Ityp within the individual instruction parameters refers to the Ityp field within the Instruction Set box that contains this Instruction

L.9.11 Association box

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 one or more boxes. It creates independent semantic associations between the boxes it contains or the entities represented by those boxes. In particular, associations are create between the first box (or entities represented by it) (BF) and each of the other boxes (or represented entities) (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.

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 with 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 it s metadata with an image or image set. These examples are illustrated in Figures L-28 to L-31.

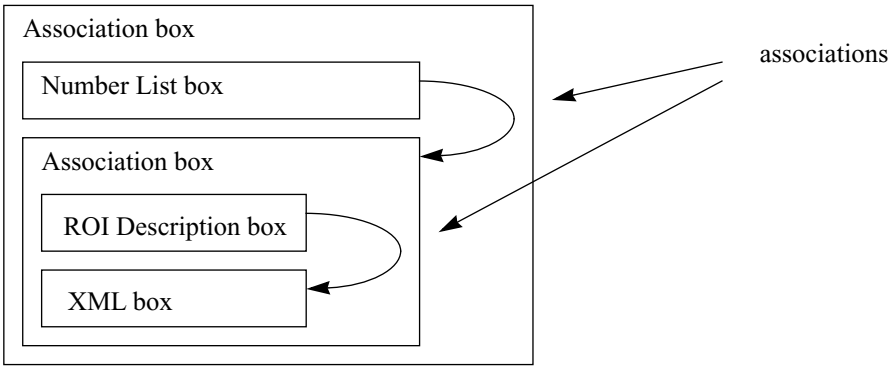

Figure L-28 Example of ROI specific metadata associated with one or more images

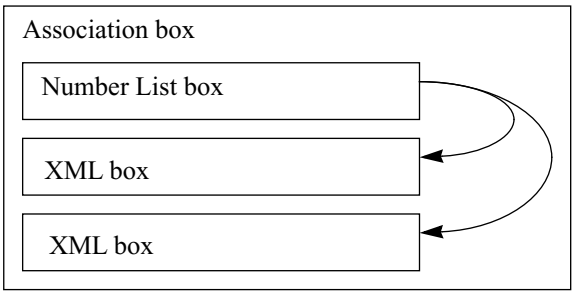

Figure L-29 Example of Multiple XML documents associated with one or more images

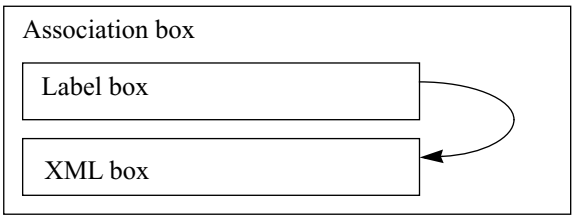

Figure L-30 Example of a Labeled XML document

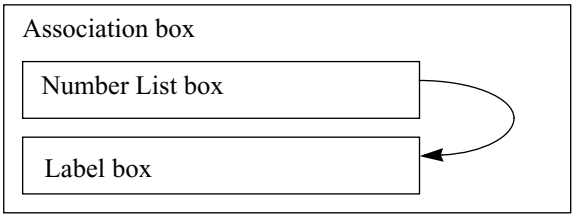

Figure L-31 Example of a labeled image

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:

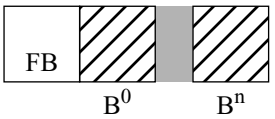

Figure L-32 Organization of the contents of an Association box

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 the box BF.

Table L-40 Format of the contents of the Association box

Parameter	Size (bits)	Value
BF	Variable	Variable
B^i	Variable	Variable

L.9.12 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:

Figure L-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 as follows:

Table L-41 — 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 *in* the JPX file.
0xFFXX XXXX	The lower three order bytes (of value *i*) specify an entity which has been numbered explicitly in the file
All other values of the high byte	Reserved for ISO use

Table L-42 Format of the contents of the Number List box

Parameter	Size (bits)	Value
AN^i	32	0 (2^{32}—1)

L.9.13 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 with 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:

Figure L-34 Organization of the contents of a Label box

S: Label string. A textual label associated with the entity or entities referred to by the Number List in the same Association box. This value is stored 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. Also, label strings are not null-terminated or padded in any other way; every character that is present is significant.

L.9.14 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:

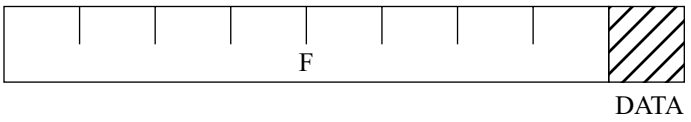

Figure L-35 Organization of the contents of a Binary Filter box

F: Filter type. This field specifies how the data was transformed before storage. This value is encoded as a UUID. Standard defined values are:

Table L-43 — 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.
Other values	Reserved for ISO use.

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. Note that 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.

Table L-44 Format of the contents of the Binary Filter box

Parameter	Size (bits)	Value
F	128	Variable
DATA	Variable	Variable

L.9.15 Desired Reproductions box (superbox)

The Desired Reproductions box specifies a set of transformations that must 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 must 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 must 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.

However, 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:

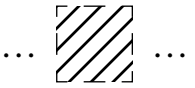

Figure L-36 Organization of the contents of the Output ICC Profile box

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 Annex L.9.15.1

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.

L.9.15.1 Graphics Technology Standard Output box

A Graphics Technology Standard Output box specifies the desired reproduction of the rendered result for chimerical 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, which 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:

OUTP

Figure L-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).

Table L-45 Format of the contents of the Output ICC Profile box

Parameter	Size (bits)	Value
OUTP	Variable	Variable

L.9.16 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 ROI s described in this box are not necessarily coded as ROI s within the codestream; it also allows an application or user to signal the importance of certain parts of an image even if theses 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 or a Codestream Header box (or within an Association box within either of those boxes). If the ROI Description box is found within a Codestream Header box, then the ROI s 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 ROI s described at a top-level box are not directly associated with coded ROI s within any codestream.

The type of the ROI Description box shall be roid (0x726F 6964). The contents of this box shall be as follows:

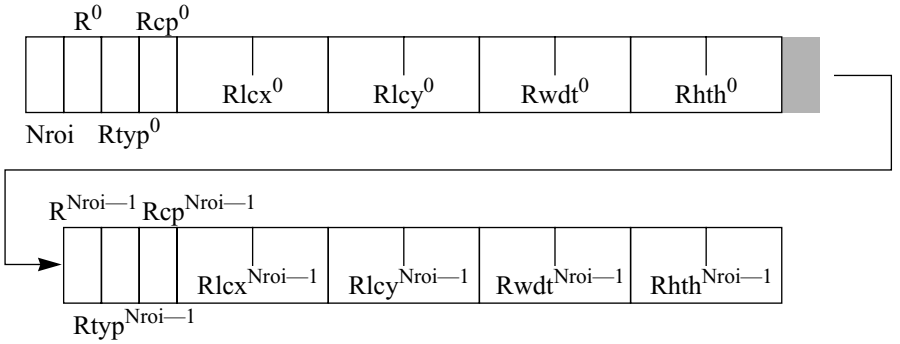

Figure L-38 Organization of the contents of the ROI Description box

Nroi: Number of Regions of Interest. Encoded as a 8 bit integer.

R^i: Region of Interest present in codestream. Encoded as a 8 bit integer. Legal values of the R^i field are as follows:

Table L-46 — 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
2 255	Reserved for ISO use.

$Rtyp^i$: Region of Interest type, can be either rectangular or ellipse. Encoded as a 8 bit integer. Legal values of the $Rtyp^i$ field are as follows:

Table L-47 — Legal $Rtyp^i$ values

Value	Meaning
0	Rectangular region of interest
1	Elliptical region of interest
2 255	Reserved for ISO 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 rectangular area this is the location of the top left corner of the rectangle. In the case of an elliptic Region of Interest this is the horizontal position of the center point. This value is stored as a 4-byte big endian unsigned integer.

$Rlcy^i$: Region of Interest vertical location. In the case of rectangular area this is the location of the top left corner of the rectangle. In the case of an elliptic Region of Interest this is the vertical position of the center point. This value is stored as a 4-byte big endian unsigned integer.

$Rwdt^i$: Region of Interest width. In the case of rectangular Region of Interest this is the width of the rectangle. In the case of an elliptic Region of Interest this is the horizontal axis. This value is stored as a 4-byte big endian unsigned integer.

$Rhth^i$: Region of Interest height. In the case of rectangular Region of Interest this is the height of the rectangle. In the case of an elliptic Region of Interest this is the vertical axis. This value is stored as a 4-byte big endian unsigned integer.

Table L-48 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	1 (2^{32}—1)
$Rlcy^i$	32	1 (2^{32}—1)
$Rwdt^i$	32	1 (2^{32}—1)
$Rhth^i$	32	1 (2^{32}—1)

L.9.17 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 is 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:

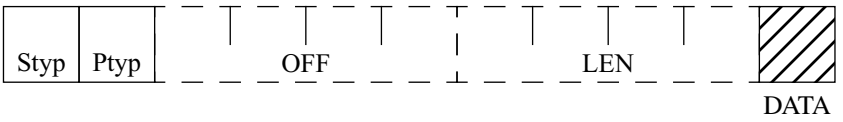

Figure L-39 — Organization of the contents of a Digital Signature box

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 follows:.

Table L-49 — Legal Styp values

Value	Meaning
0	A checksum, generated by applying the MD5 algorithm 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 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.
2	The digital signature is generated by applying the DSA algorithm, as defined in FIPS 186—1, to the source data. The resulting signature is stored in the DATA field of this Digital Signature box.
3	A digital signature, generated by applying the RSA signature algorithm with the MD5 message digest algorithm (according to PKCS #1 Version 1.5) to the source data, is stored in the DATA field of this Digital Signature box.
4	A digital signature, generated by applying the RSA signature algorithm with the SHA-1 message digest algorithm (according to PKCS #1 Version 1.5) to the source data, is stored in the DATA field of this Digital Signature box.
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.
6 255	Reserved for ISO use.

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.

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 follows:

Table L-50 — 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 must be at the top level of the file.
1	The source data that is signed by this Digital Signature box shall be a range 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.
2 255	Reserved for ISO use.

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.

Table L-51 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

L.9.18 XML box

The JP2 file format defines the XML box to contain XML data as defined by REC—XML—19980210. 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 instance document:

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.

The XML schema or DTD document should contain a comment field that contains the URL of a human readable document that describes the schema or DTD. This document will support application developers and users of the metadata. That comment shall be in the following form:

<!--HUMAN_SCHEMA_DTD_LOCATION: *LOC* --> where *LOC* is the URL of the human readable document that describes the schema or DTD.

If the XML document or instance data contains neither the xsi:schemaLocation attribute nor the !DOCTYPE line, then this XML document or instance data is not well-formed as defined by the XML specification.

L.9.19 MPEG—7 Binary box

This box contains metadata in MPEG—7 binary format (BiM) as defined by ISO/IEC [CD] 15938.

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:

Figure L-40 Organization of the contents of a MPEG—7 Binary box

DATA:MPEG—7 BiM Stream

L.9.20 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.

L.10 Dealing with unknown boxes

A conforming JP2 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.

Annex M

JPX file format extended metadata definition and syntax

(This Annex forms a normative and integral part of this Recommendation | International Standard.
This Annex is optional to a JPX reader.)

This Annex defines a comprehensive set of metadata elements that may be embedded in a JPX file within XML boxes. Use of this form of metadata is optional. Metadata encoded according to this Annex shall be either correctly interpret or ignored by a JPX reader.

M.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.

M.2 Additional references for extended metadata

ASTM. Standard Practice for Electronic Interchange of Color and Appearance Data. E1708-95. 1995

DIG. DIG35 Specification - Metadata for Digital Images. Version 1.0. August 2000

IETF. Tags for Identification of Languages. RFC 1766. March 1995

IETF. Uniform Resource Identifiers (URI). RFC 2396. August 1998

IETF. vCard MIME Directory Profile. RFC 2426. September 1998

ISO. Photography - Electronic still picture cameras - Determination of ISO speed. ISO 12232:1998.

ISO. Photography - Electronic still picture cameras - Resolution measurements. ISO/DIS 12233.

ISO. Photography - Electronic still picture imaging - Removable memory - Part 2: Image data format - TIFF/EP. ISO/DIS 12234-2

ISO. Photography - Electronic still picture cameras - Methods for measuring opto-electronic conversion functions (OECF's). ISO/DIS 14524.

JEIDA. Digital Still Camera File Format Standard (Exif). Version 2.1. June 1998

John S. Denker. See How It Files. 1996

NMEA. NMEA 0183: Standard For Interfacing Marine Electronic Devices. Version 2.30. March 1998

WIPO. Berne Convention for the Protection of Literary and Artistic Works. Paris Act of July 24. 1971. amended September 28. 1979.

WIPO. World Intellectual Property Organization Copyright Treaty. 1996.

W3C, XML Schema Part 1: Structures, CR-xmlschema-1-20001024, <http://www.w3.org/TR/xmlschema-1>.

W3C, XML Schema Part 2: Datatypes, CR-xmlschema-2-20001024, <http://www.w3.org/TR/xmlschema-2>.

M.3 Scope of metadata definitions

This Annex consists of four logical groups of metadata as well as common definitions of dataypes that is referred to by other metadata definitions. While each group is logically partitioned, they may be linked to each other to form additional semantics.

M.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.

M.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.

M.3.3 Metadata History metadata

The Metadata 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.

M.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.

M.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 the are commonly referenced within other metadata groups. These include a definition for language specification and a timestamp.

M.4 Metadata syntax

As defined in ITU-T T.800 | IS 15444-1 Annex I, the JP2 file format allows XML format metadata to be contained within the box structure. Metadata defined in this Annex shall be well formed XML as defined by REC-xml-19980210 and validated by the DTD defined in Annex M.8. The default character encoding shall be UTF-8 otherwise specified in the XML document.

M.4.1 Metadata schema definition language (informative)

This Recommendation | International Standard uses the XML Schema syntax as defined by XML Schema Part 1 (CR-xmlschema-1-20001024) and XML Schema Part 2 (CR-xmlschema-2-20001024) to describe the elements of the metadata.

M.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:jp2="http://www.jpeg.org/jp2"
```

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/2000/10/XMLSchema"
```

M.4.3 Document type definitions

A XML Document type definition (DTD) for this Recommendation | International Standard is defined by the DTD specified in Annex M.8.

The Formal Public Identifier (FPI) shall be:

```
PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN"
```

This FPI must be used on the DOCTYPE statement within a XML document referencing the DTD defined by this Recommendation | International Standard.

The URI for the DTD shall be:

```
"http://www.jpeg.org/jp2/FCD15444-2.dtd
```

The Name in the document type declaration shall be set to the root element name for the defined box.

M.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. A JPX reader which supports the metadata defined in this Annex shall understand all the elements within each box.

M.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 pedigree of the image.

The type of the Image Creation box shall be xml\040 (0x786D 6C20) as defined in ITU-T T.800 | IS 15444-1 Annex I.7.1. The contents of this box shall be as follows:

ICre

Figure M-1 Organization of the contents of Image Creation box

ICre: Image Creation metadata field. This field shall be valid XML as defined by REC-xml-19980210.

Table M-1 Format of the contents of the Image Creation box

Field name	Size (bits)	Value
ICre	Variable	The XML document shall include the following DOCTYPE declaration and metadata defined in Annex M.6.1: <!DOCTYPE IMAGE_CREATION PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN" "http://www.jpeg.org/FCD15444-2.dtd">

M.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 | IS 15444-1 Annex I.7.1. The contents of this box shall be as follows:

CDes

Figure M-2 Organization of the contents of Content Description box

CDes: Content Description field. This field shall be valid XML as defined by REC-xml-19980210.

Table M-2 Format of the contents of the Content Description box

Field name	Size (bits)	Value
CDes	Variable	The XML document shall include the following DOCTYPE declaration and metadata defined in Annex M.6.2: <!DOCTYPE CONTENT_DESCRIPTION PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN" "http://www.jpeg.org/FCD15444-2.dtd">

M.5.3 Metadata History box

This box contains the history of metadata of an image. The Metadata history 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 Metadata 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 Metadata history box shall be xml\040 (0x786D 6C20) as defined in ITU-T T.800 | IS 15444-1 Annex I.7.1. The contents of this box shall be as follows:

MHist

Figure M-3 Organization of the contents of Metadata History box

MHist:Metadata History field. This field shall be valid XML as defined by REC-xml-19980210.

Table M-3 Format of the contents of the Metadata History box

Field name	Size (bits)	Value
MHist	Variable	The XML document shall include the following DOCTYPE declaration and metadata defined in Annex M.6.3: <!DOCTYPE HISTORY PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN" "http://www.jpeg.org/FCD15444-2.dtd">

M.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 (0x6266 696C) as defined in ITU-T T.800 | IS 15444-1 Annex I.6. The contents of this box shall be as follows:

IPR

Figure M-4 Organisation of the contents of Intellectual Property Rights box

IPR: Intellectual Property Rights field. This field shall be valid XML as defined by REC-xml-19980210.

Table M-4 Format of the contents of the Content Description box

Field name	Size (bits)	Value
IPR	Variable	The XML document shall include the following DOCTYPE declaration and metadata defined in Annex M.6.4: <!DOCTYPE IPR PUBLIC "-//SC29WG1/DTD JPXXML/XML//EN" "http://www.jpeg.org/FCD15444-2.dtd">

M.6 Metadata definitions

Each of the following metadata elements is based on the XML format as defined in REC-xml-19980210. The metadata shall be either correctly interpreted or ignored by a JPX reader.

M.6.1 Image Creation metadata

This element specifies information that are relevant to the creation of the image file. The image creation metadata is stored in a number of optional sub-elements.

```
<xsd:element name="IMAGE_CREATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="GENERAL_CREATION_INFO" minOccurs="0"/>
      <xsd:element ref="CAMERA_CAPTURE" minOccurs="0"/>
      <xsd:element ref="SCANNER_CAPTURE" minOccurs="0"/>
      <xsd:element ref="CAPTURED_ITEM" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-5 Schema of the Image Creation metadata

GENERAL_CREATION_INFO: General creation information. This element specifies generic information on how the image was created. Its contents are specified in Annex M.6.1.1.

CAMERA_CAPTURE: This element specifies a camera capture metadata of a scene. Its contents are specified in Annex M.6.1.2.

SCANNER_CAPTURE: This element specifies scanner capture metadata that may be used for various scanners such as flatbed and film scanners. Its contents are specified in Annex M.6.1.8.

CAPTURED_ITEM: This element contains description of the item that was digitally captured. Its contents are specified in Annex M.6.1.10.

M.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..

```
<xsd:element name="GENERAL_CREATION_INFO">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="CREATION_TIME" type="xsd:timeInstant" minOccurs="0"/>
      <xsd:element name="IMAGE_SOURCE" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="SCENE_TYPE" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="IMAGE_CREATOR" type="jp2:tPerson" minOccurs="0"/>
      <xsd:element name="OPERATOR_ORG" type="jp2:tOrganization" minOccurs="0"/>
      <xsd:element name="OPERATOR_ID" type="jp2:tLangString" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-6 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 M-5 lists suggested for this element.

Table M-5 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 M-6 lists suggested for this element.

Table M-6 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 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 for the format of this element.

OPERATOR_ID: This element specifies a name or ID for the person conducting the capture process.

M.6.1.2 Camera Capture metadata

This element specifies a camera capture of a scene. It optionally contains camera and lens information, device characterization and camera capture settings.

```
<xsd:element name="CAMERA_CAPTURE">
 <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="CAMERA_INFO" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element name="SOFTWARE_INFO" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element name="LENS_INFO" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element ref="jp2:DEVICE" minOccurs="0"/>
      <xsd:element ref="jp2:CAMERA_SETTINGS" minOccurs="0"/>
      <xsd:element name="ACCESSORY" type="jp2:tProductDetails" minOccurs="0"
                                                               maxOccurs=”unbounded”/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-7 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 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 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 for the format of this element.

DEVICE_CHARACTER: Device characterization. This element specifies the technical characterization of the digital capture device. The syntax of the DEVICE_CHARACTER element is specified in Annex M.6.1.3.

CAMERA_SETTINGS: Camera capture settings. This element specifies the camera settings used when the image was captured. The syntax of the CAMERA_SETTINGS element is specified in Annex M.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 for the format of this element.

M.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.

```
<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="jp2:tIntSize" minOccurs="0"/>
      <xsd:element name="SPECTRAL_SENSITIVITY" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:any processContents="skip"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="ISO_SATURATION" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="ISO_NOISE" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element ref="SPATIAL_FREQ_RESPONSE" minOccurs="0"/>
      <xsd:element ref="CFA_PATTERN" minOccurs="0"/>
      <xsd:element ref="OECF" minOccurs="0"/>
      <xsd:element name="MIN_F_NUMBER" type="jp2:tNonNegativeDouble" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-8 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 M-7 lists suggested values for this element.

Table M-7 Sensor technology values

Value	Meaning
One-Chip Color Area	An one-chip color area sensor technology used.
Two-Chip Color Area	A two-chip color area sensor technology used.
Three-Chip Color Area	A three-chip color area sensor technology used.
Color Sequential Area	A color sequential area sensor technology used.
Trilinear	An trilinear sensor technology used.
Color Sequential Linear Sensor	A color sequential linear sensor technology used.

FOCAL_PLANE_RES: Focal plane resolution. This element specifies the number of pixels in the X (width) and Y (height) directions for the main image. The width and height stored are the width and height 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.

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 the SPATIAL_FREQ_RESPONSE element is specified in Annex M.6.1.4.

CFA_PATTERN: Color filter array pattern. This element specifies the color filter array (CFA) pattern of the image sensor used to capture a single-sensor color image. The syntax of the CFA_PATTERN element is specified in Annex M.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 the OECF element is specified in Annex M.6.1.6.

MIN_F_NUMBER: Minimum F-number. This element specifies the minimum lens f-number of the camera or image capturing device.

M.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.

```
<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="jp2:tNonNegativeDouble"/>
            <xsd:element name="HORIZ_SFR" type="jp2:tNonNegativeDouble"/>
            <xsd:element name="VERT_SFR" type="jp2:tNonNegativeDouble"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-9 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.

M.6.1.5 Color Filter Array Pattern metadata

This element encodes the actual color filter array (CFA) geometric pattern of the image sensor used to capture a single-sensor color image. It is not relevant for all sensing methods. The data contains the minimum number of rows and columns of filter color values that uniquely specify the color filter array.

```
<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 M-10 Schema of the Color Filter Array Pattern metadata

COLOR_ROW: This element specifies the list of color values of the CRA pattern.

COLOR:CFA pattern values. The values shall be either Red, Green, Blue, Cyan, Magenta, Yellow, or White.

M.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.

```
<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="jp2:tNonNegativeDouble"
                                                          maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    <xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-11 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.

M.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.

```
<xsd:element name="CAMERA_SETTINGS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:element name="EXP_TIME" type="jp2:tNonNegativeDouble"/>
        <xsd:element name="R_EXP_TIME" type="jp2:tRational"/>
      </xsd:choice>

      <xsd:element name="F_NUMBER" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="EXP_PROGRAM" type="jp2: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="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name ="METERING_MODE" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name ="SCENE_ILLUMINANT" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="COLOR_TEMP" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="FOCAL_LENGTH" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="FLASH" type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="FLASH_ENERGY" type="jp2: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="jp2: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="jp2:tLocation" minOccurs="0"/>
      <xsd:element name="ORIENTATION" type="jp2:tDirection" minOccurs="0"/>
      <xsd:element name="PAR" type="jp2:tRational" minOccurs="0"/>

      <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
      <xsd:attribute ref="xml:lang"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-12 Schema of the Camera Capture Setting 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 M-8 lists suggested for this element.

Table M-8 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 under-exposure 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 M-9 lists suggested values for this element.

SCENE_ILLUMINANT: This element specifies the light source (scene illuminant) that was present when the image was captured. Table M-10 lists suggested values for this element.

COLOR_TEMP: Color temperature. This element specifies the actual color temperature value of the scene illuminant stored in units of Kelvin.

Table M-9 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.

Table M-10 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.

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 know. 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 M-11 lists BACK_LIGHT values used for lighting situations.

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 M-12 lists values used for auto focus status.

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

Table M-11 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 center and the surrounding area is greater than one full step (APEX). The frame is exposed for the subject center.
Back Light 2	The brightness value difference between the subject center and the surrounding area is greater than one full step (APEX). The frame is exposed for the surrounding area.

Table M-12 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.

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 Colored, 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.

M.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.

```
<xsd:element name="SCANNER_CAPTURE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="SCANNER_INFO" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element name="SOFTWARE_INFO" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element ref="SCANNER_SETTINGS" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-13 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 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 for the format of this element.

SCANNER_SETTINGS: This element specifies the scanner settings used when the image was scanned. The syntax of the SCANNER_SETTINGS element is specified in Annex M.6.1.9.

M.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.

```
<xsd:element name="SCANNER_SETTINGS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="PIXEL_SIZE" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="PHYSICAL_SCAN_RES" type="jp2:tDoubleSize" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-14 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 element 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.

M.6.1.10 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.

```
<xsd:element name="CAPTURED_ITEM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice>
        <xsd:element ref="REFLECTION_PRINT" minOccurs="0"/>
        <xsd:element ref="FILM" minOccurs="0"/>
      </xsd:choice>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-15 Schema of the Captured Item metadata

REFLECTION_PRINT: This element specifies information about a reflection print that was digitally captured. The syntax of the REFLECTION_PRINT element is specified in Annex M.6.1.11.

FILM: This element specifies information about the film. The syntax of the FILM element is specified in Annex M.6.1.12.

M.6.1.11 Reflection Print metadata

This element specifies information about a reflection print that was digitally captured. This element may contain the sub-elements listed below.

```
<xsd:element name="REFLECTION_PRINT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="DOCUMENT_SIZE" type="jp2: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 M-16 Schema of the Reflection Print metadata

DOCUMENT_SIZE: This element specifies the lengths of the X (width) and Y (width) 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.

M.6.1.12 Film metadata

This element specifies information on the film that was digitized. This element may contain the sub-elements listed below.

```
<xsd:element name="FILM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="BRAND" type="jp2:tProductDetails" minOccurs="0"/>
      <xsd:element name="CATEGORY" minOccurs="0">
        <xsd:simpleType base="xsd:string">
          <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="jp2:tDoubleSize" minOccurs="0"/>
      <xsd:element name="ROLL_ID" type="jp2: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 name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-17 Schema of the Film metadata

BRAND: This element specifies the name of the film manufacturer. See Product Details Type for the format of this element.

CATEGORY: This element specifies the category of film used. Legal values include Negative B/W, Negative Color, Reversal B/W, Reversal Color, Chromagenic, Internegative B/W, and Internegative Color. The category Chromagenic refers to B/W negative film that is developed with a C41 process (i.e., color 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 ASA.

M.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.

```
<xsd:element name="CONTENT_DESCRIPTION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="GROUP_CAPTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="CAPTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="CAPTURE_TIME" type="jp2:tDateTime" minOccurs="0"/>
      <xsd:element name="LOCATION" type="jp2:tLocation" minOccurs="0"/>
      <xsd:element ref="PERSON" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="THING" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ORGANIZATION" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="EVENT" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="AUDIO" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="DICTIONARY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-18 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 (Annex M.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 (Annex M.7.1.15) for the format of this element.

PERSON: Person Description. This element specifies a person within an image. Its contents are specified in Person Description metadata (see Annex M.6.2.1).

THING: Thing Description. This element specifies the names of tangible things depicted in the image. Its contents are specified in Thing Description metadata (see Annex M.6.2.2).

ORGANIZATION: Organization Description. This element specifies an organization within an image. Its contents are specified in Organization Description metadata (see Annex M.6.2.3).

EVENT: Event description. This element specifies events depicted in the image. Its contents are specified in Event metadata (see Annex M.6.2.4).

AUDIO: This element specifies audio streams associated with an image. Its contents are specified in Audio metadata (see Annex M.6.2.7).

PROPERTY: This element specifies information used to describe an image or an object within an image. Its contents are specified in Property metadata (see Annex M.6.2.8).

DICTIONARY: This element specifies a dictionary of a property. Its contents are specified in Dictionary metadata (see Annex M.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 (Annex M.7.3.1) for the format of this element.

M.6.2.1 Person Description metadata

This element specifies a person within an image. See Person type (Annex M.7.1.13) for the format of this element. This element may contain the sub-elements listed below.

```
<xsd:element name="PERSON">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jp2:tPerson">
        <xsd:sequence>
          <xsd:element name="POSITION" type="jp2:tPosition" minOccurs="0"/>
          <xsd:element name="LOCATION" type="jp2:tLocation" minOccurs="0"/>
          <xsd:element ref="PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Figure M-19 Schema of the Person Description metadata

POSITION: This element specifies the position of the person within the image. See Position type (Annex M.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 (Annex M.7.1.15) for the format of this element.

PROPERTY: This element specifies additional information describing the person. See Property metadata (Annex M.6.2.8) for the format of this element.

M.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.

```
<xsd:element name="THING">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NAME" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
      <xsd:element name="POSITION" type="jp2:tPosition" minOccurs="0"/>
      <xsd:element name="LOCATION" type="jp2:tLocation" minOccurs="0"/>
      <xsd:element ref="PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="THING" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>

    <xsd:attribute name="ID" type="xsd:string"/>
    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-20 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 (Annex M.7.3.1) for more information on this element.

POSITION: The position of the thing within the image can be specified. The format of this element is defined in Position type (see Annex M.7.1.17).

LOCATION: The physical location of the thing within the image can be specified. This element does not specify the relative position of the thing within the image. The format of this element is defined in Location type (see Annex M.7.1.15).

PROPERTY: The Thing also contains multiple Propertys. These Propertys describe the thing. See Property metadata (Annex M.6.2.8) for the format of this element.

THINGSub-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.

M.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 (Annex M.7.1.14) for the format of this element. This element may contain the sub-elements listed below.

```
<xsd:element name="ORGANIZATION">
  <xsd:complexType base="jp2:tOrganization" derivedBy="extension">
    <xsd:complexContent>
      <xsd:extension base="jp2:tOrganization">
        <xsd:sequence>
          <xsd:element name="POSITION" type="jp2:tPosition" minOccurs="0"/>
          <xsd:element name="LOCATION" type="jp2:tLocation" minOccurs="0"/>
          <xsd:element ref="PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Figure M-21 Schema of the Organization Description metadata

POSITION: This element specifies the position of the organization within the image. See Position type (Annex M.7.1.17) for the format of this element.

LOCATION: The physical location of the organization. This element does not specify the relative position of the organization. See Location type (Annex M.7.1.15) for the format of this element.

PROPERTY: This element specifies additional information describing the organization. See Property metadata (Annex M.6.2.8) for the format of this element.

M.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.

```
<xsd:element name="EVENT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="EVENT_TYPE" type="jp2:tLangString"/>
      <xsd:element name="DESCRIPTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="LOCATION" type="jp2:tLocation" minOccurs="0"/>
      <xsd:element name="EVENT_TIME" type="jp2:tDateTime" minOccurs="0"/>
      <xsd:element name="DURATION" type="xsd:timeDuration" minOccurs="0"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
      <xsd:element ref="PARTICIPANT" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="EVENT_RELATION" minOccurs="0" maxOccurs="unbounded"/>
      <!-- Sub-events -->
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="EVENT"/>
        <xsd:element name="EVENT_REF" type="xsd:string"/>
      </xsd:choice>
    </xsd:sequence>

    <xsd:attribute name="ID" type="xsd:string"/>
    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-22 Schema of the Event Description metadata

EVENT_TYPE: Event type. If there is an Event or Sub-event element the Event type element must 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 (Annex M.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 (Annex M.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 (Annex M.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 the PARTICIPANT element is specified in Annex M.6.2.5.

EVENT_RELATION: Event relationships. This element specifies relationships to other events. The syntax of the PARTICIPANT element is specified in Annex M.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.

M.6.2.5 Participant metadata

This element specifies the participants in the event. A participant may be a Person, an Organization or a Thing.

```
<xsd:element name="PARTICIPANT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ROLE" type="jp2:tLangString" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice>
        <xsd:element name="OBJECT_REF" type="xsd:string"/>
        <xsd:element ref="PERSON"/>
        <xsd:element ref="THING"/>
        <xsd:element ref="ORGANIZATION"/>
      </xsd:choice>
    </xsd:sequence>

    <xsd:attribute name="xml:lang" type="xsd:language"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-23 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. Its contents are specified in Person description metadata (see Annex M.6.2.1).

THING: This element specifies a Thing that is a participant of an event yet not depicted within the image. Its contents are specified in Thing description metadata (see Annex M.6.2.2).

ORGANIZATION: This element specifies the Organization that is a participant of an event yet not depicted within the image. Its contents are specified in Organization description metadata (see Annex M.6.2.3).

M.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.

```
<xsd:element name="EVENT_RELATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="RELATION" type="jp2:tLangString" minOccurs="0"
                                                          maxOccurs="unbounded"/>
      <xsd:element name="EVENT_REF" type="xsd:string" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-24 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.

M.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.

```
<xsd:element name="AUDIO">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="AUDIO_STREAM" type="xsd:uriReference"/>
      <xsd:element name="AUDIO_FORMAT" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="MIME_TYPE" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DESCRIPTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-25 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 (Annex M.7.3.1) for more information on this element.

M.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.

```
<xsd:element name="PROPERTY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NAME" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="VALUE" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
      <xsd:element ref="PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>

    <xsd:attribute name="DICT_REF" type="xsd:string"/>
    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-26 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 (Annex M.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 Annex M.6.2.9).

M.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.

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="jp2:tLangString" minOccurs="0"/>
      <xsd:element ref="COMMENT" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="DICT_ID" type="xsd:string"/>
    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-27 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 (Annex M.7.3.1) for more information on this element.

DICT_ID: Dictionary ID. This element specifies the unique identifier for the dictionary.

M.6.3 Metadata History metadata

This element may contain the sub-elements listed below.

```
<xsd:element name="HISTORY">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="PROCESSING_SUMMARY" minOccurs="0"/>
      <xsd:element ref="IMAGE_PROCESSING_HINTS" minOccurs="0"/>
      <xsd:element name="METADATA" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="IMAGE_CREATION" minOccurs="0"/>
            <xsd:element ref="CONTENT_DESCRIPTION" minOccurs="0"/>
            <xsd:element ref="HISTORY" minOccurs="0"/>
            <xsd:element ref="IPR" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-28 Schema of the Metadata History metadata

PROCESSING_SUMMARY: This element specifies a list of operations previously applied to an image during the course of its workflow. Its contents are specified in Processing Summary metadata (see Annex M.6.3.1).

IMAGE_PROCESSING_HINTS: This element specifies a list of the operations previously performed when editing an image. Its contents are specified in Image Processing Hints metadata (see Annex M.6.3.2).

METADATA: Previous History 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). Its contents are specified in Previous History metadata (see Annex M.6.3.3).

M.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.

```
<xsd:element name="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 name="TIMESTAMP" type="xsd:timeInstant"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-29 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 a image has been transformed.

IMG_GTC_ADJ: Global Tone / Color Adjustment. The presence of this element indicates that a contrast or density adjustment has been applied to the image, or that the image coloring has been adjusted.

IMG_STC_ADJ: Selective Tone / Color 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 a 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.

M.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 fields as the Processing Summary metadata. See Processing Summary (Annex M.6.3.1) for the definition of each element. Each sub-element may appear more than once within each field and each field 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 a Image Cropped hint is no assurance that the image has never been cropped.

```
<xsd:element name="IMAGE_PROCESSING_HINTS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="IMG_CREATED" type="jp2:tLangString"/>
        <xsd:element name="IMG_CROPPED" type="jp2:tLangString"/>
        <xsd:element name="IMG_TRANSFORMED" type="jp2:tLangString"/>
        <xsd:element name="IMG_GTC_ADJ" type="jp2:tLangString"/>
        <xsd:element name="IMG_STC_ADJ" type="jp2:tLangString"/>
        <xsd:element name="IMG_SPATIAL_ADJ" type="jp2:tLangString"/>
        <xsd:element name="IMG_EXT_EDITED" type="jp2:tLangString"/>
        <xsd:element name="IMG_RETOUCHED" type="jp2:tLangString"/>
        <xsd:element name="IMG_COMPOSITED" type="jp2:tLangString"/>
        <xsd:element name="IMG_METADATA" type="jp2:tLangString"/>
      </xsd:choice>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-30 Schema of the Image Processing Hints metadata

M.6.3.3 Previous History metadata

This element contains a previous version of the metadata (including previous history information). The format of this element is defined in Metadata History metadata (Figure M-28).

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 metadata history. The contributing image(s) Image Creation, Content Description, and IPR metadata may be recorded in a Previous History Metadata element. Careful consideration shall be made with regards to this previous metadata, particularly previous IPR information.

M.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., an 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. However, implementors should take into account the World Intellectual Property Organization (WIPO) documents listed in the References and other WIPO publications.

This element may contain the sub-elements listed below.

```
<xsd:element name="IPR">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="IPR_NAMES" minOccurs="0"/>
      <xsd:element ref="IPR_DESCRIPTION" minOccurs="0"/>
      <xsd:element ref="IPR_DATES" minOccurs="0"/>
      <xsd:element ref="IPR_EXPLOITATION" minOccurs="0"/>
      <xsd:element ref="IPR_IDENTIFICATION" minOccurs="0"/>
      <xsd:element ref="IPR_CONTACT_POINT" minOccurs="0"/>
      <xsd:element ref="IPR_HISTORY" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="IPR" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-31 Schema of the Intellectual Property Rights metadata

IPR_NAMES: This element specifies names related to the represented image. Its contents is specified in IPR Names metadata (see Annex M.6.4.1).

IPR_DESCRIPTION: This element specifies the description of the content such as the title and caption. Its contents is specified in IPR Description metadata (see Annex M.6.4.2).

IPR_DATES: This element specifies the IPR-related date information. Its contents is specified in IPR Dates metadata (see Annex M.6.4.3).

IPR_EXPLOITATION: This element specifies exploitation information such as type of protection, use restriction and obligations to exploit an image. Its contents is specified in IPR Exploitation metadata (see Annex M.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. Its contents is specified in IPR Identification metadata (see Annex M.6.4.6).

IPR_CONTACT_POINT: This element specifies the contact point of the right holder. Its contents is specified in IPR Contact Point metadata (see Annex M.6.4.9).

IPR_HISTORY: This element contains previous IPR information. Its content is specified in IPR History (see Annex M.6.4.10).

M.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 (Annex M.7.1.13) and Organization type (Annex M.7.1.14), respectively for the format of this element. This element may contain the sub-elements listed below.

```
<xsd:element name="IPR_NAMES">
  <xsd:complexType>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element name="IPR_PERSON">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="jp2: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="jp2: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>
    </xsd:choice>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-32 Schema of the IPR Names metadata

IPR_PERSON: Person. This element specifies the person description. See Person type (Annex M.7.1.13) for the format of this element.

IPR_ORG: Organization. This element specifies the organization description. See Organization type (Annex M.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 M-13 lists suggested for this element and have the following meanings.

Table M-13 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.

M.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.

```
<xsd:element name="IPR_DESCRIPTION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_TITLE" type="jp2:langString" minOccurs="0"/>
      <xsd:element name="IPR_LEGEND" type="jp2:langString" minOccurs="0"/>
      <xsd:element name="IPR_CAPTION" type="jp2:langString" minOccurs="0"/>
      <xsd:element name="COPYRIGHT" type="jp2:langString" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-33 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.

M.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.

```
<xsd:element name="IPR_DATES">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_DATE maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="jp2:tDateTime">
              <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-34 Schema of the IPR Dates metadata

IPR_DATE: The date element contains a date of arbitrary precision. See DateTime type (Annex M.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 field.

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 M-14 lists suggested for this element and have the following meanings.

Table M-14 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.

M.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.

```
<xsd:element name="IPR_EXPLOITATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_PROTECTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_USE_RESTRICTION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_OBLIGATION" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element ref="IPR_MGMT_SYS" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-35 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 for JPEG Utilities Registration Authority (JURA) use. Other values may exist. If this element is not present, then the watermark content (or it s 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 format of this element is specified in IPR Management System metadata (see Annex M.6.4.5).

M.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. These information is commonly stored on a server describing the IPR of the image, and depending upon whether IPR licensing is mandatory or recommended, there must be a link to where all information about it is kept. This element may contain the sub-elements listed below.

```
<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="jp2:reference" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-36 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.

M.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 must be registered again, and a new identifier must be acquired, because there are now two objects instead of merely one. However, the parent image must appear in the metadata set of the child. This element may contain the sub-elements listed below.

```
<xsd:element name="IPR_IDENTIFICATION">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="IPR_IDENTIFIER" minOccurs="0"/>
      <xsd:element ref="LICENCE_PLATE" minOccurs="0"/>
    </xsd:sequence>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-37 Schema of the IPR Identification metadata

IPR_IDENTIFIER: Generic IPR identifier. This element contains a generic purpose IPR identifier. The format of this element is specified in Generic IPR Identifier metadata (see Annex M.6.4.7).

LICENCE_PLATE: This element specifies License plate of the content. The format of this element is specified in Licence Plate metadata (see Annex M.6.4.8).

M.6.4.7 Generic IPR Identifier metadata

This element specifies a generic IPR identifier. This element may contain the sub-elements listed below.

```
<xsd:element name="IPR_IDENTIFIER">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IPR_ID_MODE" type="jp2:tLangString" minOccurs="0"/>
      <xsd:element name="IPR_ID" type="jp2:tLangString" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-38 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.

M.6.4.8 License Plate metadata

This element specifies the license plate of the original image, defined in ISO 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.

```
<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:timeInstant" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-39 Schema of the Licence 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.

M.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. Failing to fill this element would result in the impossibility to use the image, unless another element gives the link to the author.

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.

```
<xsd:element name="IPR_CONTACT_POINT">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="jp2:IPR_PERSON"/>
      <xsd:element ref="jp2:IPR_ORG"/>
      <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>
    </xsd:choice>

    <xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-40 Schema of the IPR Contact Point metadata

IPR_PERSON: This element specifies the person description. The format of this element is defined in IPR Names (see Annex M.6.4.1).

IPR_ORG: Organization. This element specifies the organization description. The format of this element is defined in IPR Names (see Annex M.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 Annex M.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 M-13. The value listed in Table M-15 is added and have the following meaning.

Table M-15 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

M.6.4.10 IPR History metadata

This element specifies previous IPR metadata. The format of this element is defined in Intellectual Property Rights metadata (Figure M-31).

Each time an IPR information of an image is updated, some of the IPR metadata defined through Annex M.6.4.1 and Annex M.6.4.9 may be moved to this IPR History metadata element. The IPR History metadata stores all IPR metadata-related modifications.

M.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 section 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.

M.7.1 Defined types

M.7.1.1 Non-negative double type

This type is used for double numbers greater than or equal to zero.

```
<xsd:simpleType name="tNonNegativeDouble">
  <xsd:restriction base="xsd:double">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure M-41 Schema of the non-negative double type

M.7.1.2 Rational type

This type is used to define rational numbers. It contains an enumerator and denominator in a single string.

```
<xsd:simpleType name="tRational">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(\-|\+)?[0-9]+/[0-9]+"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure M-42 Schema of the rational type

M.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.

```
<xsd:complexType name="tLangString">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute ref="xml:lang"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Figure M-43 Schema of the string incluting language attribute type

M.7.1.4 Degree type

This type specifies a direction in degrees and fractions of degrees. The exact meaning of the values is dependant on usage.

```
<xsd:simpleType name="tDegree">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="-180"/>
    <xsd:maxInclusive value="180"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure M-44 Schema of the degree type

M.7.1.5 Half degree type

This type specifies a direction in degrees and fractions of degrees. The exact meaning of the values is dependant on usage. This type defines a smaller range than Degree Type (see Annex M.7.1.4).

```
<xsd:simpleType name="tHalfDegree">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="-90"/>
    <xsd:maxInclusive value="90"/>
  </xsd:restriction>
</xsd:simpleType>
```

Figure M-45 Schema of the half degree type

M.7.1.6 Double size type

This type specifies a size in double coordinates.

```
<xsd:complexType name="tDoubleSize">
  <xsd:sequence>
    <xsd:element name="WIDTH"type="jp2:tNonNegativeDouble"/>
    <xsd:element name="HEIGHT"type="jp2:tNonNegativeDouble"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure M-46 Schema of the double size type

M.7.1.7 Integer size type

This type specifies a size in integer coordinates (e.g. pixels).

```
<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 M-47 Schema of the integer size type

M.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 more broad definition such as Winter . A date may or may not include a time. This type may contain the sub-elements listed below.

```
<xsd:complexType name="tDateTime">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element name="EXACT" type="xsd:timeInstant"/>
      <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:year" minOccurs="0"/>
        <xsd:element name="CENTURY" type="xsd:century" minOccurs="0"/>
      </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="jp2:COMMENT" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-48 Schema of the Date Time 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 .

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 (Annex M.7.3.1) for more information on this element. Examples include Easter Sunday, Morning, Just after lunch.

M.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.

```
<xsd:complexType name="tAddress">
  <xsd:sequence>
    <xsd:element name="ADDR_NAME" type="jp2:tLangString" minOccurs="0"/>
    <xsd:element name="ADDR_COMP" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="jp2:tLangString">
            <xsd:attribute name="TYPE" type="xsd:string"/>
          </xsd:extension>
        </xsd:complexContent>
      </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="jp2:tLangString" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute name="TYPE" type="xsd:string"/>
  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-49 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 M-16. Multiple values shall not be specified within a single element.

Table M-16 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 zipcode) of the address. This element is not limited in length. The element has the title "Postcode" or "Zipcode. An address cannot contain both a postcode and a zipcode.

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 M-17. Multiple type values may be specified delimited with a comma (,).

Table M-17 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.

M.7.1.10 Phone number type

This type specifies a phone number. This type may contain the sub-elements listed below.

```
<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="jp2:TIMESTAMP"/>
</xsd:complexType>
```

Figure M-50 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 M-18. Multiple type values may be specified delimited with a comma (,).

Table M-18 Phone number type values

Value	Meaning
Home	Phone number associated with a residence.
Message	Phone number that has voice massage support.
Work	Phone number associated with a place of work.
Voice	Phone number to 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.

M.7.1.11 Email address type

This type specifies an email address. This type may contain the sub-elements listed below.

```
<xsd:complexType name="tEmail">
  <xsd:complexContent>
    <xsd:extension base="jp2:tLangString">
      <xsd:attribute name="TYPE" type="xsd:string"/>
      <xsd:attribute ref="jp2:TIMESTAMP"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Figure M-51 Schema of the Email address type

TYPE: This element contains the type of the email address.

M.7.1.12 Web address type

This type specifies a web page address. This type may contain the sub-elements listed below.

```
<xsd:complexType name="tWeb">
  <xsd:complexContent>
    <xsd:extension base="jp2:tLangString">
      <xsd:attribute name="TYPE" type="xsd:string"/>
      <xsd:attribute ref="jp2:TIMESTAMP"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Figure M-52 Schema of the Web address type

TYPE: This element contains the type of the web page.

M.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.

```
<xsd:complexType name="tPerson">
  <xsd:sequence>
    <xsd:element name="NAME_TITLE" type="jp2: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 base="xsd:string" derivedBy="extension">
              <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                  <xsd:attribute name="TYPE">
                    <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Prefix"/>
                        <xsd:enumeration value="Given" use="default" 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="jp2: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 ref="ORGANIZATION"/>
      <xsd:element name="ORG_REF" type="xsd:string"/>
    </xsd:choice>
    <xsd:element name="ADDRESS" type="jp2:tAddress" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="PHONE" type="jp2:tPhone" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="EMAIL" type="jp2:tEmail" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="WEB" type="jp2:tWeb" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="BIRTH_DATE" type="xsd:date" minOccurs="0"/>
    <xsd:element name="AGE" type="xsd:timeDuration" minOccurs="0"/>
    <xsd:element ref="jp2:COMMENT" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute name="ID" type="xsd:string"/>
  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-53 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 M-19. Multiple values shall not be specified within a single type filed.

Table M-19 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 (Annex M.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 (Annex M.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 (Annex M.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 (Annex M.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 (Annex M.7.3.1) for more information on this element.

ID: This element specifies the unique identifier for the person.

M.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.

```
<xsd:complexType name="tOrganization">
  <xsd:sequence>
    <xsd:element name="ORG_NAME" type="jp2:tLangString" minOccurs="0"/>
    <xsd:element name="ADDRESS" type="jp2:tAddress" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="PHONE" type="jp2:tPhone" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="EMAIL" type="jp2:tEmail" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="WEB" type="jp2:tWeb" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="jp2:COMMENT" minOccurs="0"/>
    <xsd:element name="LOGO_FILE" type="xsd:uriReference" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="LOGO_FORMAT" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="MIME_TYPE" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>

  <xsd:attribute name="ID" type="xsd:string"/>
  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-54 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 (Annex M.7.1.9) for the format of this element.

PHONE: Phone number. This element specifies phone number information. See Phone number type (Annex M.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 (Annex M.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 (Annex M.7.1.12) for the format of this element.

COMMENT: This element specifies user- and/or application-defined information beyond the scope of other properties in the organization type. See Comment element (Annex M.7.3.1) for more information on this element.

LOGO_FILE: This field specifies a reference to a logo file of the organization.

LOGO_FILE_FORMAT: This field specifies the name of the logo file format. For example, EPS, JP2 and TIFF.

MIME_TYPE: This field specifies the Internet media type of the logo file.

ID: This element specifies the unique identifier for the organization.

M.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.

```
<xsd:complexType name="tLocation">
  <xsd:sequence>
    <xsd:element ref="jp2:COORD_LOC" minOccurs="0"/>
    <xsd:element name="ADDRESS" type="jp2:tAddress" minOccurs="0"/>
    <xsd:element ref="jp2:GPS" minOccurs="0"/>
    <xsd:element ref="jp2:COMMENT" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-55 Schema of the Location type

COORD_LOC: Coordinate location. This element specifies the exact longitude, latitude and altitude of an object. See Coordinate location (Annex M.7.1.15.1) for the format of this element.

ADDRESS: This element specifies the location of an object using an address. See Address type (Annex M.7.1.9) for the format of this element.

GPS: Global Positioning System. This element specifies location information received from a GPS receiver. See Raw GPS Information (Annex M.7.1.15.2) for the format of this element.

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 (Annex M.7.3.1) for the format of this element.

M.7.1.15.1Coordinate 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 must 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.

```
<xsd:element name="COORD_LOC">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="LONGITUDE" type="jp2:tDegree" minOccurs="0"/>
      <xsd:element name="LATITUDE" type="jp2:tHalfDegree" minOccurs="0"/>
      <xsd:element name="ALTITUDE" type="xsd:double" minOccurs="0"/>
    </xsd:sequence>

  <xsd:attribute ref="jp2:TIMESTAMP"/>
  </xsd:complexType>
</xsd:element>
```

Figure M-56 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.

M.7.1.15.2Raw GPS Information

The information in these elements are 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 must be filled in.

This element may contain the sub-elements listed below.

```
<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="xsd: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="xsd:tNonNegativeDouble" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GPS_ALTITUDE" type="jp2:tNonNegativeDouble" minOccurs="0"/>
      <xsd:element name="GPS_TIME" type="xsd:timeInstant" 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="jp2:tNonNegativeDouble" minOccurs="0"/>
```

Figure M-57 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="jp2: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="jp2: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="jp2: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="xsd: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="xsd: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="jp2:tNonNegativeDouble" minOccurs="0"/>
```

Figure M-58 Schema of the Raw GPS Information element (cont.)

```
      <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="jp2:tNonNegativeDouble" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure M-59 Schema of the Raw GPS Information element (continued)

GPS_LAT_REF:GPS Latitude Reference. This element specifies whether the GPS Latitude is North or South. Table M-20 lists legal values of this element.

Table M-20 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 M-21 lists legal values of this element.

Table M-21 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 M-22 lists legal values of this element.

Table M-22 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 M-23 lists legal values of this element.

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.

Table M-23 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_STATUS:GPS Status. This element contains information on the GPS receiver at time of image capture. Table M-24 lists legal values of this element.

Table M-24 GPS Status values

Value	Meaning
A	Measurement is 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 M-25 lists legal values of this element.

Table M-25 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 M-26 lists legal values of this element.

Table M-26 GPS Speed reference unit values

Value	Meaning
K	Kilometers 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 M-27 lists legal values of this element.

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 M-27 lists legal values of this element.

Table M-27 Direction reference values

Value	Meaning
T	True north
M	Magnetic north

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 M-20 lists legal values of this element.

GPS_DEST_LATITUDE:GPS Destination Latitude. This element contains the destination latitude of the GPS receiver. Table M-21 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 M-22 lists legal values of this element.

GPS_DEST_LONGITUDE:GPS Destination Longitude. This element contains the destination longitude of the GPS receiver. Table M-23 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 M-27 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 M-28 lists legal values of this element.

Table M-28 GPS Destination distance reference unit values

Value	Meaning
K	Kilometers 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.

M.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.

```
<xsd:complexType name="tDirection">
  <xsd:sequence>
    <xsd:element name="YAW" type="jp2:tDegree" minOccurs="0"/>
    <xsd:element name="PITCH" type="jp2:tHalfDegree" minOccurs="0"/>
    <xsd:element name="ROLL" type="jp2:tDegree" minOccurs="0"/>
    <xsd:element ref="jp2:COMMENT" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute ref="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-60 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 (Annex M.7.3.1) for more information on this element.

M.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 | IS 15444-1. See Figure B-1 in ITU-T T.800 | IS 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 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 (Annex M.7.1.15) for the difference between the Position and Location types.

```
<xsd:complexType name="tPosition">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element name="POINT" type="jp2:tPoint"/>
      <xsd:element name="RECT" type="jp2:tRect"/>
      <xsd:sequence>
        <xsd:element name="RECT" type="jp2:tRect"/>
        <xsd:element name="REGION" type="jp2:tRegion"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:element ref="jp2:COMMENT" minOccurs="0"/>
  </xsd:sequence>

  <xsd:attribute ref="jp2:TIMESTAMP"/>
</xsd:complexType>
```

Figure M-61 Schema of the Position type

POINT: Single point. This element specifies a single point in the coordination system. See Point type (Annex M.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 (Annex M.7.1.19) for more information of this element.

REGION: Arbitrary region. This element specifies an arbitrary region. See Region type (Annex M.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 (Annex M.7.3.1) for more information on this element.

M.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.

```
<xsd:complexType name="tPoint">
  <xsd:sequence>
    <xsd:element name="X" type="jp2:tNonNegativeDouble"/>
    <xsd:element name="Y" type="jp2:tNonNegativeDouble"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure M-62 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.

M.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 (Annex M.7.1.18) for the base format of this type. Additionally, this type shall contain the sub-elements listed below.

```
<xsd:complexType name="tRect">
  <xsd:complexContent>
    <xsd:extension  base="jp2:tPoint">
      <xsd:sequence>
        <xsd:element name="WIDTH" type="jp2:tNonNegativeDouble"/>
        <xsd:element name="HEIGHT" type="jp2:tNonNegativeDouble"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Figure M-63 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).

M.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 must 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.

```
<xsd:complexType name="tRegion">
  <xsd:sequence>
    <xsd:element name="POINT" type="jp2:tPoint"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="POINT" type="jp2:tPoint"/>
      <xsd:element name="SPLINE">
        <xsd:complexType>
          <xsd:element name="X1" type="jp2:tNonNegativeDouble"/>
          <xsd:element name="Y1" type="jp2:tNonNegativeDouble"/>
          <xsd:element name="X2" type="jp2:tNonNegativeDouble"/>
          <xsd:element name="Y2" type="jp2:tNonNegativeDouble"/>
          <xsd:element name="X"  type="jp2:tNonNegativeDouble"/>
          <xsd:element name="Y"  type="jp2:tNonNegativeDouble"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Figure M-64 Schema of the Region type

POINT: Start Point: This is the starting point of the spline in the coordinate system. See Point type (Annex M.7.1.18) for the format of this field.

POINT: This element specifies a line starting at the end of the previous spline and ending at the new point. See Point type (Annex M.7.1.18) for the format of this field.

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.

M.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.

```
<xsd:complexType name="tProductDetails">
  <xsd:sequence>
    <xsd:element name="MANUFACTURER" type="jp2: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="jp2:TIMESTAMP"/>
  <xsd:attribute ref="xml:lang"/>
</xsd:complexType>
```

Figure M-65 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 (Annex M.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.

M.7.2 Defined attributes

M.7.2.1 Language attribute

The attribute is formatted according to RFC 1766. 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 a 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.

```
<xsd:attribute name="xml:lang" type="xsd:language"/>
```

Figure M-66 Schema of the Language attribute

xml:lang: This element contains a string values that is RFC 1766 compliant. The syntax of this element must be match the Language Identification format of XML (Rec-xml-19980210).

M.7.2.2 Timestamp attribute

When a metadata element contains a Timestamp attribute, it specifies the time that the metadata was generated. Where a 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.

```
<xsd:attribute name="TIMESTAMP" type="xsd:timeInstant"/>
```

Figure M-67 Schema of the Timestamp attribute

TIMESTAMP: This element contains a string that is ISO 8601 compliant.

M.7.3 Defined elements

M.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.

```
<xsd:element name="COMMENT">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jp2:tLangString">
        <xsd:attribute ref="jp2:TIMESTAMP"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Figure M-68 Schema of the Comment element

M.8 JPX extended metadata document type definition

```
<!-- ================================================================== -->
<!-- Fundamental Type and Field Definitions                             -->
<!-- ================================================================== -->

<!--The physical location of the human readable description of the schema or DTD is:
http://www.jpeg.org/FCD15444-2.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*,
                                          COMMENT?, LOGO_FILE?, LOGO_FORMAT?, MIME_TYPE?)">

<!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 field -->

<!ELEMENT COMMENT                           (#PCDATA)>
<!ATTLIST COMMENT                           %att-lang-ts;>


<!-- ====================================================================== -->
<!-- Image Creation Metadata                                                -->
<!-- ====================================================================== -->

<!ELEMENT IMAGE_CREATION                    (GENERAL_CREATION_INFO?,
                                             CAMERA_CAPTURE?,
                                             SCANNER_CAPTURE?,
                                             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?,
                                          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                         (SENSOR_TECHNOLOGY?,
                                          FOCAL_PLANE_RES?,
                                          SPECTRAL_SENSITIVITY?,
                                          ISO_SATURATION?, ISO_NOISE?,
                                          SPATIAL_FREQ_RESPONSE?,
                                          CFA_PATTERN?, OECF?, MIN_F_NUMBER?)>
<!ATTLIST DEVICE                         %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;>

<!-- 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 | PRESON | 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;>

<!-- 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             (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 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)>


<!-- ========================================================================= -->
<!-- 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;>
```

Annex N

Examples and guidelines, extensions

N.1 Arbitrary decomposition examples

Figure N-1, Figure N-2 and Figure N-3 show example wavelet decompositions achieveable with this Recommendation | International Standard along with the appropriate syntax strings $d_\theta()$, $d_R()$ and $d_S()$ (see Annex F). The first of these examples represents the so called SPACL (an acronym for the Signal Processing and Coding Lab at the University of Arizona) decomposition. The decomposition shown in Figure N-2 represents the Packet decomposition which is quite effective for compression of synthetic aperture radar imagery. The final decomposition in Figure N-3 is a more complicated decomposition which was developed by the Federal Bureau of Investigation (FBI) for compression of fingerprint images. To alleviate crowding, most of the subbands in the three lowest resolutions in this figure are labeled with superscript indices which indicate the actual subband labels via Table N-1.

<table>
<tr><td>a_{4LL}</td><td>a_{4HL}</td><td rowspan="2">a_{3HL}</td><td rowspan="2">a_{2HL}</td><td rowspan="2">a_{1HL:LL}</td><td rowspan="2">a_{1HL:HL}</td></tr>
<tr><td>a_{4LH}</td><td>a_{4HH}</td></tr>
<tr><td colspan="2">a_{3LH}</td><td>a_{3HH}</td></tr>
<tr><td colspan="3">a_{2LH}</td><td>a_{2HH}</td><td>a_{1HL:LH}</td><td>a_{1HL:HH}</td></tr>
<tr><td colspan="3">a_{1LH:LL}</td><td>a_{1LH:HL}</td><td>a_{1HH:LL}</td><td>a_{1HH:HL}</td></tr>
<tr><td colspan="3">a_{1LH:LH}</td><td>a_{1LH:HH}</td><td>a_{1HH:LH}</td><td>a_{1HH:HH}</td></tr>
</table>

Figure N-1 SPACL decomposition: N_L=4; I_θ=2, d_θ()=21; I_R=0; I_S=0.

a_{4LL}	a_{4HL}	a_{3HL}	$a_{2HL:LL}$	$a_{2HL:HL}$	$a_{1HL:LL:LL}$	$a_{1HL:LL:HL}$	$a_{1HL:HL:LL}$	$a_{1HL:HL:HL}$
a_{4LH}	a_{4HH}							
a_{3LH}		a_{3HH}	$a_{2HL:LH}$	$a_{2HL:HH}$	$a_{1HL:LL:LH}$	$a_{1HL:LL:HH}$	$a_{1HL:HL:LH}$	$a_{1HL:HL:HH}$
$a_{2LH:LL}$		$a_{2LH:HL}$	$a_{2HH:LL}$	$a_{2HH:HL}$	$a_{1HL:LH:LL}$	$a_{1HL:LH:HL}$	$a_{1HL:HH:LL}$	$a_{1HL:HH:HL}$
$a_{2LH:LH}$		$a_{2LH:HH}$	$a_{2HH:LH}$	$a_{2HH:HH}$	$a_{1HL:LH:LH}$	$a_{1HL:LH:HH}$	$a_{1HL:HH:LH}$	$a_{1HL:HH:HH}$
$a_{1LH:LL:LL}$		$a_{1LH:LL:HL}$	$a_{1LH:HL:LL}$	$a_{1LH:HL:HL}$	$a_{1HH:LL:LL}$	$a_{1HH:LL:HL}$	$a_{1HH:HL:LL}$	$a_{1HH:HL:HL}$
$a_{1LH:LL:LH}$		$a_{1LH:LL:HH}$	$a_{1LH:HL:LH}$	$a_{1LH:HL:HH}$	$a_{1HH:LL:LH}$	$a_{1HH:LL:HH}$	$a_{1HH:HL:LH}$	$a_{1HH:HL:HH}$
$a_{1LH:LH:LL}$		$a_{1LH:LH:HL}$	$a_{1LH:HH:LL}$	$a_{1LH:HH:HL}$	$a_{1HH:LH:LL}$	$a_{1HH:LH:HL}$	$a_{1HH:HH:LL}$	$a_{1HH:HH:HL}$
$a_{1LH:LH:LH}$		$a_{1LH:LH:HH}$	$a_{1LH:HH:LH}$	$a_{1LH:HH:HH}$	$a_{1HH:LH:LH}$	$a_{1HH:LH:HH}$	$a_{1HH:HH:LH}$	$a_{1HH:HH:HH}$

Figure N-2 Packet decomposition: N_L=4; I_θ=3, d_θ()=321; I_R=0; I_S=0.

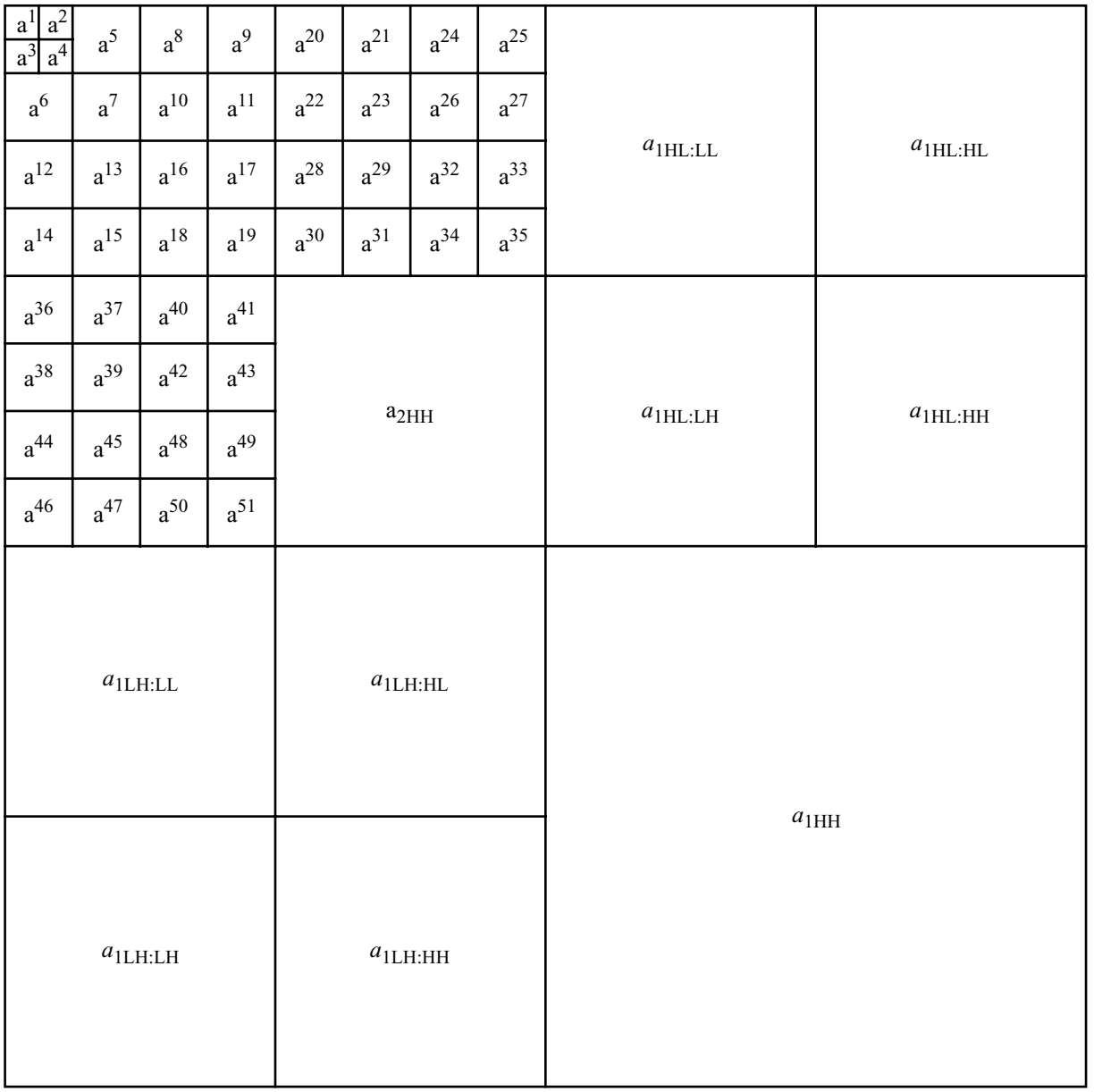

Figure N-3 FBI decomposition: N_L=5; I_θ=4, d_θ()=2321; I_R=0; d_R()=0; I_S=21, d_S()=10111011111111111111.

Table N-1 Subband labels for Figure N-3.

Superscript Label	Subband Label		Superscript Labe	Subband Label		Superscript Labe	Subband Label
a^1	a_{5LL}		a^{18}	$a_{3HH:LH}$		a^{35}	$a_{2HL:HH:HH}$
a^2	a_{5HL}		a^{19}	$a_{3HH:HH}$		a^{36}	$a_{2LH:LL:LL}$

Table N-1 Subband labels for Figure N-3.

Superscript Label	Subband Label		Superscript Labe	Subband Label		Superscript Labe	Subband Label
a^{3}	a_{5LH}		a^{20}	$a_{2HL:LL:LL}$		a^{37}	$a_{2LH:LL:HL}$
a^{4}	a_{5HH}		a^{21}	$a_{2HL:LL:HL}$		a^{38}	$a_{2LH:LL:LH}$
a^{5}	a_{4HL}		a^{22}	$a_{2HL:LL:LH}$		a^{39}	$a_{2LH:LL:HH}$
a^{6}	a_{4LH}		a^{23}	$a_{2HL:LL:HH}$		a^{40}	$a_{2LH:HL:LL}$
a^{7}	a_{4HH}		a^{24}	$a_{2HL:HL:LL}$		a^{41}	$a_{2LH:HL:HL}$
a^{8}	$a_{3HL:LL}$		a^{25}	$a_{2HL:HL:HL}$		a^{42}	$a_{2LH:HL:LH}$
a^{9}	$a_{3HL:HL}$		a^{26}	$a_{2HL:HL:LH}$		a^{43}	$a_{2LH:HL:HH}$
a^{10}	$a_{3HL:LH}$		a^{27}	$a_{2HL:HL:HH}$		a^{44}	$a_{2LH:LH:LL}$
a^{11}	$a_{3HL:HH}$		a^{28}	$a_{2HL:LH:LL}$		a^{45}	$a_{2LH:LH:HL}$
a^{12}	$a_{3LH:LL}$		a^{29}	$a_{2HL:LH:HL}$		a^{46}	$a_{2LH:LH:LH}$
a^{13}	$a_{3LH:HL}$		a^{30}	$a_{2HL:LH:LH}$		a^{47}	$a_{2LH:LH:HH}$
a^{14}	$a_{3LH:LH}$		a^{31}	$a_{2HL:LH:HH}$		a^{48}	$a_{2LH:HH:LL}$
a^{15}	$a_{3LH:HH}$		a^{32}	$a_{2HL:HH:LL}$		a^{49}	$a_{2LH:HH:HL}$
a^{16}	$a_{3HH:LL}$		a^{33}	$a_{2HL:HH:HL}$		a^{50}	$a_{2LH:HH:LH}$
a^{17}	$a_{3HH:HL}$		a^{34}	$a_{2HL:HH:LH}$		a^{51}	$a_{2LH:HH:HH}$

N.2 Multiple component transform examples

This specification defines the procedure for pre- and post-processing of multiple component imagery to reduce the redundant information between the components, thereby allowing a standard spatial compression technique to more efficiently compress the multiple components. If the multiple component data does not have correlation between components then this technique will not increase the compression efficiency. Multiple component data that commonly has correlation includes colour images, multispectral and hyperspectral. Once correlation has been identified in the multiple component data, several techniques can be used to decorrelate the data. Each of these techniques have unique advantages and disadvantages. There are also several trade-off parameters that can be optimized for a given system. The techniques run from simple linear prediction to the optimal decorrelation technique the KL transform. The simple linear prediction has the advantages of computational and memory efficiency but does not achieve the compression of the full KL transform. While the KL transform is the most efficient method of decorrelation it is computationally and memory intense. Several other technique can be used within the this structure. Other transforms that do not decorrelate but transform the data into a space where the importance of the data is known. For example, colour transforms are used to transform from RGB colour to IHS space. In this space the intensity components are known to be more important to the human eye than the hue and saturation components. For the decorrelation techniques that compute across all components as the number of components increase the complexity increases exponentially. Also, sometimes not all of the components are correlated with each other. Therefor; it is sometimes useful to segment the components into correlated blocks.

N.2.1 Linear prediction using the method of least squares

Linear Prediction is a technique based on linear regression. It uses the means of each component, the variance of the prediction component and covariance between the prediction component and the predicted component. The following are the simple equations for least squares method of Linear Prediction.

N.2.2 LGram-Schmidt vector orthogonalization

Gram-Shmidt is an extension of the linear prediction technique into multiple components. This technique decorrelates the data by projecting the previous components onto the predicted component.

N.2.3 Karhunen-Loeve Transform

The KL transform is the most complex but also gives the best compaction of data. It uses the variance/covariance matrix to produce the optimum compacted decorrelated components.

N.2.4 Colour Transform

The most common method is to use a decorrelation technique (e.g., Karhunen-Loeve) to decorrelate the multiple correlated components, then compress these components with a spatial compression scheme (e.g., JPEG). There are several different component decorrelation techniques and multiple spatial compression techniques which can be used. It is the intent of this proposal to standardize the decorrelation syntax within the framework of the current standard spatial compression (JPEG). Future (JPEG 2000) spatial compression standards may also adopt this within the new standard. Standardizing only the syntax of the decorrelation technique allows the use of multiple decorrelation techniques (including proprietary ones) on the compression side while only requiring the information to back transform the component on the decompression side. This is similar to the motion compensation of MPEG where the compressor does not need to include the motion prediction/compensation technique, only the information required to decompress that data. In the development of this compression algorithm, we reviewed multiple decorrelation techniques and the requirements of each. It was desirable to have as much flexibility without significant increase in overhead information, therefore, several decorrelation techniques were not included because of complexity, compatibility or capability.

CT is a decorrelating transformation which is applied to a group of components of an image.

Annex O

Bibliography

O.1 General

[1] Special Session on JPEG 2000, Proc. of the Int. Conf. on Image Processing (ICIP-2000), Vancouver, Canada, 13-16 September 2000.

[2] C. Christopoulos, A. Skodras, The JPEG 2000, JPEG2000 Tutorial presented in IEEE International Conference on Image Processing (ICIP 99), October 25-28, 1999, Kobe, Japan (available at http://etro.vub.ac.be/~chchrist/jpeg2000_contributions.htm)

O.2 Wavelet transform

[3] Wavelet Scalar Quantizer (WSQ) Gray-scale Fingerprint Image Compression Specifications, version. 2.0, Document #IAFIS-IC-0110v2, United States Federal Bureau of Investigation., 16 February 1993.

[4] C. M. Brislawn, "Classification of nonexpansive symmetric extension transforms for multirate filter banks," *Appl. Comput. Harmonic. Analysis*, vol. 3, pp. 337-57, 1996.

[5] C. Chrysafis, A. Ortega, An algorithm for low memory wavelet image compression, Proc. IEEE Int. Conference on Image Processing (ICIP), 24-28 October 1999, Kobe, Japan.

[6] A.V. Oppenheim, R.W. Schafer, Discrete Time Signal Processing, Prentice-Hall, 1989.

O.3 Quantization and Entropy coding

[7] M.W. Marcellin and T. R. Fischer, Trellis Coded Quantization of Memoryless and Gauss-Markov Sources, IEEE Trans Commun., January 1990.

[8] T. R. Fischer and M. Wang, Entropy Constrained Trellis Coded Quantization, IEEE Trans Inform. Th., March 1992.

[9] J.H. Kasner, M.W. Marcellin, and B.R. Hunt, Universal trellis coded quantization, submitted to IEEE Trans. Image Proc., available as preprint at www-spacl.ece.arizona.edu.

[10] A. Bilgin, P. Sementilli and M. W. Marcellin, Progressive image coding using trellis coded quantization, to appear in IEEE Trans. Image Processing, preprint available at www-spacl.ece.arizona.edu.

[11] A. Zandi, J. D. Allen, E. L. Scwhartz, M. Boliek, CREW: Compression with reversible embedded wavelets, Proc. of Data Compression Conference, Snowbird, UT, pp. 212-21, March 1995.

O.4 Region of Interest coding and shape coding

[12] C. Christopoulos, J. Askelof and M. Larsson, Efficient region of interest encoding techniques in the upcoming JPEG2000 still image coding standard, invited paper to IEEE Int. Conference on Image Processing (ICIP 2000), Special Session on JPEG2000, September 10-13, 2000, Vancouver, Canada.

[13] D. Nister, C. Christopoulos, Lossless region of interest coding, Signal Processing, Vol. 78, No. 1, pp. 1-17, October 1999.

[14] Christopoulos C.A, and Skodras A.N, New schemes for progressive transmission of digital images, IEEE Transactions on Consumer Electronics, Vol. 43, No. 4, pp. 1028-1033, November 1997.

[15] Christopoulos C.A, Philips W., Skodras A.N. and Cornelis J, Segmented Image Coding: techniques and experimental results, Signal Processing: Image Communication, Vol. 11, No. 1, pp. 63-80, 1997

O.5 Visual frequency weighting

[16] P. Jones, S. Daly, R. Gaborski and M. Rabbani, Comparative study of wavelet and DCT decompositions with equivalent quantization and encoding strategies for medical images, Proceedings of Conference on Medical Imaging, SPIE vol. 2431, pp. 571-582, February 1995.

[17] S. Daly, W. Zeng, J. Li, and S. Lei, Visual masking in wavelet compression for JPEG2000, in Proc. IS&T/SPIE Conf. Image and Video Communications and Processing, vol. 3974, pp. 66-80, January 2000.

[18] W. Zeng, S. Daly and S. Lei, Point-wise extended visual masking for JPEG2000 image compression, IEEE Inter. Conf. Image Proc., Vancouver, Canada, September 2000.

O.6 Post-processing

[19] M. Shen and C.-C. Jay Kuo, Artifact Reduction in Low Bit Rate Wavelet Coding with Robust Non-linear Filtering, Proc. of IEEE 1998 Workshop on Multimedia Signal Processing (MMSP-98), Los Angeles, California, 7-9 December 1998.

Index

Annex P

Patent Statement

There is the possibility that, for some of the processes specified in this Recommendation | International Standard, conformance or compliance may require use of an invention covered by patent rights.

By publication of this Recommendation | International Standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. Information regarding such patents can be obtained from the any organizations. The table summarizes the formal patent and intellectual property rights statements that have been received.

Table P-1 Received intellectual property rights statements

Number	Company
1	Algo Vision
2	Canon Incorporated
3	Digital Accelerator Corporation
4	Digital Imaging Group (DIG)
5	Telefonaktiebolaget L M Ericsson
6	Hewlett Packard Company
7	International Business Machines, Inc.
8	LizardTech, Incorporated
9	LuraTech
10	MITRE Incorporated
11	Mitsubishi Electric Corporation
12	Motorola Corporation
13	PrimaComp Incorporated
14	Rensselaer Polytechnic Institute (RPI)
15	Ricoh Company, Limited
16	SAIC
17	Sarnoff Corporation
18	Sharp Corporation
19	Sony Corporation
20	TeraLogic Incorporated

Table P-1 Received intellectual property rights statements

Number	Company
21	University of Arizona
21	University of New South Wales
22	Washington State University