ISO/IEC JTC 1/SC 29

Date: 2005-03-21

ISO/IEC FCD 15444-11

ISO/IEC JTC 1/SC 29/WG 1

Secretariat:

Information technology — JPEG 2000 image coding system — Part 11: Wireless

Élément introductif — Élément central — Partie 11: Titre de la partie

Contents

Page

Foreword

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

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

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

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

ISO/IEC 15444-11 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 29, *Coding of Audio, Picture, Multimedia and Hypermedia Information*.

ISO/IEC 15444 consists of the following parts, under the general title *Information technology — JPEG 2000 image coding system*:

— *Part 1: Core coding system*

— *Part 2: Extensions*

— *Part 3: Motion JPEG 2000*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Compound image file format*

— *Part 8: Secure JPEG 2000*

— *Part 9: Interactivity tools, APIs and protocols*

— *Part 10: 3-D and floating point data*

— *Part 11: Wireless JPEG 2000*

— *Part 12: ISO base media file format*

— *Part 13: Entry level Part 1 encoder*

Information technology — JPEG 2000 image coding system — Part 11: Wireless

1 Scope

This International Standard defines, in an extensible manner, syntaxes and methods for the protection against errors that may occur during the transmission of JPEG 2000 codestreams compliant with International Standard | Recommendation ITU-T T.800 | ISO/IEC 15444-1.

In this International Standard, these are referred to as Wireless JPEG 2000, “JPWL”, and applications using JPWL are referred to as a “JPWL system.”

JPWL specifies a set of tools consisting of additional data structures to JPEG 2000 codestreams and error protection techniques, necessary for error correction and signaling. This International Standard | Recommendation includes definitions of the semantics, and suggests how these may be used.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ITU–T Rec. T.800 | ISO/IEC 15444-1:2000, *Information technology — JPEG 2000 image coding system: Part 1: Core coding system*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply. The definitions defined in ITU–T Rec. T.800 | ISO/IEC 15444–1:2000 Clause 3 apply to this International Standard.

3.1 Backward Compatible

Includes all techniques that produce a bitstream that will lead the Part-1 decoder to decode/display according to JPEG 2000 Part 4 (Rec. ITU-T T.803 | ISO/IEC 15444-4) specifications in case of error-free environment.

3.2 Backward Compatible with Extensions

Includes all techniques that produce a bitstream that will not lead the Part-1 decoder to crash in case of error-free environment. A JPWL decoder is required to correctly decode/display images.

3.3 Non Backward Compatible

Includes all techniques that produce a bitstream that may lead the JPEG 2000 Part 1 decoder to crash also in case of error-free environment. This kind of techniques is outside of the scope of this International Standard | Recommendation.

3.4 Codestream

A collection of one or more bit streams and the Main header, Tile-part headers, and the EOC required for their decoding and expansion into image data. This is the image data in a compressed form with all of the signalling needed to decode.

3.5 Bitstream

The actual sequence of bits resulting from the coding of a sequence of symbols. It does not include the markers or marker segments in the main and Tile-part headers or the EOC marker. It does include any packet headers and in stream markers and marker segments not found within the main or Tile-part headers.

3.6 Bit Error Rate (BER)

The BER is defined as the statistical expected value of the ratio between the number of erroneous bits in the received data and the size of the received data themselves.

3.7 Packet Loss Rate (PLR)

The PLR is defined as the statistical expected value of the ratio between the number of packets discarded during the transmission, and the number of packets sent during the transmission. Within this definition, it is intended that a packet is considered at transmission level, and not as a basic entity of a JPEG 2000 codestream.

3.8 Forward Error Correcting (FEC)

The FEC consists of any techniques aiming at providing error detection and/or correction capability by adding redundancy to the codestream.

3.9 Systematic codes

A systematic code is one that produces a given number of redundancy symbols in addition to the original input data symbols.

3.10 Unequal Error Protection (UEP)

UEP refers to the act of assigning different degrees of error protection to different parts of a codestream.

3.11 Data partitioning

Data partitioning is a modification of the organization of the codestream, with a separation of the compressed data in different parts.

3.12 Interleaving

Interleaving is a modification of the data ordering of a codestream.

3.13 Encoder

An embodiment of an encoding process.

3.14 Encoding process

A process that takes as its input all or part of a source image data and outputs a codestream.

3.15 Decoder

An embodiment of a decoding process, and optionally a color transformation process.

3.16 Decoding process

A process which takes as its input all or part of a codestream and outputs all or part of a reconstructed image.

3.17 Transcoder

An embodiment of a transcoding process.

3.18 Transcoding process

A process which takes as its input all or part of a codestream and outputs all or parts of it, together with the possible addition of other data.

3.19 Big endian

The bits of a value representation occur in order from most significant to least significant.

3.20 Little endian

The bits of a value representation occur in order from least significant to most significant.

3.21 Marker

A two-byte code in which the first byte is hexadecimal FF (0xFF) and the second byte is a value between 1 (0x01) and hexadecimal FE (0xFE).

3.22 Marker segment

A marker and associated (not empty) set of parameters.

3.23 Packet

A part of the bit stream comprising a packet header and the compressed image data from one layer of one precinct of one resolution level of one tile-component.

3.24 Packet header

Portion of the packet that contains signaling necessary for decoding that packet.

3.25 Pointer markers and pointer marker segments

Markers and marker segments that offer information about the location of structures in the codestream.

3.26 Precinct

A rectangular region of a transformed tile-component, within each resolution level, used for limiting the size of packets.

3.27 Precision

Number of bits allocated to a particular sample, coefficient, or other binary numerical representation.

3.28 Code-block

A rectangular grouping of coefficients from the same subband of a tile-component.

3.29 Layer

A collection of compressed image data from coding passes of one, or more, code-blocks of a tile-component. Layers have an order for encoding and decoding that must be preserved.

3.30 JPWL Registration Authority

An organization that is in charge of delivering a unique ID to reference a JPWL tool and storing the parameter list of its description.

4 Symbols (and abbreviated terms)

4.1 Abbreviations

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

ITU: International Telecommunication Union

ITU-T: International Telecommunication Union – Telecommunication Standardization Sector (formerly the CCITT)

JPEG: Joint Photographic Experts Group - The joint ISO/ITU committee responsible for developing standards for continuous-tone still picture coding. It also refers to the standards produced by this committee: ISO/IEC 10918 and their corresponding ITU Recommendations..

JPEG 2000: Joint Photographic Experts Group - The joint ISO/ITU committee responsible for developing standards for continuous-tone still picture coding. It also refers to the standards produced by this committee: ISO/IEC 15444 and their corresponding ITU Recommendations.

JPEG 2000 Part 1: Refers to Part 1 of JPEG 2000, ITU-T T.800| ISO/IEC 15444-1

JPEG 2000 Part 11: Refers to this International Standard | Recommendation

JPWL: Refers to this International Standard | Recommendation

RA: Refers to Registration Authority

4.2 Symbols

For the purposes of this Recommendation | International Standard, the following symbols apply.

0x----: Denotes a hexadecimal number.

\nnn: A three-digit number preceded by a backslash indicates the value of a single byte within a character string, where the three digits specify the octal value of that byte.

ε_b**:** Exponent of the error sensitivity value defined in ESD.

μ_b**:** Mantissa of the error sensitivity value defined in ESD.

COC: Coding style component marker

COD: Coding style default marker

COM: Comment marker

CRG: Component registration marker

EPH: End of packet header marker

EOC: End of codestream marker

PLM: Packet length, Main header marker

PLT: Packet length, Tile-part header marker

POC: Progression order change marker

PPM: Packed packet headers, Main header marker

PPT: Packed packet headers, Tile-part header marker

QCC: Quantization component marker

QCD: Quantization default marker

RGN: Region of interest marker

SIZ: Image and tile size marker

SOC: Start of codestream marker

SOP: Start of packet marker

SOD: Start of data marker

SOT: Start of Tile-part marker

TLM: Tile-part lengths marker

EPB: Error protection block marker

ESD: Error sensitivity descriptor marker

RED: Residual error descriptor marker

EPC: Error protection capacity marker

FEC : forward error correction

UEP: unequal error protection

RA: registration authority

CRC: Cyclic Redundancy Check

RS: Reed Solomon

BCH: Bose-Chaudhuri-Hocquenghem

5 JPWL General Description

This specification defines a set of tools and methods to achieve the efficient transmission of JPEG 2000 Part 1 imagery over an error-prone transmission/storage environment. The main target of this International Standard is wireless applications, although the same tools can be employed in other types of applications, which are prone to errors.

Wireless networks are characterized by the frequent occurrence of transmission errors, henceforth putting strong constraints on the transmission of digital images. Since JPEG 2000 provides high compression efficiency, it is a good candidate for wireless multimedia applications. Moreover, due to its high scalability, JPEG 2000 enables a wide range of quality of service strategies for network operators. However, to be suitable for wireless multimedia applications, JPEG 2000 has to be robust to transmission errors.

JPEG 2000 Part 1 defines error resilience tools to improve performances over noisy channels. However, these tools can only detect occurrence of errors in the bitstream, conceal the erroneous data, and resynchronize the decoder. More specifically, they do not correct transmission errors. Furthermore, these tools do not apply to the Main and Tile-part headers which are the most important parts of the codestream. For these reasons, they are not sufficient in the context of wireless transmissions.

For the purpose of efficient transmission over error-prone transmission/storage environments, this specification defines additional mechanisms for error protection and correction. These mechanisms extend the elements in the core coding system described in ITU-T T.800 | ISO/IEC 15444-1. These extensions are backward compatible or backward compatible with extensions, as specified in Section 3.

This specification is not linked to a specific network or transport protocol, but provides a general solution for the robust transmission of JPEG 2000 imagery over error-prone channels and networks. JPWL would normally act at the application level. However, if appropriate, the JPWL tools can be used for direct transmission of images on the channel physical layer.

5.1 JPWL system description

The main functionalities of the JPWL system are to protect the codestream against transmission errors, to describe the degree of sensitivity to transmission errors of different parts of the codestream, and to describe the locations of residual errors in the codestream.

The JPWL system can either be applied to an input source image or to a Part 1 codestream, as is illustrated in Figure 1 and Figure 2 respectively. In Figure 1, at the transmission side, a JPWL encoder consists of three modules running concurrently: a JPEG 2000 Part 1 encoder compressing the input image, a generator of the error sensitivity description, and a processor applying the error protection tool. The result is a JPEG 2000 Part 11 codestream robust to transmission errors. At the receiving side, a JPWL decoder is also composed of three modules: a processor to correct errors, a generator of the residual errors description and a JPEG 2000 Part 1 decoder. Alternatively, in Figure 2 at the transmission side a JPWL transcoder processes a JPEG 2000 Part 1 codestream, generating the error sensitivity description and applying error protection tools. At the receiving side, a JPWL transcoder corrects the transmission errors and generate the residual errors description, producing a Part 1 codestream which can be sent to a Part 1 decoder, along with residual errors information.

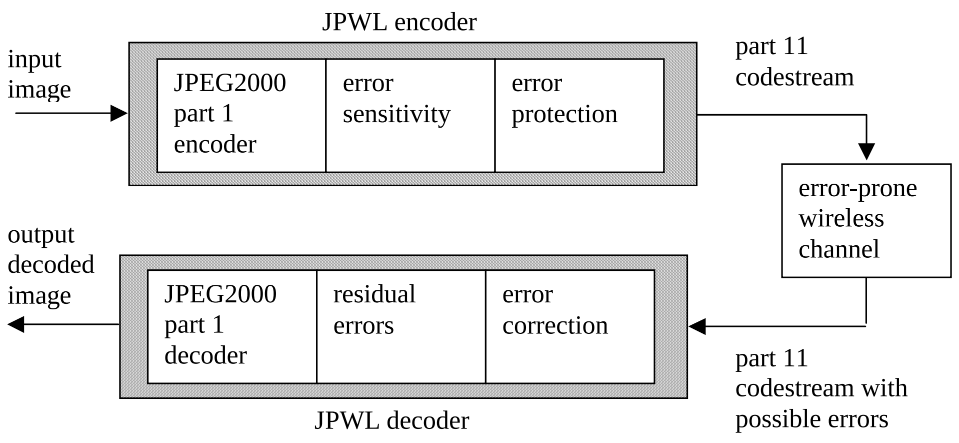

Figure 1: JPWL system description: JPWL encoder and decoder.

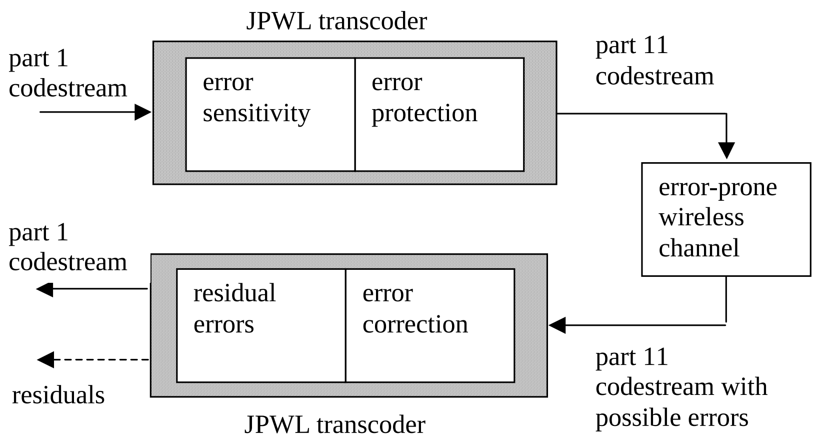

Figure 2: JPWL system description: JPWL transcoder.

Other similar configurations are also possible as illustrated in Figure 3 and Figure 4. Whereas in Figure 1 and Figure 2 the generation of the error sensitivity description and the application of the error protection tool are concurrent, in Figure 3 and Figure 4 the two operations are performed successively. More precisely, in a first step a JPWL encoder/transcoder produces a JPEG 2000 Part 11 codestream containing error sensitivity information. In a second step, a JPWL transcoder uses this information to optimize the error protection tool, generating a JPEG 2000 Part 11 codestream robust to transmission errors.

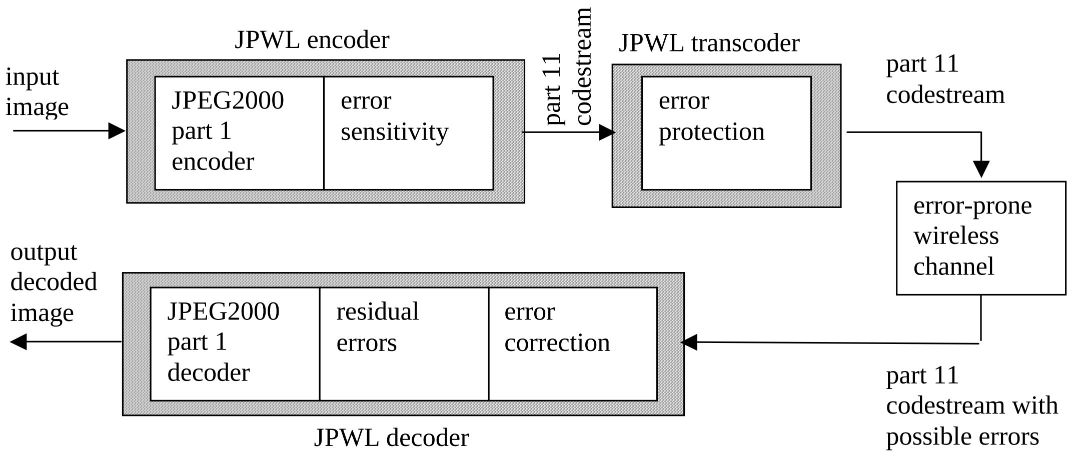

Figure 3: JPWL system description: another configuration.

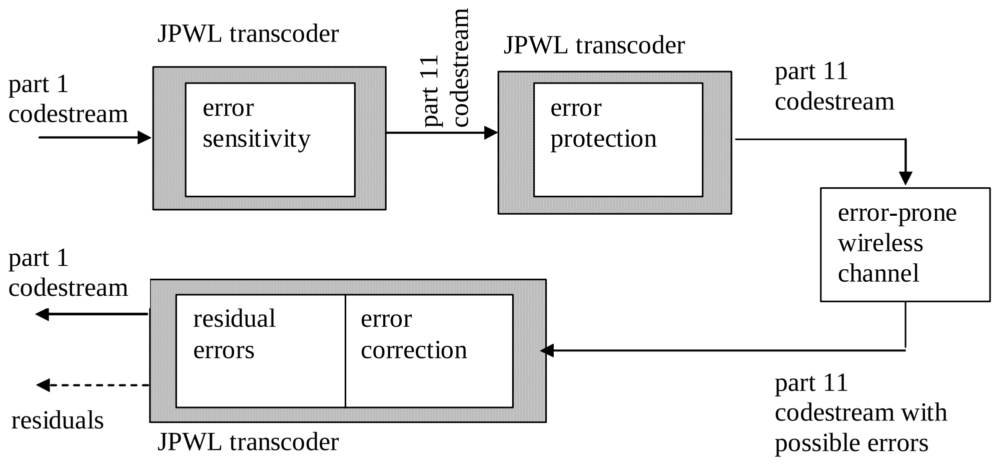

Figure 4: JPWL system description: another configuration.

The error protection process modifies the codestream to make it more resilient to errors, e.g. by adding redundancy or by partitioning and interleaving the data. The error correction process detects the occurrence of errors and corrects them whenever possible. Techniques to protect the codestream include Forward Error Correcting (FEC) codes, data partitioning and interleaving, robust entropy coding, and unequal error protection.

The error sensitivity descriptor describes the degree of sensitivity of different parts of the codestream to transmission errors. This information is typically generated when the image is encoded using a JPEG 2000 Part 1 encoder (e.g. Figure 1 and Figure 3), but it can also be directly derived from a Part 1 codestream (e.g. Figure 2 and Figure 4). This information can subsequently be used when protecting the image. More specifically, sensitive parts of the codestream can be more strongly protected than less sensitive parts (unequal error protection).

The residual errors descriptor specifies the locations of residual errors in the codestream. The residual errors are the errors which cannot be corrected by the error protection tool. This information is typically generated during the error correction process. This information can subsequently be used in the JPEG 2000 Part 1 decoder to prevent decoding of corrupted parts of the stream.

The above figures, describing the JPWL system, are examples and different configurations are possible.

JPWL system has provision for future techniques, in addition to those described in this International Standard | Recommendation. The process of adding new techniques is managed by the Registration Authority as described in Annex K.

6 JPWL Normative parts

An encoding process converts source image data to compressed image data. All encoding processes are specified informatively.

An encoder is an embodiment of the encoding process. In order to conform to this International Standard, an encoder shall convert source image data to compressed image data that conform to the codestream syntax specified in Annex A.

A decoding process converts compressed image data to reconstructed image data. Some parts of a decoding process are normative, and namely those related to extracting information contained in the JPEG 2000 Part 11 specific marker segments, as well as those that refer to the decoding of JPEG 2000 Part 1 features. All other aspects of the decoding process, for instance the procedure that the decoder shall follow in order to cope with the possible presence of errors and the actions it shall take to minimize their effect, are not specified as part of this International Standard | Recommendation; guidelines are however specified in Annex G.

A decoder is an embodiment of the decoding process. In order to conform to this International Standard | Recommendation, a decoder shall convert all, or specific parts of, any compressed image data that conform to the codestream syntax specified in Annex A to a reconstructed image.

There is no normative or required implementation for the encoder or decoder. In some cases, the descriptions use particular implementation techniques for illustrative purposes only.

Annex A describes the syntax that defines the coded representation of compressed image data for exchange between application environments. Any compressed image data shall comply with the syntax and code assignments appropriate for the coding processes defined in this International Standard | Recommendation.

The remaining of this section outlines the normative parts of this specification and refers to the respective Annexes for detailed description:

- Codestream syntax (Annex A): Definition of the codestream syntax every JPWL codestream must conform to.

- Error protection block (Annex B): Tool to protect the image header (Main header, tile/Tile-part header) and to correct the possible presence of transmission errors using FEC codes.

- Error protection capability descriptor (Annex C): Description of the tools which have been used to protect the codestream and to correct the possible presence of transmission errors. This descriptor relies on a registration authority as to the informative error protection techniques.

- Error sensitivity descriptor (Annex D): Description of the degree of sensitivity of different parts of the codestream to transmission errors. This information is typically generated when encoding the image. It can subsequently be used to apply Unequal Error Protection (UEP).techniques which take into account the error sensitivity.

- Residual errors descriptor (Annex E): Description of the locations of residual errors in the codestream. The residual errors are the errors which cannot be corrected by the tools used to protect the image. This information is typically generated when decoding the codestream.

- Registration authority (Annex K): Specification of the Registration Authority (RA).

7 JPWL informative parts

This section outlines informative parts of this specification and refers to the respective Annexes for detailed description:

Comment [FD1]: "Informative requirements" is a contradiction. If it's informative, it is not a requirement...

- Encoding guidelines (Annex F): Guidelines for error resilient coding at the encoder side in the context of error prone environments.

- Decoding guidelines (Annex G): Guidelines for error handling behaviour at the decoder side.

- Error resilient entropy coding (Annex H): Tools to protect the codestream and to detect and correct the possible errors based on error resilient entropy coding.

- Unequal error protection (Annex I): Tools to protect differently parts of the codestream based upon the error sensitivity of the respective parts.

- Interoperability with ISO/IEC 15444 (Annex J): Guidelines for interoperability with other specifications in the JPEG 2000 family.

- Patents (Annex L): Received intellectual property rights statements that apply to this International Standard | Recommendation.

Annex A: Codestream Syntax

(This Annex forms a normative and integral part of this International Standard.)

A.1 Definitions of markers and marker segments

This International Standard | Recommendation relies on the use of marker segments to delimit and signal the characteristics of the codestream protected against errors. For backward compatibility, the JPWL markers and marker segments must be included in JPEG 2000 Part 1 codestream headers, which can be of two types only:

1. the Main header, found at the beginning of the codestream,

2. the Tile-part headers, found at the beginning of each Tile-part.

Main and Tile-part headers are collections of markers and marker segments.

As for every other standard marker defined in JPEG 2000 Part 1, each marker defined in this proposal is two bytes long, and its first byte value is 0xFF. The second byte specifies the marker use and can take any value in the range 0x01 to 0xFE, apart from those already used by the ITU-T Rec. T.81 | ISO/IEC 10918-1 and ITU-T Rec. T.84 | ISO/IEC 10918-3 (recalled in Table A.1).

A marker segment includes a marker and associated parameters, called marker parameters. By definition, the first two bytes of any marker segment immediately after the marker must correspond to an unsigned big endian integer value that denotes the length in bytes of the marker parameters (including two bytes of this length parameter but not including the two bytes of the marker itself). When the decoder finds a marker segment that is not specified in JPEG 2000 Part 1 or in this International Standard | Recommendation, it shall use the length parameter to discard the marker segment.

A.2 Marker code range defined in this document

Following the syntax used for each marker and marker segment defined in ITU-T Rec. T.81 | ISO/IEC 10918-1, this International Standard reserves some markers for signalling, as specified in Table A.1. Table A.1 recalls the various values of already existing or reserved markers.

Table A.1 – Marker definitions

Marker value range	Standard definition
0xFF00, 0xFF01, 0xFFFE, 0xFFC0 – 0xFFDF	Defined in ITU-T Rec. T.81 | ISO/IEC 10918-1
0xFFF0 – 0xFFF6	Defined in ITU-T Rec. T.84 | ISO/IEC 10918-3
0xFFF7 – 0xFFF8	Defined in ITU-T Rec. T.87 | ISO/IEC 14495-1
0xFF4F – 0xFF6F, 0xFF90 – 0xFF93	ITU-T Rec. T.800 | ISO/IEC 15444-1
0xFF66-0xFF69	Defined in this International Standard | Recommendation
0xFF30 – 0xFF3F	Reserved for definition as markers only (no marker

	segments)
	All other values reserved.

A.3 Marker and marker segment and codestream rules

Marker segments, described in this International Standard, are respecting the rules given in JPEG 2000 Part 1, at the section A.1.3.

A.4 Information in the marker segments

As standardized in JPEG 2000 Part 1, marker segments, and therefore the headers, are a multiple of 8 bits (one byte). They give elements and inform on what should be applied to the Tile-part header or a start of packet header.

All markers and marker segments in a Tile-part header or a start of packet header apply only to the tile or the packet to which it belongs.

If truncation, alteration, or editing of the codestream has been done, the impacted marker segments (like TLM/PLT or JPWL marker segments) shall be updated accordingly. Note that several JPWL marker segments contain codestream indexing information (e.g., byte ranges); this information must be updated upon insertion or cancellation of a marker segment.

Table A.2 lists the markers specified in this International Standard and Table A.3 lists the information provided by the syntax and indicates the marker segment containing that information.

Table A.2 – List of marker segments

	Name	Code	Main header [a]	Tile-part header [a]
Error Protection Block	EPB	0xFF66	Optional	Optional
Error Sensitivity Descriptor	ESD	0xFF67	Optional	Optional
Error Protection Capability	EPC	0xFF68	Required	Optional
Residual Errors Descriptor	RED	0xFF69	Optional	Optional

a. Required means the marker segment shall be in this header, optional means it may be used.

If the EPC, ESD or RED marker segments appear both in the Main header and Tile-part header, the marker present in the Tile-part header is overriding the one present in the Main header for the current Tile-part. The EPC and RED marker segments are allowed to appear at most once in one header (Main or Tile-part header). Multiple ESD in one single header are allowed.

A.5 Construction of the codestream

The construction of the codestream of this International Standard | Recommendation complies with the codestream construction defined in JPEG 2000 Part 1, at the section A.3. The EPB marker segment(s) are required to be in a specific location, as specified in Annex B of this International Standard.

Table A.3 – Information in the marker segments

Information	Marker segment
Signals the presence of JPWL protected data in the header. It includes: Set of error protection parameters used in the codestream Error protection data generated from a systematic code	EPB
Indicates the methods used in the current codestream to protect it against transmission errors. Its presence signals the fact that the codestream complies with this International Standard | Recommendation.	EPC
Describes the sensitivity to errors of the current codestream	ESD
Describes the index of the residual errors of the current codestream	RED

A.6 JPWL marker segments

A.6.1 Error Protection Block (EPB)

The EPB marker segment contains information about the error protection parameters and data used to protect the codestream against errors. The primary function of EPB is to protect the Main and Tile-part header (see Annex B). However, it can also be used to protect the bitstream (see Annex I). There can be one or more EPB marker segments in the Main header and/or Tile-part headers. The first EPB marker segment in a Main header is required to be placed immediately after the SIZ marker segment. The first EPB marker segment in a Tile-part header is required to be placed immediately after the SOT marker.

Function: The EPB marker segment contains necessary error correction data for the header where it is located. See Annex B and Annex I for more information on how to use EPB marker segments.

Usage: Main header and Tile-Part Headers. The first EPB marker segment of the codestream must be placed after the SIZ marker segment. The first EPB marker segment of a Tile-part header must be placed after the SOT marker segment.

Length: Variable depending on the parameters used to protect the headers and the length of the headers to be protected. Table A.4 describes the syntax of the EPB marker segment.

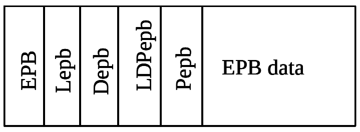

Figure A.1 – Error Protection Block description syntax.

EPB: Marker code. Table A.4 indicates the size and parameter values of the marker symbol itself and of each parameter of the marker segment.

Lepb: Length of marker segment in bytes.

Depb: EPB style (for example it defines if the current EPB is the latest in the current header) (length 8 bits)

LDPepb: length of the data to be protected by the redundant information (EPB data) carried within the current EPB (length 32 bits)

Pepb: EPB parameters defining the next error correction tool to be used for protecting the remaining data.

EPB data: Contains the data enabling the correction (typically redundancy bits)

Table A.4 – Error Protection Block parameter values

Parameter	Size (bits)	Values
EPB	16	0xFF66
Lepb	16	11 – (2^{16}-1)
Depb	8	See Table A.5
LDPepb	32	0 – (2^{31}-1)
Pepb	32	See Table A.6 Defines the next error management method to be used.
EPB data	variable	

When the EPB is included in a Main header, the SOC marker, the SIZ marker segment, the EPB marker, the Lepb, Depb, LDPepb, Pepb data are protected with a predefined RS(N1,K1) code. The redundant data needed for error correction are placed at the beginning of the EPB data.

When the EPB is included in a Tile-part header, the SOT marker, the EPB marker, the Lepb, Depb, LDPepb and Pepb data are protected with a predefined RS(N2,K2) code. The redundant data needed for error correction are placed at the beginning of the EPB data.

There can be several EPB marker segments in the Main or Tile-part headers. When an EPB is not the first one in the header, a predefined RS(N3,K3) code is used.

The predefined codes are:

Reed Solomon RS(160,64) to be used for the first EPB marker segment of the Main Header

Reed Solomon RS(80,25) to be used for the first EPB marker segment of a Tile-part Header

Reed Solomon RS(40,13) to be used for the other EPB marker segments of both the Main Header and the Tile-part header.

A.6.1.1 EPB style parameter

Table A.5 – Depb parameter values

Values (bits) MSB LSB	EPB configuration and index
x0xx xxxx	The EPB marker segment is not the latest in the current header

x1xx xxxx	The EPB marker segment is the latest in the current header
0xxx xxxx	EPBs marker segments are unpacked
1xxx xxxx	EPBs marker segments are packed
xx00 0000 – xx11 1111	EPB index values (0-63). The first EPB marker segment in an header has the index value zero. For every successive EPB in the same header this index value is incremented by one. When the maximum number is reached, the number rolls over to zero.

A.6.1.2 EPB parameters

These parameters allow to select another error correction code or other method for managing errors like a CRC, than the default error correction codes. The error management methods specified in this parameter has to be applied to the codestream data following the current EPB marker segment, for which redundant data are included within the EPB marker segments after the redundant data of the codestream data before and including the beginning of the marker segment itself.

Table A.6: Pepb parameter

Error Management method index	EPB configuration and index
0x00000000	Predefined codes: Reed Solomon RS(160,64) to be used for the first EPB marker segment of the Main Header Reed Solomon RS(80,25) to be used for the first EPB marker segment of a Tile-part Header Reed Solomon RS(40,13) to be used for the other EPB marker segments of both the Main Header and the Tile-part header.
0x10000000-0x1FFFFFFF	CRC, see Table 7
0x20000000-0x2FFFFFFF	Reed Solomon codes, see Table 8
0x30000000-0xFFFFFFFE	Use and registration managed by the JPWL Registration Authority
0xFFFFFFFF	No method to be used for the next data

Table A.7: CRC types

Pepb value	CRC type

0001 0000 0000 0000	CRC-CCITT (X25) 16bits CRC
0001 0000 0000 0001	Ethernet CRC 32bits
0001 0000 0000 0010 – 0001 1111 1111 1111	Use and registration managed by the JPWL Registration Authority

Table A.8 Reed-Solomon default codes

Pepb value	Reed Solomon code
0x20002520	RS(37,32)
0x20002620	RS(38,32)
0x20002820	RS(40,32)
0x20002B20	RS(43,32)
0x20002D20	RS(45,32)
0x20003020	RS(48,32)
0x20003320	RS(51,32)
0x20003520	RS(53,32)
0x20003820	RS(56,32)
0x20004020	RS(64,32)
0x20004B20	RS(75,32)
0x20005020	RS(80,32)
0x20005520	RS(85,32)
0x20006020	RS(96,32)
0x20007020	RS(112,32)
0x20008020	RS(128,32)
Other RS index values	Use and registration managed by the JPWL Registration Authority

A.6.2 Error Protection Capability (EPC)

The EPC marker segment indicates which JPWL normative and informative tools are used in the codestream. Namely, it indicates the presence of the ESD marker segment, the RED marker segment, and the EPB marker segment in the codestream. Furthermore, EPC signals the use of informative tools which have been previously registered with the JPWL Registration Authority (JPWL RA, see Annex K). These informative tools allows for error resilience and/or error correction, and include techniques such as error resilient entropy coding, UEP, data partitioning and interleaving. EPC may also contain parameters relative to these informative tools.

Function: The EPC marker segment signals the use of JPWL tools (ESD, RED, EPB) or informative tools in the codestream. See Annex C for more information on how to use EPC marker segment.

Usage: Mandatory in Main header, optional in Tile-part headers. No more that one EPC shall appear in each Main or Tile-part header.

Length: Variable.

The syntax of the EPC marker segment is defined in Figure A.2 . The meaning of the data fields is discussed below, and the range of possible values for each parameter is defined in Table A.9. A more detailed description of EPC is given in Annex C.

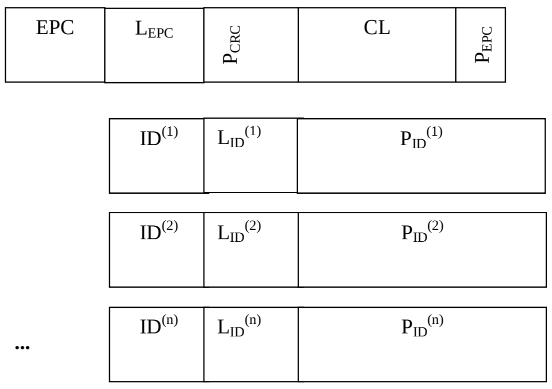

Figure A.2 – The EPC marker segment.

EPC: two bytes marker (0xFF67).

L_{EPC}: length of the data structure in bytes (two bytes).

P_{CRC}: parity check bits which verify whether the EPC marker segment is corrupted (16bits)

CL: field describing the total codestream length (four bytes).

P_{EPC}: field signaling the usage of ESD, RED, EPB or informative techniques in the codestream (8 bits).

$ID^{(i)}$: registered ID for protection technique *i* (2 bytes), optional, present only when informative technique is used.

$L_{ID}^{(i)}$: length of $P_{ID}^{(i)}$ (2 bytes), optional, present only when informative technique is used.

$P_{ID}^{(i)}$: parameters for protection technique *i* (variable), optional, present only when informative technique is used.

Table A.9 – Error Protection Capacity parameter values

Parameter	Size (in bits)	Values
EPC	16	0xFF68
L_{EPC}	16	$[9,2^{16}-1]$
Pcrc	16	Cyclic Redundancy Check of EPC marker segment, excluding the Pcrc data field. Uses the CRC-CCITT (see Annex B).
CL	32	$[0,2^{32}-1]$ Length of the codestream, expressed in bytes as an unsigned integer number, from the first byte of the SOC marker to the last byte of the EOC marker. 0 means that this information is not available.
P_{EPC}	8	See Table A.10
$ID^{(i)}$	16	$[0,2^{16}-1]$ 0 indicates the EPB technique Refer to Annex B for the use of EPB 1-15 are reserved other values are registered with the RA
$L_{ID}^{(i)}$	16	$[0,2^{16}-1]$
$P_{ID}^{(i)}$	variable	If $ID^{(i)}$=0, indicating the EPB technique, $P_{ID}^{(i)}$ is the concatenation of all Pepb present in the EPB marker segments, except those corresponding to the predefined and default codes as described in Table A.8. as well as CRC codes defined in Table

		A.7 Otherwise specified by means of the JPWL RA

When EPB is used for protecting the codestream, ID parameters of EPC marker segment shall not be present to describe this technique, if the method used is one of those included in Table A.6 (pre-defined codes), Table A.7 (CRC codes) or Table A.8 (Reed Solomon default codes).

Table A.10: Pepc parameter

Pepc	Parameter value
xxx0 xxxx	ESD is not present
xxx1 xxxx	One or more ESD are present
xx0x xxxx	RED is not present
xx1x xxxx	One or more RED are present
x0xx xxxx	EPB is not present
x1xx xxxx	One or more EPB are present
0xxx xxxx	informative techniques are not used
1xxx xxxx	One or more informative techniques are used
0000 0000 – 0000 1111	Reserved for future use

A.6.3 Error Sensitivity Descriptor (ESD)

The ESD marker segment can be placed in any valid position in a codestream Main and/or Tile-part header. It is allowed that more than one ESD marker segments are present in a Main or Tile-Part header.

Function: The ESD marker segment contains the sensitivity information for a given codestream or tile. See Annex D for more information on how to use ESD marker segments.

Usage: Main header and/or Tile-part headers.

Length: Variable, depending on the usage and on the granularity of the error sensitivity description.

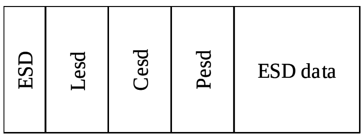

Figure A.3 – Syntax of the ESD marker segment

The syntax of the ESD marker segment is depicted in Figure A.3. The meaning of the data fields is discussed below; the range of possible values taken on by each parameter is discussed in **Error! Reference source not found.**. A detailed description of the ESD nomenclature and functionalities is provided in Annex D.

ESD: two bytes marker (0xFF68).

Lesd: length of the data structure in bytes (two bytes).

Cesd: specifies which component the ESD data refer to (one or two bytes).

Pesd: field describing the usage of the data structure (one byte).

ESD data: Records of error sensitivity values (variable length data field).

Table A.11 – Parameters of the ESD marker segment

Parameter	Size (bits)	Values
ESD	16	0xFF67
Lesd	16	4 – (2^{16}-1)
Cesd	8 16	0-255 if Csiz < 257 0-16383 if Csiz ≥ 257 Specifies which component the error sensitivity data refer to.
Pesd	8	0 — 255 (see **Error! Reference source not found.**).
ESD data	Variable	This field contains sensitivity information related to the codestream data, in the format specified in Annex D.

Table A.12 – Value of Pesd parameter. Format: 0x$b_7b_6b_5b_4b_3b_2b_1b_0$

b_7b_6	These bits specify the codestream addressing mode: 00: packet mode 01: byte-range mode 10: packet-range mode 11: reserved for future use

$b_5b_4b_3$	These bits specify the type of error sensitivity description employed. 000: relative error sensitivity. 001: MSE 010: MSE reduction 011: PSNR 100: PSNR increase 101: MAXERR (absolute peak error) 110: TSE (total squared error) 111: reserved for future use.
b_2	If it is set to 0, one byte is used to represent each sensitivity value; if it is set to 1, two bytes are used to represent each sensitivity value.
b_1	0: two bytes are used to indicate the start and end bytes in the *byte-range mode*, and the start and end packets in the *packet-range mode*. 1: four bytes are used. When *packet mode* is used, this bit shall be set to 0.
b_0	If it is set to 1, error sensitivity values are average values among all the components. In this case, Cesd must be equal to 0.

When packet mode is used, use of JPEG 2000 Part 1 PLM or PLT marker segments is recommended.

A.6.4 Residual Error Descriptor (RED)

The RED marker segment can be placed in any valid position in the main or tile-part header. RED marker segment signals the presence of residual errors and may help to handle them.

After any form of channel decoding some residual error may still affect the codestream. As described in previous sections, these errors may be very harmful if located in one of the JPEG 2000 Part 1 headers. In order to allow the JPEG 2000 decoder to be aware of the presence and the location of these errors, as well as their category (e.g., bit flipping or erasures), JPWL uses RED to embed this information in the codestream. The RED marker segment can be operated in three different modes, namely byte-range mode, packet mode and packet-range mode

- In *byte-range mode*, each data unit is described by explicitly specifying its start and end byte in the codestream; the residual error value refers to that specific byte range. Start and end bytes are specified as two or four unsigned integers; this allows to deal with “normal” and “long” codestreams. Byte numbering in the codestream starts from zero If the RED is located in the Main header, byte numbering refers to the start of the codestream (including the SOC marker segment). If the RED is located in a Tile-part header, byte numbering refers to the start of that Tile-part (including the SOT marker segment).

- In *packet mode*, the data units are packets as defined in JPEG 2000 Part 1. A residual error value is specified for each and every packet in the codestream or tile-part, according to whether the RED is contained in the main header or in a tile-part header.

- In *packet-range mode*, a range of JPEG 2000 packets, defined by a start and an end packet identifies a data unit for which a residual error value is provided. Start and end packets are specified as two or four bytes unsigned integers.

When RED is in the Main header, and packet mode or packet range is used, the numbering of the packets corresponds to the order of the packets in the codestream. When RED is in Tile-Part header and packet mode or packet range is used, the numbering of the packets corresponds to the numbering used in JPEG 2000 part 1 Annex A.8.1, starting at zero at every new Tile.

The following figure describes the syntax of the RED data structure. It consists of the following fields:

- RED: two bytes marker (0xFF69).

- L_{RED}: length of the data structure in bytes (two bytes).

- P_{RED}: field describing the usage of the data structure (two bytes).

- RED data: Record of parameters related to residual error descriptor (variable length data field).

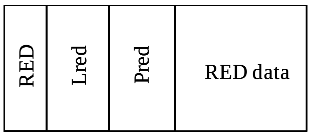

Figure A.4 – Syntax of the residual error descriptor marker segment.

Table A.13 – Residual error descriptor parameter values

Parameter	Size (bits)	Values
RED	16	0xFF69
LRED	16	3 – (216-1)

PRED	8	0 — 28-1 PRED Format: 0xb7b6b5b4b3b2b1b0 b7b6 Addressing Mode b7b6 = 00 Packet Addressing Mode b7b6 = 01 Byte-Range Addressing Mode b7b6 = 10 Packet-Range Addressing Mode b7b6 = 11 reserved for future use b5b4b3 Residual corruption level 000 – 111 b2 Reserved for future use b1 Address length b1 = 0 - 2-Bytes Addressing Mode b1 = 1 - 4-Bytes Addressing Mode b0 Error Free Codestream Indicator b0 = 0 Error Free codestream b-0 = 1 Error / Erasure presents in the codestream
	Variable	This field contains residual error information related to the codestream data, in the format specified in Annex E.

When packet mode is used, use of JPEG 2000 Part 1 PLM or PLT marker segments is recommended.

.

Annex B: Header Error Protection

(This Annex forms a normative and integral part of this International Standard.)

B.1 Introduction

During the establishment of JPEG 2000 standard, a set of error resilience tools have been selected for JPEG 2000 Part 1, for the transmission of JPEG 2000 compressed images in an error prone environment. Two types of tools are available, at the packet level, which enable synchronization, and at the entropy coding level, enabling error detection. For more information about the use of JPEG 2000 Part 1 error resilience tools, refer to Annex G and Annex H of this International Standard | Recommendation.

These tools are however based on one major hypothesis, namely that the headers (Main Header and Tile-part(s) header(s)) of the codestream syntax are guaranteed to be error free. However, in the case of error within the headers, the codestream is not decodable in a proper way, which might conduct to a decoder application crash. The worse is that, generally, it might not be possible to guarantee that the headers will be kept free of errors in many applications. The header protection mechanism detailed hereafter in this Annex describes a scheme that embeds the protection within the JPEG 2000 codestream. This mechanism is backward compatible with JPEG 2000 Part 1 codestream syntax.

B.1.1 JPEG 2000 Part 1 codestream syntax backward compatibility

A JPEG 2000 Part 1 compressed image uses markers and marker segments to delimit and signal the compressed information, organized in headers (Main and Tile-parts) and packets. This modular organization allows flexible codestream organization for progressive data representation, such as quality progressive and resolution progressive data progression. A JPEG 2000 Part 1 codestream always starts with the Main Header followed by one or several Tile-part Headers, each of them followed by compressed data packets, and ends with an End Of Codestream (EOC), as shown in Figure B.1.

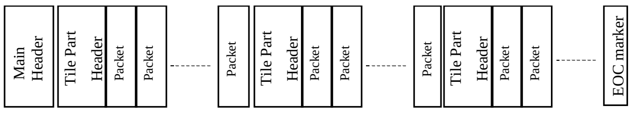

Figure B.1 – JPEG 2000 codestream structure.

The objective being to obtain a codestream compliant with JPEG 2000 Part 1 specifications [1] after the insertion of the redundant information, it is necessary to place this information in such a way that any JPEG 2000 Part 1 decoder will not try to interpret it. A solution for this is to insert the redundant information in a dedicated marker segment. A JPEG 2000 Part 1 decoder will then skip the unknown marker segment and be oblivious to the added data, whereas a JPWL decoder will be able to interpret and use the redundancy for header protection.

The conditions for such a mechanism to work are:

- that the decoder is able to locate the redundant information data block in the codestream without generating complex data indexing mechanism (that would also have to be protected against errors) nor by modifying the first marker segments imposed for the backward compatibility;
- that the marker itself and its length are included in the data range to be protected;
- that a defined block error code is used to protect at least up to the Error Protection Block marker segment parameters data.

The Error Protection Block marker segment (EPB) is placed immediately after the JPEG 2000 Part 1 markers mandatory locations:

- after the SOC and SIZ marker segments for the Main header;
- after the SOT marker for the Tile-part header.

The use of a systematic forward error-correction mechanism ensures that the first two conditions are verified.

B.1.2 Forward Error Correction mechanism

Error correction and detection codes are traditionally used to provide forward error correction capabilities in error prone environments [9]. Systematic codes are those that produce a given amount of redundant information, while keeping untouched the original data.

Considering that JPEG 2000 Part 1 codestreams are byte aligned, it is especially interesting to work with the Galois Field GF(2^8) to provide error-correction capability. A well-known and well-suited family of systematic codes in this context is Reed-Solomon (RS) codes. In the following, we will consider the example of RS codes as FEC codes for header protection, and denote them by RS(N,K), where N is the codewords symbol length and K the number of information symbols.

The RS(N,K) applied to K bytes will generate N-K redundancy bytes, that may be placed after the K original (systematic) bytes, this process being applied as long as necessary, as illustrated in Figure. B.2.

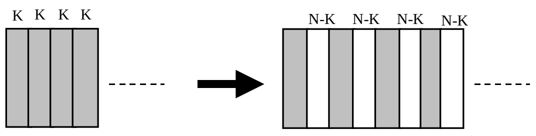

Figure B.2 – Example of redundancy generation with an RS(N,K) code.

B.2 Predefined error correction codes

Since during the transmission on error prone environment errors may occur anywhere in the JPEG 2000 codestream, the header protection tool cannot rely on parameter information for indicating the error correction code to use. Therefore, a set of predefined codes has been defined whereas EPB marker segment syntax allows to choose others for some part of the headers. The possible systematic error correction codes are listed in

Table A.6.

In order to fight efficiently harsh transmission conditions, these predefined codes offer a large correction capacity, while limiting byte padding. Three predefined error correction codes have been defined for protecting the Main Header and the Tile Part Headers:

RS(160,64) to be used for the first EPB marker segment of the Main Header

RS(80,25) to be used for the first EPB marker segment of a Tile-part Header

RS(40,13) to be used for the other EPB marker segments of both the Main Header and the Tile-part header.

These Reed-Solomon codes are always used for the protection of the beginning of Main and Tile-part headers, as well as the parameters of any EPB marker segment. Other codes may be used for protecting the other parts of the headers by using an appropriate Pepb value.

The use of error protection can be stopped within the current header by using the appropriate LDPepb data length and by indicating the end of the error protected data range through Pepb value.

B.3 Use of EPB for Header protection

B.3.1 Main Header error protection

When encountering an EPB marker segment, the JPWL decoder may apply the correction of the codestream it refers to. For the Main header, when doing this correction, first, the JPWL decoder applies this correction to the SOC and SIZ marker segments, as well as to the EPB marker segment parameters. This range of data corresponds to L1 in Figure B.3. The redundant information necessary for this correction are located at the beginning of the EPB redundant data, illustrated by L2 in Figure B.3.

Once the EPB parameters have been corrected, it is then possible to take these parameters into account, especially the Depb, LDPepb and the Pepb parameters. These parameters are necessary for using the error correction for the remaining parts of the Main header. They allow to adapt the error correcting code redundancy to the error conditions. This structure enables to protect differently fundamental JPEG 2000 Part 1 marker segments, such as QCD, whereas optional marker segments like PLM can be protected with less redundancy or even not protected at all.

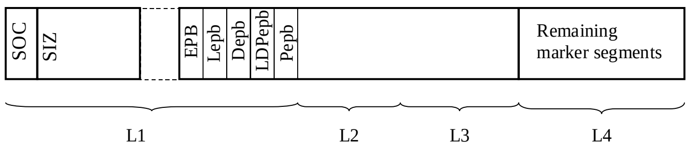

Figure B.3 – EPB marker position in Main Header and protection regions

Figure B.3 illustrates the case where a single EPB marker segment is used to protect the Main header. In this case, L1 data are protected by the L2 part of the EPB data, using the default Main header error correction code. L4 data are protected using L3, with the error correction code specified in the Pepb parameter.

The LDPepb parameter allows to stop the error protection at any byte aligned location within the Main header. LDPebp gives the number of bytes which are protected using the default error correction code and the code specified in Pepb parameter. As an example in Figure B.3, LDPepb is equal to L1 + L4 bytes. LDPepb shall not conduct to consider data which are outside of the Main header.

The Main Header can contain several EPB marker segments, that may be unpacked or packed, which means that they appear one after another, before the remaining Main Header information. Unpacked EPBs means that they will appear just before the data part they refer (or “refer to”, to be checked). An example of packed and unpacked EPBs is given later in this Annex. For each new EPB, the predefined code RS(40,13) has to be used for the correction of its own EPB parameters.

B.3.2 Tile-Part Header error protection

When EPB is present in Tile-part header(s), the JPWL decoder may apply correction to the SOT marker segment, as well as to the EPB marker segment parameters. This range of data correspond to L1 in Figure B.4. The redundant information necessary for this correction are located in the beginning of the EPB redundant data, illustrated by L2 in Figure B.4

Once the EPB parameters have been corrected, it is then possible to take these parameters into account, especially the Depb, LDPepb and the Pepb parameters. These parameters are necessary for using the error correction for the remaining parts of the Tile-part header. They allow to adapt the error correcting code redundancy to the error conditions. This structure enables to protect differently fundamental JPEG 2000 Part 1 marker segments, such as QCD, whereas optional marker segments like PLT can be protected with less redundancy or even not protected at all.

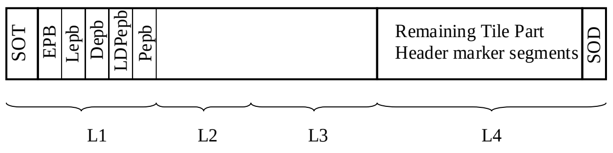

Figure B.4 – EPB marker position in Tile Part Header and protection regions (single EPB case)

Figure B.4 illustrates the case where a single EPB marker segment is used to protect the Tile-part header. In this case, L1 data are protected by the L2 part of the EPB data, using the default Tile-part header error correction code. L4 data are protected using L3, with the error correction code specified in the Pepb parameter.

The LDPepb parameter allows to stop the error protection at any byte aligned location within the Tile-part header. LDPebp gives the number of bytes which are protected using the default error correction code and the code specified in Pepb parameter. As an example in Figure B.4, LDPepb is equal to L1 + L4 bytes. LDPepb for EPBs present in Tile-part headers may conduct to consider data which are outside of the Tile-part header. This feature is necessary for enabling the use of EPB for Unequal Error Protection purposes as explained in Annex I.

B.3.3 Packed and unpacked Error Protection Blocks

In the case where the Main or Tile-part Header is of large size, for example due to the inclusion of several PPM or PPT marker segments, it is possible to use more than one EPB marker segment. The Depb parameter, specified in Table A.5, allows this functionality. This parameter allows also to indicate how the EPB information was placed in the header. There are two possibilities for chaining this information, while keeping the error protection feature among them:

One way is to interleave between the several EPB, some marker segments of the header to be protected. This structure is called "unpacked EPB marker segments".

The other way, which provides optimal redundant information length, called "packed EPB marker segments" consists in grouping together all the EPB marker segments before the remaining marker segments of the header.

In both cases, the "Last EPB marker" information allows to identify that the EPB marker segments is the latest in the header. It is particularly interesting when using packed EPB option, where it allows to locate the remaining header data, which is to be found just after the current EPB marker segment.

In both cases, for each new EPB, except the first one in the header, the predefined code RS(40,13) has to be used for the correction of the EPB parameters, whereas the remaining data considered by LDPepb parameter are protected using the tools described in Pepb.

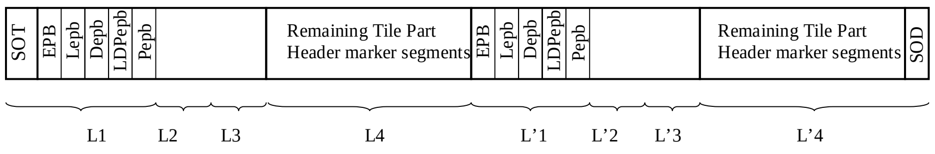
SOT
EPB
Lepb
Depb
LDPepb
Pepb
Remaining Tile Part Header marker segments
EPB
Lepb
Depb
LDPepb
Pepb
Remaining Tile Part Header marker segments
SOD
L1
L2
L3
L4
L’1
L’2
L’3
L’4

Figure B.5 – unpacked EPB markers position in Tile Part Header and protection regions (several EPB case)

Figure B.9 illustrates the case where two unpacked EPB marker segments are used to protect the Tile Part header. In this case, L1 data are protected by the L2 part of the first EPB data, and L’1 data are protected by the L’2 part of the second EPB data , using the default Tile-Part header error correction code. L4 data are protected using L3, with the error correction code specified in the Pepb parameter of the first EPB marker segment. L’4 data are protected using L’3, with the error correction code specified in the Pepb parameter of the second EPB marker segment.

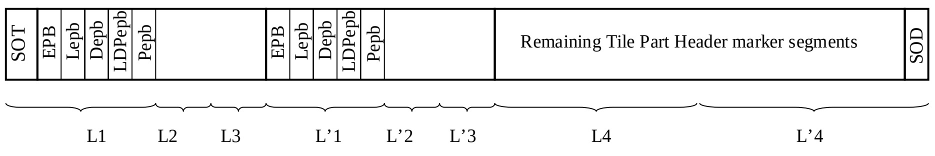

Figure B.6 – packed EPB markers position in Tile Part Header and protection regions (several EPB case)

Figure B.10 illustrates the case where two packed EPB marker segments are used to protect the Tile Part header. In this case, L1 data are protected by the L2 part of the first EPB data, and L’1 data are protected by the L’2 part of the second EPB data , using the default Tile-part header error correction code. L4 data are protected using L3, with the error correction code specified in the Pepb parameter of the first EPB marker segment. L’4 data are protected using L’3, with the error correction code specified in the Pepb parameter of the second EPB marker segment.

B.3.4 Cyclic Redundancy Check

Pepb parameter can describe two types of different techniques, Cyclic Redundancy Check and error correction, while describing the parameters to be used by these techniques. In order to insure that data have been transmitted error free, most communications protocols use a parity check process called Cyclic Redundancy Check (CRC) [12]. CRC codes are a subset of linear block codes.

CRC can be used in EPB, instead of error correction redundancy data, except for the parameters of the EPB marker segments, which are always protected using the appropriate default error protection code. The use of CRC is signaled by the Pepb parameter of the EPB marker segment (See

Table A.6 and Table A.7).

M bits CRC has the mathematical property of detecting all errors that occur in M or fewer consecutive bits, and the probability of 1 over 2^M of not detecting an error. In typical applications, the CRC is 16 bits long.

An M bits CRC is based on a polynomial of degree M. JPWL is using the two following polynoms:

For 16 bits CRC (CCITT-CRC/X25): $x^{16}+x^{12}+x^5+1$

For 32 bits CRC (AUTODIN/ETHERNET): $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Annex C: Error Protection Capability

(This Annex forms a normative and integral part of this International Standard.)

C.1 Usage of the EPC marker segment

The EPC marker segment signals whether the three other normative marker segments defined by JPWL, namely the error sensitivity descriptor (ESD), the residual error descriptor (RED) and the error protection block (EPB) are present in the codestream. In addition, it signals the use of informative tools in order to protect the codestream against transmission errors. These tools include techniques such as error resilient entropy coding, FEC codes, UEP and data partitioning/interleaving. These informative tools are not defined in this International Standard | Recommendation. Instead, they are registered with the JPWL RA. Upon registration, each tool is assigned an ID, which uniquely identifies it. More information about the use of the RA can be found in Annex K. The EPC marker segment also makes provisions for the handling of parameters related to these informative tools. When encountering a JPWL codestream, the decoder can identify the tool(s) which have been used to protect this codestream by parsing the EPC marker segment and by querying the RA. The decoder can then take the appropriate steps to decode the codestream, e.g. acquire or download the appropriate tool.

The EPC marker segment is mandatory in the Main header, and optional in the Tile-part header. At most one EPC can be present in each Main and Tile-part header.

An EPC marker segment can contain more than one ID (with the related parameters), indicating that more than one error protection technique has been applied to the codestream. The order in which the IDs have to appear in the EPC is the order in which the techniques have to be applied at the *decoder* side. An EPC marker segment is allowed to contain no ID.

If a technique applies to the entire codestream, its ID has to be indicated in the EPC in the Main header. An EPC in a Tile-part header can contain IDs of techniques that are applied to that Tile-part.

It is up to the encoder to ensure that combinations of two or more techniques lead to consistent and meaningful results, and that the decoder has enough resources to carry out decoding. To avoid processing overload, in case of multiple techniques it is not mandatory for the decoder to decode all techniques; this also allows a decoder to process only those parts of the codestream protected by known techniques. Moreover, it is worth noticing that combinations of two or more techniques may be registered in the RA as a single new technique.

C.2 P_{CRC}

The P_{CRC} is a 16 bits parameter that contains parity check bits to verify whether the EPC marker has been corrupted by errors. Specifically, the CRC is computed on a codeword consisting of the concatenation of EPC, L_{EPC}, CL, P_{EPC} and the complete sequence of $ID^{(i)}$, $L_{ID}^{(i)}$ and $P_{ID}^{(i)}$ (i.e., the complete marker segment excluding P_{CRC} itself). The CCITT-CRC/X25 defined in Annex B.3.4 has to be used to generate the parity bits.

C.3 Codestream length (CL)

A compressed video sequence can be transmitted as a sequence of raw codestreams. In this case, the decoder has to take care of correctly synchronizing on the start of each new frame. While in the error-free case this is not an issue, as the SOC and EOC markers can be parsed to locate the start and the end of each codestream, in an error-prone environment this may not be the case, since these markers can be corrupted and hence unusable. For this reason, it is useful that some additional “redundant” information is inserted, which may be exploited by the decoder to improve its ability to resynchronize after a decoding failure. To this end, the EPC marker segment contains the CL parameter, which specifies the total length *L* in bytes of the current codestream. As a consequence, if the EOC marker is not found where it is expected, the decoder can skip *L* bytes starting from the SOC, and verify if the SOC marker of the next frame is uncorrupted. Otherwise, if the SOC marker of the current frame is also corrupted, the decoder can seek the EOC marker of the last frame, skip *L*+2 bytes, and verify the presence of the SOC marker of the next frame.

The CL parameter is an unsigned integer number represented on four bytes, and represents the length in bytes of the current codestream, or zero if this information is not available.

C.4 P_{EPC}

P_{EPC} is a 8-bits parameter which indicates the presence of ESD, RED and EPB marker segments in the codestream, as well as the use of informative tools. This information is useful in order for a decoder to quickly know whether the codestream can be decoded and which information is available in the codestream.

C.5 Identification of tools (ID)

Informative tools to protect the codestream against transmission errors have to be registered with the RA (see Annex K). Upon registration, each tool is assigned an ID, which uniquely identifies it.

At the encoder side, when using a registered informative tool, the corresponding ID is inserted in EPC in order to signal its use. At the decoder side, the decoder parses the EPC marker segment and can identify the registered informative tools which have been used. The decoder can then query the RA about these tools, and take the most appropriate actions to decode the codestream (e.g. acquire or download the appropriate tool).

The values 0 to 15 of ID are reserved.

C.6 Parameters for tools (P_{ID})

This parameter can be used to signal parameters for the tools which have been applied to the codestream.

The format of P_{ID} is not specified in this standard, but is registered by means of the RA at the time of the tool registration.

Annex D: Error Sensitivity Descriptor

(This Annex forms a normative and integral part of this International Standard.)

D.1 Introduction and applications (informative)

Error sensitivity information provides a measure of how much different parts of the codestream are sensitive to errors, i.e. the effect of losing each part on the quality of the decoded image. Error sensitivity signalling has several possible applications; a few of them are described in the following.

- Unequal error protection. In UEP, more powerful codes are assigned to the most sensitive parts of the codestream. This typically provides higher average PSNR with respect to the equal protection strategy. The allocation of codes to each part of the codestream depends on the sensitivity of each part. It is worth noticing that, for unequal protection, the error sensitivity information is exploited by the encoder, but is not required by the decoder, which just needs to know which protection parameters have been employed (see example of UEP in Annex I).

- Rate transcoding. In some applications, there may be a subsystem handling the transmission of images and video from a source to one or more users. The subsystem can be aware of the codestream syntax, and can carry out a basic parsing. If rate transcoding must be carried out to adapt the incoming data-rate to the current transmission conditions, the subsystem can adopt intelligent quality of service policies by not only truncating the codestream, but also looking up the error sensitivity table to make sure that the selected truncation rate provides a reasonable degree of image quality.

- Selective retransmissions. The subsystem capabilities can also be exploited to optimize the retransmission management, by allocating a larger number of retransmission attempts to those codestream portions that are more critical from the quality standpoint based on the error sensitivity information.

- Smart prefetching. In streaming video applications, the subsystem can decide to prefetch the most important packets of the current and next frames, and to send them in advance. This allows to carry out a larger number of retransmissions if some of these packets should be lost. The most important codestream portions can be selected by simply looking up the ESD content.

It is worth noticing that the error sensitivity information is less critical than other portions of a JPEG 2000 Part 11 compliant codestream, as it is not strictly needed for decoding.

D.2 Marker definition and position in the codestream (normative)

The ESD is a marker segment that contains information related to the error sensitivity of different parts of a codestream or tile.

The ESD marker segment shall appear in the Main header and/or in the Tile-part headers. If it appears in the Main header, its sensitivity description shall apply to the complete codestream, whereas if it appears in a Tile-part header the description shall apply only to that Tile-part. It is intended that, if an ESD marker segment is present in both the Main header and the Tile-part header, in case of ambiguity the information in the Tile-part header ESD will override that in the Main header ESD. More than one ESD is allowed in each Main and Tile-Part header; this can be used to provide error sensitivity using *different* metrics, e.g., both MSE and MAXERR. It is however possible that two ESD marker segments are present in a given header and use the same error metric, and they cover overlapping portions of the codestream. In order to avoid ambiguity in the error sensitivity description, it is intended that, as far as the overlapping portions with same metric are concerned, the error sensitivity values in the last ESD have to be used.

D.3 Codestream subdivision into data units (Normative)

Sensitivity information is provided for one or more specific data units in the codestream. This Annex specifies three different addressing modes for defining data units, namely *packet mode, byte-range mode* and *packet-range mode*.

- In *byte-range mode*, each data unit is described by explicitly specifying its start and end byte in the codestream; the sensitivity value refers to that specific byte range. Start and end bytes are specified as two- or four-bytes unsigned integers; this allows to deal with “normal” and “long” codestreams. Byte numbering in the codestream starts from zero. If the ESD is located in the Main header, byte numbering refers to the start of the codestream (including the SOC marker segment). If the ESD is located in a Tile-part header, byte numbering refers to the start of that Tile-part (including the SOT marker segment).

- In *packet mode*, the data units are packets as defined in JPEG 2000 Part 1. A sensitivity value is specified for each and every packet in the codestream or Tile-part, according to whether the ESD is contained in the Main header or in a Tile-part header.

- In *packet-range mode*, a range of JPEG 2000 packets, defined by a start and an end packet, identifies a data unit for which a sensitivity value is provided. Start and end packets are specified as two- or four-bytes unsigned integers.

When ESD is in the Main header, and packet mode or packet range is used, the numbering of the packets corresponds to the order of the packets in the codestream. When ESD is in Tile-part header and packet mode or packet range is used, the numbering of the packets corresponds to the numbering used in JPEG 2000 Part 1 Annex A.8.1, starting at zero at every new Tile-part.

D.4 Sensitivity information (Normative)

D.4.1 Meaning of sensitivity values

For multi-component images, error sensitivity values contained in the ESD marker segment can refer to a single component, or can be intended as average values among all components, as specified by Pesd.

Sensitivity values can be expressed in two different ways, i.e. as *relative* or *absolute* sensitivity values. (Note that the definition of *relative sensitivity* in JPWL is equivalent to the *relative importance* in JPSEC.) Relative sensitivity is expressed as an unsigned integer number describing the error sensitivity of a given codestream portion with respect to other portions. Absolute sensitivity refers to sensitivity information related to a specific error metric, such as MSE, PSNR or MAXERR (maximum absolute error). The Pesd parameter specifies whether the relative or absolute sensitivity mode is being used.

Relative sensitivity information for each codestream data unit shall be expressed as an unsigned integer number ranging from 0 to 2^P-1. The parameter P can be either 8 or 16; this allows choosing between a rough but compact description, and a more precise one. It is intended that the highest values of sensitivity shall be assigned to the “most important” parts of the codestream. The value 2^P-1 shall be exclusively reserved to Main and Tile-part headers. In particular, data units partially or totally containing the Main or Tile-part header of a given codestream may have sensitivity equal to zero; conversely, data units *not* containing portions of the Main or Tile-part header shall not have sensitivity equal to zero. The value 0 shall be used for parts of the codestream for which sensitivity information is not specified. All the other values shall represent the relative importance of the considered portion of the codestream, in the [1,2^P-2] range, with larger numbers indicating the highest levels of importance.

Absolute sensitivity values can also be expressed using one or two bytes, as indicated in the Pesd parameter. The value 0xFF for the one-byte case (resp. 0xFFFF for the two-byte case) shall be exclusively reserved to Main and Tile-part headers. In particular, data units partially or totally containing the Main or Tile-part header of a given codestream may have sensitivity equal to zero; conversely, data units *not* containing portions of the Main or Tile-part header shall not have sensitivity equal to zero. The value 0 shall be used for parts of the codestream for which sensitivity information is not specified. All the other values shall represent the metric value related to the considered portion of the codestream.

Absolute sensitivity values are tied to a specific error/quality metric, such as MSE, TSE, PSNR or MAXERR. Normal and incremental error metrics can be used, such as “MSE” and “MSE decrease”, or “PSNR” and “PSNR increase”. “MSE” means the mean-squared-error incurred by decoding up to (and including) the data unit for which MSE is specified; “MSE decrease” specifies the improvement in MSE achieved by decoding that data unit; and analogously for PSNR. TSE refers to the total squared error, as opposed to the mean squared error.

The error/quality metrics refer to the whole image or to a tile, according to whether the ESD is included in a Main or Tile-part header.

Since this information may be difficult to estimate, no specific degree of accuracy is required. These metrics shall be expressed in linear units; in particular, denoting as x_i (with $i=1,...,N$) the values of N pixels of the original image, and as r_i those of the decoded image, the error metrics are defined as follows:

$$\mathrm{MSE} = \frac{1}{\mathrm{N}}\sum_{i=1}^{N}(x_i - r_i)^2$$

$$\mathrm{TSE} = \mathrm{MSE} * \mathrm{N}$$

$$\mathrm{PSNR} = \frac{M^2}{\mathrm{MSE}}$$

$$\mathrm{MAXERR} = \max_i |x_i - r_i|$$

where M is the maximum value that the original image can assume in the given representation (e.g., for 8-bit images M=255); assuming that the image is stored using Q significant bits, M shall be equal to 2^{Q}-1 if the data are unsigned integers, and to 2^{Q-1}-1 if they are signed integers.

In the two-byte format, absolute sensitivity values shall be expressed as a two-byte number in pseudo floating-point format. Each 16-bit number contains the exponent (5 bits) and mantissa (11 bits) of the metric value. Note that a sign bit is unnecessary since metric values are non-negative. In particular, the floating-point value V of the metric is given by the following formula (which is the same as in Annex. E.1.1.1 of [1] for the determination of the quantization step size):

$$V = 2^{\varepsilon-15}\left(1 + \frac{\mu}{2^{11}}\right) \quad \text{if } \varepsilon \neq 0$$

$$V = 0 \quad \text{if } \varepsilon = 0$$

where ε is the unsigned integer obtained from the first five most significant bits of the parameter, and μ the unsigned integer obtained from the remaining 11 bits. The special case of $V = \infty$ correspond to μ=0 and ε=31. Notice that values that would underflow the representation are set to zero.

The algorithm to compute s, ε and μ is not defined as a mandatory part of this standard. A possible technique performs the following steps (an example of conversion of the number 12.25 is provided). If V=0, set $\varepsilon = \mu = 0$. Otherwise:

– convert V to a binary number ($12.25_{10} = 1100.01_2$);

- Normalize the number; this means there should be a 1 digit to the left of the binary point and multiplication by the appropriate power of two to represent the original value. The normalized form of 1100.01 is 1.10001×2^3;

- the exponent is the power of 2, presented in excess notation. The exponent bias is 15; hence for this example the exponent is represented as 18_{10} (10010_2);

- the mantissa represents the significant bits, *except for the bit to the left of the binary point*, which is always one and therefore does not need to be stored; zeros are possibly appended so as to obtain 11 bits. For this example, the mantissa is 10001000000.

The one-byte format is defined as follows, and is exactly the same as the one-byte format for total distortion field in JPSEC. The metric value is expressed using a one-byte distortion field with a pseudo floating-point type representation. The 8 bits available in the distortion field are allocated to the mantissa (m) and base-16 exponent (exp) of the metric value to provide an appropriate trade-off between accuracy and dynamic range. Note that, as in the two-byte format, a sign bit is unnecessary since metric values are non-negative. To cover a sufficient dynamic range, base 16 is used and 4 bits are used for the exponent (exp). The mantissa (m) is expressed using 4 bits. Therefore, the metric value V is given by

$$V = m \times 16^{exp}$$

where m has a value in the range $0 \le m \le 15$ and exp has a value in the range $0 \le exp \le 15$. A value of zero is represented by m = 0 and exp = 0, that is by the metric field being zero. By allocating 4 bits for the mantissa m the accuracy is within $\frac{1}{2} \times (1/2^4) = 1/32$ or about 3%. With 4 bits for the exponent and using base-16 the dynamic range is from 0 to max, where max is given by m = 15 and exp = 15 which corresponds to a metric value of $15 \times 16^{15} = 1.7 \times 10^{19}$.

Note that with this format for the metric value, a comparison between two metrics to determine which is larger can be simply achieved by comparing the two values as unsigned char. Specifically, to perform this comparison there is no need to convert from the pseudo floating-point format to the actual value in order to determine which of two values is larger or smaller. This property can simplify the processing in various applications.

D.4.2 ESD data field (Normative)

Three cases are considered in the definition of the ESD data field, according to the codestream addressing method in use.

In *packet mode*, a sensitivity value is provided for each and every packet in the codestream or Tile-part, according to whether the ESD is contained in the Main header or in a Tile-part header; the ESD data field contains the concatenation of (relative or absolute) sensitivity values for each packet. If the ESD marker segment is in the Main header, it is assumed that these values appear in the order specified by the packet numbering of the SOP marker (see JPEG 2000 Part 1, Annex A.8.1); if it is in a Tile-part header, the ESD data field contains a concatenation of sensitivity values for all packets contained in that Tile-part. Notice that the Lesd parameter can be used to compute in advance the number of sensitivity values contained in the ESD data field.

In *byte-range mode*, the ESD data field is again a concatenation of records. The length of each data record depends on whether two or four bytes are employed for the specification of each start and end byte, and on whether one or two bytes are used for the sensitivity description. These parameters can be inferred from Pesd. Each record contains, in the following order, the start byte of the data unit, the end byte of the data unit, and the (absolute or relative) sensitivity value for the data unit. Start and end bytes refer to the beginning of the codestream or of a Tile-part, according to whether the ESD marker segment is included in a Main or Tile-part header. Notice that the Lesd parameter can be used to compute in advance the number of records contained in the ESD marker segment.

In *packet-range mode*, the ESD data field is again a concatenation of records. Each record has exactly the same structure as in the *byte-range mode*, except for the fact that start and end packets are employed, instead of start and end bytes, to define each data unit. Start and end packets are computed from the beginning of the codestream or of a Tile-part, according to whether the ESD marker segment is included in a main or Tile-part header.

D.5 Examples and guidelines (informative)

In the following we provide two examples of possible usage of the ESD marker segment. The first example refers to relative sensitivity, whereas the second one to absolute sensitivity.

D.5.1 Example 1 – relative sensitivity with packet mode

Consider the transmission of a grayscale image at 0.5 bits per pixel (bpp) in non-reversible mode. A JPEG 2000 Part 1 compliant encoder is employed to generate a codestream suitable for transmission over a wireless channel; JPWL is employed to add error sensitivity information to that codestream, and specifically an ESD marker segment, so as to optimize the decoder performance. A JPEG 2000 Part 1 encoder can use arithmetic encoder termination, along with SOP and EPH markers, as error-resilience tools. A PPM marker segment is used to pack all the packet headers in the Main header, so that all header information is grouped at the beginning of the codestream, and can be protected more easily. A PLM marker segment in the Main header would also be useful, so as to summarize the lengths of all packets in the codestream; however, for simplicity, it is not used in this example. The layer-progressive mode is employed for scalability, with layers at 0.25 and 0.5 bpp (i.e. the target bit-rate). The resulting codestream consists of 12 packets. During rate allocation, the encoder finds out that decoding at 0.25 bpp would yield PSNR = 33.72, whereas decoding at 0.5 bpp would yield PSNR = 37.12. Therefore, it is decided that, as to the error sensitivity information in terms of PSNR, the first half of the codestream is worth 33.72, whereas the second half is worth 3.4. In terms of JPEG 2000 Part 1 packets, the data provided by the rate allocator are reported in **Error! Reference source not found.**. In this table, the column “PSNR” reports the PSNR achieved by decoding the image up to a certain packet; the column “Δ-PSNR” contains an estimate of the relative contribution of each packet, computed as difference between the PSNR obtained by decoding up to the current and up to the previous packet. A relative error sensitivity can be simply defined by labeling with *S=1* the packet with higher PSNR contribution, and then with increasing values of *S* packets with decreasing Δ-PSNR, up to the first layer at 0.25 bpp. For all packets in the second layer the sensitivity is the same and is equal to *S*=7.

Table D.1 – Error sensitivity computation

Packet number	Rate (bpp)	PSNR	Δ-PSNR	*S*
1	0.024	14.48	14.48	1
2	0.04	21.88	7.4	2
3	0.077	24.84	2.96	5
4	0.142	29.30	4.46	3
5	0.227	33.09	3.79	4
6	0.253	33.72	0.63	6
7	0.254	33.93	0.21	7
8	0.257	33.95	0.02	7
9	0.269	34.06	0.11	7
10	0.312	34.54	0.48	7

11	0.397	35.53	0.99	7
12	0.5	37.12	1.59	7

To conclude, we write an ESD marker segment using relative error sensitivity (one byte per value) as computed above for this example codestream, and packet-mode as indexing mode for the codestream; the metric is specified for the single image component. The resulting hexadecimal representation of the ESD marker segment is the following (parameters are separated by "|", and records by blanks):

FF68 | 0010 | 01 | 00 | 01 02 05 03 04 06 07 07 07 07 07 07

D.5.2 Example 2 – absolute sensitivity with byte-range mode

In this second example we use the byte-range mode (two bytes per start and end byte) and absolute sensitivity in the two-byte format. In particular, "PSNR increase" is selected as error metric. From the previous example we recall that, at rates 0.25 and 0.5 bpp, the PSNR equals 33.72 and 37.12 respectively (notice that the 0.25 and 0.5 rates refer to the codestream *not including* the ESD marker segment); for convenience, we approximate these PSNR values with 33.5 and 37.0. Moreover, codestream parsing reveals that the first 554 bytes contain the Main and Tile-part headers. It is then decided to describe three data units, namely the headers, the first, and the second half of the codestream. In particular, the data units are from byte 1 to byte 554 with *S=0*, from byte 555 to byte 8224 with S=33.5, and from byte 8225 to byte 16288 with *S*=3.5. The sensitivity values are specified as average values among all components; since there is only one component, this is equivalent to saying that these values refer to component 1. The resulting marker segment, using the pseudo floating-point notation for *S*, is as follows:

FF68 | 0016 | 00 | 65 | 0001 022A 0000 022B 2020 A060 2021 3FA0 4300.

Annex E: Residual Errors Descriptor

(This Annex forms a normative and integral part of this International Standard.)

E.1 Introduction (Informative)

RED marker segment signals the presence of residual errors that may still affect the codestream after a JPWL decoder processing. This information on the presence and type (erasure or flipping) of errors may be exploited by a "JPWL aware" JPEG 2000 decoder to improve the decoding capabilities or to apply some techniques as:

- Selective retransmissions
- Error concealment
- Discard of the corrupted information if not visually relevant

E.2 Signaling of residual errors (Normative)

The RED marker segment can be operated in three different modes, namely byte-range mode, packet mode and packet-range mode

Byte-range mode:

- In *byte-range mode*, each data unit is described by explicitly specifying its start and end byte in the codestream; the residual error value refers to that specific byte range. Start and end bytes are specified as two or four unsigned integers; this allows to deal with "normal" and "long" codestreams. Byte numbering in the codestream starts from zero. If the RED is located in the Main header, byte numbering refers to the start of the codestream (including the SOC marker segment). If the RED is located in a Tile-part header, byte numbering refers to the start of that Tile-part (including the SOT marker segment). When adopting a Reed Solomon decoder the typical length of each data block is the one of the selected Reed Solomon codeword.
- The following two bytes contain the number (if available) of errors in the data block (0x0000 – 0xFFFE) or a generic indication of presence of errors (0xFFFF) in case the exact number of errors is not available.

Packet mode:

- In *packet mode*, the data units are packets as defined in JPEG 2000 Part 1. A residual error value is specified for each and every packet in the codestream or tile-part, according to whether the RED is contained in the main header or in a tile-part header. The following two bytes contain:
  - the number (if available) of errors in the data block (0x0000 – 0xFFFD)
  - the indication of a packet erasure (0xFFFE)
  - a generic indication of presence of errors (0xFFFF) in case the exact number of errors is not available.

Packet-range mode:

- In *packet-range mode*, a range of JPEG 2000 packets, defined by a start and an end packet, identifies a data unit for which a residual error value is provided. Start and end packets are specified as two or four bytes unsigned integers.

- The following two bytes contain the number (if available) of errors in the data block (0x0000 – 0xFFFE) or a generic indication of presence of errors (0xFFFF) in case the exact number of errors is not available.

E.3 Examples (informative)

In the following we provide two examples of possible usage of the RED marker segment. The first example refers to a packet-mode with sparse errors; the second one refers to erasures in packet mode configuration.

E.3.1 Example 1 – Residual Error Descriptor with packet mode and sparse errors

Let us consider the transmission of a grayscale image at 0.5 bits per pixel (bpp) in non-reversible mode. A JPEG 2000 Part 1 compliant encoder is employed to generate a codestream suitable for transmission over a wireless channel; JPWL is employed to add error sensitivity information to that codestream, and specifically an ESD marker segment, so as to optimize the decoder performance. A JPEG 2000 Part 1 encoder can use arithmetic encoder termination, along with SOP and EPH markers, as error-resilience tools. A PPM marker segment is used to pack all the packet headers in the main header, so that all header information is grouped at the beginning of the codestream, and can be protected more easily.

Let us consider that EPB is used to protect both headers and data, adopting Reed Solomon codes, eventually adopting a UEP scheme taking into account ESD information. When working in packet mode, it may be convenient to set EPB length equal to packet length. These simplify resynchronization in case of entire packet loss.

At receiver side after JPWL decoder it may happen that one or more of the less protected packets still contain errors as they exceed error protection capabilities of the selected Reed Solomon code inside an EPB.

If not created by the encoder the JPWL decoder will optionally create the RED marker to signal the residual errors. Assuming, as an example, that packets 7 and 8 still contains errors, the resulting representation of the RED marker segment is the following (parameters are separated by "|", and records by blanks):

```
FF69h    | 001Ch   | 00010X01b    | 00h 00h 00h  00h 00h 00h FFh FFh 00h 00h 00h 00h

RED      | Lred    | Pred         | RED data
```

E.3.2 Example 2 – Residual Error Descriptor with packet mode and packet loss

Let us consider the same scenario of example 1, i.e. the same image and the same protection scheme. In this case a packet loss model is adopted for error generation as for example in UDP connections. Let us assume that in this case an UDP packet containing JPEG 2000 packets 7 and 8 is completely lost.

If the RED marker segment is not already existing the JPWL decoder will optionally create the RED marker to signal this situation; the resulting representation of the RED marker segment is the following (parameters are separated by "|", and records by blanks):

```
FF69h    | 001Ch   | 00010X01b    | 00h 00h 00h  00h 00h 00h FEh FEh 00h 00h 00h 00h

RED      | Lred    | Pred         | RED data
```

It should be noted that when erasures occur, the decoder has to update the length parameters appearing in marker segments, or fill the gaps with padding dummy data.

Annex F: Guidelines for encoding JPEG 2000 codestreams in the context of error prone environments

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

F.1 Introduction

This Annex provides some guidelines as to the use of JPEG 2000 Part 1 and JPWL tools in error-prone environments. JPEG 2000 Part 1 defines a set of error-resilience tools that can be used to encode an image in an error-prone environment. These tools are categorized as in Table F.1. JPWL defines a set of additional error protection tools that are able to enhance the codestream resilience towards transmission errors and to help the decoder manage residual errors.

F.2 JPEG 2000 Part 1 error resilience tools

Table F.1 - JPEG 2000 Part 1 error resilience tools

Type of tool	Name
Entropy coding level	code-blocks termination of the arithmetic coder for each pass predictable termination segmentation symbols
Packet level	short packet format (packed packet headers) packet with resynchronization marker (SOP) precincts

Since channel errors (or packet losses) can exhibit different patterns, in general it is not possible to know in advance which combination of error resilience tools will yield the best results. However, an extensive study has been carried out in the context of error-prone channels, considering a few realistic application scenarios such as 3GPP networks, Digital Radio Mondiale (DRM), and IEEE 802.11 wireless LANs. From these studies, a few general guidelines can be inferred.

As to the error resilience tools defined in JPEG 2000 Part 1, the following remarks are in order.

In general, one may want to assign a high level of protection to the main and Tile-part header, as these headers are necessary in order to correctly perform decoding. In the error-prone case, the packet headers are also very useful, as they allow the decoder to skip corrupted coding passes and to resynchronize and continue decoding. It is very easy to protect the packet headers along with the main and/or tile part headers if the packed packet headers option is used.

Termination of the arithmetic coder allows the detection of transmission errors. If the contexts are reset after each coding pass, the decoder can detect an error, discard the coding passes affected by errors, and continue decoding. This strongly limits the scope of transmission errors, and does not excessively increase coding efficiency. Since the coding pass is the basic data unit that can be discarded, in error-prone environment coding passes should be "as small as possible" without limiting coding efficiency. This implies that using smaller code-blocks with respect to the error-free case will generally yield improved performance.

F.3 JPEG 2000 decoder implementation guidelines

In this section we present some guidelines for an implementation of a JPWL compliant encoder. The description of the process is graphically shown in Figure F.1. In particular, the basic actions to follow are:

- JPWL parameters acquisition
- JPEG2000 part 1 coding
- Introduction of the desired JPWL markers. In particular:
    - EPC Marker
        - Write marker (0xFF68), store the position and jump 8 bytes
        - Read JPWL parameters and write Pepc
        - If EPB is used, write Pepbs
        - Jump back after the marker
        - Compute the marker segment length and the codestream length
        - Compute and write CRC 16-CCITT
    - EPB Section
        - Write marker (0xFF66)
        - Determine EPB type and protection parameters for the first part of EPB data
        - Write LDPepb, Depb, Pepb
        - Compute marker segment length
        - Store the first part of data to protect and compute RS(n1,k1)
        - If CRC is requested, compute it on data stored in a buffer
        - If a RS(n2,k2)is requested, compute it on data stored in a buffer
    - ESD Section
        - Write marker (0xFF67) and the Cesd and Pesd parameters
        - Compute Δ-MSE, PSNR, and Δ-PSNR from the distortion value of every packet
        - Determine the metric to use
        - Determine data representation mode (packet, byte range, or packet range mode)
        - Compute error sensitivity values in the chosen metric and data representation mode
        - Compute marker segment length
- Update pointing structures (Psot field, ESD data in byte range mode, etc.)

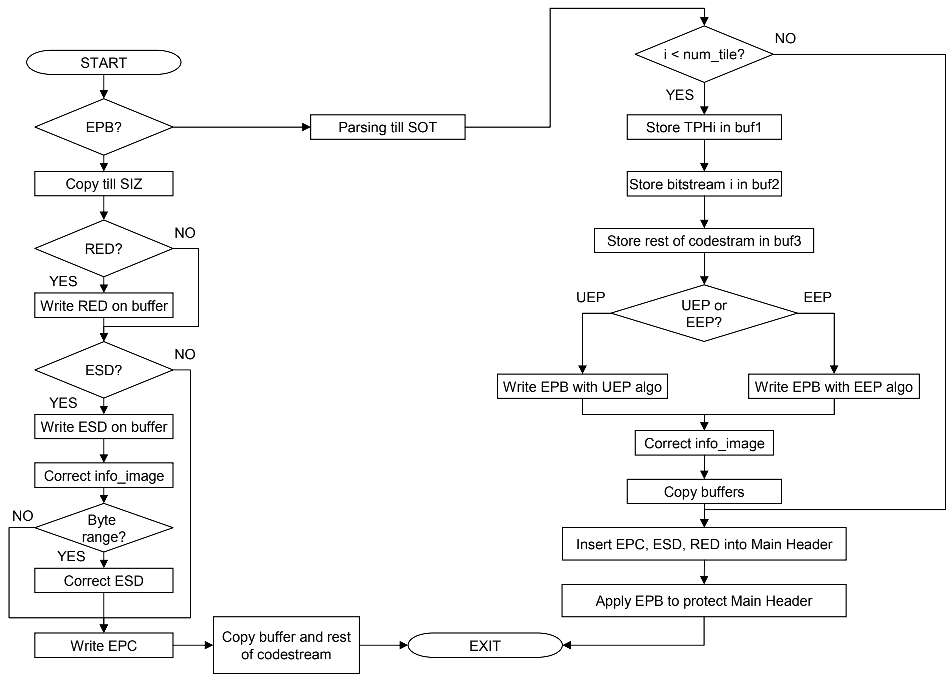

Figure F.1 – JPWL encoding procedure guidelines.

Annex G: Recommended decoder error handling behavior

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

G.1 Introduction

If JPEG 2000 Part 1 error resilience tools are present in a codestream, the decoder has to use them properly. This section is intended to define a recommended behavior for JPEG 2000 Part 1 decoders as well as JPWL decoders, in presence of errors.

G.2 JPEG 2000 Part 1 decoder recommended behavior

G.2.1 ISO/IEC 15444-1 Codestream truncation

In cases where the end of the codestream is unavailable due to transmission errors or losses, the decoder is required to decode as much information as it can, as specified in JPEG 2000 Part 1, annex A.4.4.

Similarly, if a packet is lost within the middle of the codestream, the information in the following packets may not be usable. Nonetheless, the decoder should decode the codestream at least up to the lost packet. Actually, all the code-blocks which have already been included in a previous packet may be de-synchronized. Therefore, no further information should be added to these code-blocks. Data from code-blocks that have not already been included in a previous packet may be recovered properly if the corresponding tag trees are correctly detected.

G.2.2 ISO/IEC 15444-1 Codestream segmentation

Codestream is segmented into Main header, Tile-part header, packet headers and entropy coded data.

There is no specific tool in JPEG 2000 Part 1 either to detect or correct errors in the Main header. A standard JPEG 2000 Part 1 decoder will eventually crash in presence of errors in the Main header. No specific behavior can be defined. Reader interested in Main header protection has to refer to JPEG 2000 Part 11.

There is no specific tool in JPEG 2000 Part 1 to protect the Tile-part headers. However, if it can be stated that a given Tile-part header is erroneous (for example, SOD not detected properly due to an inconsistency in the header), the decoder may jump to the next tile part header by scanning the codestream for SOT marker.

There is no specific tool in JPEG 2000 Part 1 to protect the content of packet headers. However, there are tools dedicated to the prevention of codestream de-synchronization. If SOP and EPH, and/or PLM/PLT markers are present, the decoder may check the consistency with the decoding process. During the decoding of a packet header, if the EPH marker is not detected at the expected location or if the length of the packet found while decoding the packet header is not consistent with the length indicated by PLM/PLT markers, then the packet may be considered as erroneous and dropped. The SOP marker and/or PLM/PLT markers are then used to re-synchronize on the next packet. In any case, the decoder should decode as much information as possible.

G.2.3 Use of ISO/IEC 15444-1 entropy coding options

Once the codestream has been properly segmented, a certain number of options allows for better entropy coded data error resilience.

More particularly, the predictable termination associated to the termination on each coding pass and the segmentation symbols can be used to detect and to locate errors.

In case of the detection of an error by the predictable termination mechanism, the decoder should :

- If the termination on each coding pass and the segmentation symbols are not used, drop the whole block.

- If the termination on each coding pass is used, decode up to the last correctly decoded termination, i.e. up to the pass before the one for which the error was detected.

- If the segmentation symbols are used, decode up to the last correctly decoded segmentation symbol, i.e. skip the last bitplane of the block.

- If both termination on each coding pass and segmentation symbols are used, decode up to the last correctly decoded termination.

In case of the detection of an error by the segmentation symbols mechanism, the decoder should :

- If the predictable termination and the termination on each coding pass are not used, or if only predictable termination or termination on each coding pass is used, decode up to the last correctly decoded segmentation symbol, i.e. up to the bit-plane before the one for which the error was detected.

- If both predictable termination and termination on each coding pass are used, decode up to the last correctly decoded termination, i.e. up to the pass before the one for which the error was detected.

At the encoder side, it is obviously recommended to combine the predictable termination and termination on each coding pass options. As specified in JPEG 2000 Part 1, the segmentation symbols and predictable termination / termination on each coding pass options can be used separately or combined together.

Other error resilience options (Context reset and bypass) are intended to limit entropy decoder desynchronization in case of errors. No specific behavior is defined in this Annex.

G.3 JPWL decoder implementation guidelines

In this section we present some guidelines for an implementation of a JPWL compliant decoder. The description of the process is graphically shown in Figure G.1. In particular, the basic actions to follow are:

- Synchronization with the marker segment EPC
- Read EPC
  - Read Lepc, Pcrc, CL
  - Read Pepc and flag usage of JPWL tools
  - Store ID structures and create an array with Pepb fields, useful for EPB decoding
  - Check CRC and flag presence of errors to the calling function
- Read EPB
  - EPB parameters correction (RS decoding)
  - Read Lepb, Depb, LDPepb, Pepb and check it against the Pepb stored in EPC
  - Determine packed/unpacked mode
    - Correct following data with RS decoding
    - Flag erroneous data with CRC decoding
    - Store residual error locations
- Write RED
  - Go to end of Main Header and store codestream until EOC
  - Write RED parameters and RED data
    - Copy structure generated while decoding EPB's
    - Correct locations with offset originated by adding RED in the codestream
  - Write the rest of the codestream
- Read ESD
  - Go to SOC and start codestream parsing

- For each ESD marker encountered
    - If byte-range mode, then correct sensitivity level locations with Lred
    - Read ESD parameters and ESD data
- Optionally, create an “esdmap” file

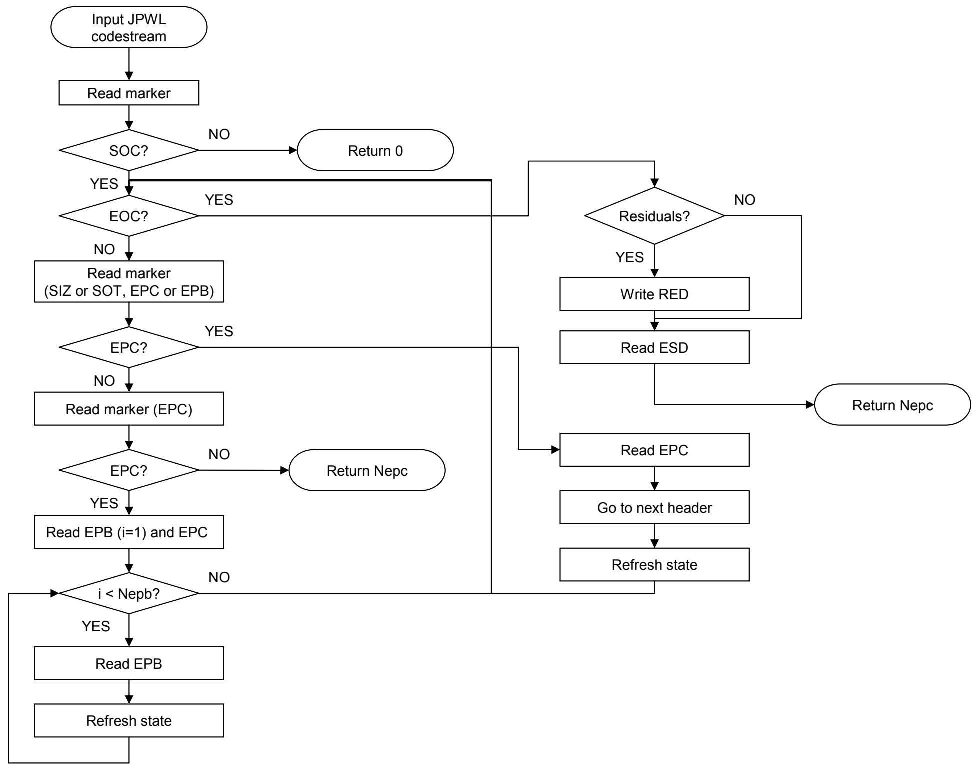

Figure G.1 – JPWL decoding procedure guidelines

Annex H: Error Resilient Entropy Coding

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

In this annex a JPEG 2000 Part 11 error resilient tool at entropy coding level is described. Note that all the techniques here described use the terminology and the assumptions introduced in JPEG 2000 Part 1 Annex C on Arithmetic Entropy coding.

H.1 Introduction

Entropy coding, and specifically arithmetic codes are particularly sensitive to bit errors. Indeed, due to the memory inherent to the technique, a single flipped bit may cause de-synchronization of the decoder. Hence all the remaining symbols can be erroneous. Moreover, in the JPEG 2000 Part 1 Entropy coding case, erroneous symbols can cause unpredictable behavior in the coding passes e.g. erroneous context generation and coefficient modeling, thus deeply impairing the decoded image quality. Therefore, even raw encoded data using the JPEG 2000 Part 1 bypass entropy coding option is concerned by error propagation.

In JPEG 2000 Part 1 some error resilience tools have been designed in order to cope with the intrinsic entropy coder error sensitivity. These techniques are based on encoder termination, segmentation and resynchronization markers, which allow the JPEG 2000 decoder to implement error detection strategies; as a consequence, the standard decoder has the ability to skip the erroneous sections of the bitstream so as to avoid the propagation of bit errors at the image level. This approach can be viewed as a concealment technique, that is generally able to cope with transmission channel with moderate bit error rates. In presence of very harsh transmission conditions, such as in the wireless environment, the employment of powerful error correction techniques turns out to be essential.

In this Annex, a modified arithmetic entropy coder is defined, with extended features namely the soft resynchronization markers and the forbidden symbol [10][11], that allow the implementation of error correcting strategies at the code-block level, thus greatly improving the received image quality with respect to the standard concealment approach.

H.2 Syntax

EPC (Error Protection Capability marker) is used for the specification of the coding parameters as specified in Table A-1. An ID=2 is allocated to the error resilient arithmetic coding technique.

The associated P_{ID} in the EPC marker field is composed by a variable number of 16 bits words that represent the entropy coding parameters associated to each code-block (see Table A-2). The code-block order is as specified in JPEG 2000 Part 1 Annex B. The first byte in P_{ID} is the Forbidden Symbol Parameter (FSP), whereas the second one is the Soft Synchronization Parameter (SSP). It is not required to specify the P_{ID} parameter for all code-blocks. The last (FSP, SSP) pair applies to all the remaining code-blocks. As an example, a single pair specifies the parameters for all the codestream.

Table H.1 – EPC marker segment fields for error resilient entropy coding

EPC marker field	Size in bits	Contents
ID	16	0000 0000 0000 0010
L_{ID}	16	Length of the following P_{ID} parameter
P_{ID}	Variable	A concatenation of parameters pairs (FSP, SSP)

Table H.2 – P_{ID} parameters for error resilient entropy coding

Error resilient arithmetic coding parameters	Size in bits	Contents
FSP	8	0000 0000 – 1111 1010
SSP	8	xxxx xabc

H.3 Binary Encoding with Forbidden Symbol

Binary Encoding with Forbidden Symbol is based on the Arithmetic entropy coding with forbidden symbols (MQF) , which stems from the standard JPEG 2000 Part 1 entropy coder.

H.3.1 MQF Probability Interval Subdivision

The probability interval is subdivided in three zones as shown in 0. The first interval corresponds to the forbidden symbol (FS), which is never encoded and serves as an error detection tool. The FS probability is Q_f, and it is represented on a 16 bit word, adopting the same convention used for LPS probability Q_e in JPEG 2000 Part 1 Annex C. The value of the FS probability is available from the FSP parameter in the EPC marker segment. In order to convert the FSP (8 bit) parameter to Q_f (16 bit), the FSP is multiplied by 0x56. In order to evaluate the corresponding decimal probability value, Q_f must be divided by (4/3)*0x8000, as specified in JPEG 2000 Part 1 Annex C. The admissible FSP range is between 0x00 and 0xFA, being FSP=0x00 the default value that guarantees backward compatibility with MQ. Some conversion examples are reported in Table A-3.

Table H.3 – FSP conversion examples.

FSP	Q_f	FS decimal probability
0x00	0x0000	0.000000
0x01	0x0056	0.001968
0x22	0x0B6C	0.066925
0xFA	0x53FC	0.492096

The following encoding intervals are defined:

a) forbidden sub-interval $Q_f \approx A \cdot Q_f$;

b) LPS sub-interval $Q_e \approx A \cdot Q_e$;

c) MPS sub-interval $A - Q_e - Q_f \approx A - A \cdot (Q_e + Q_f)$

In order to use the FS, the standard LPS probability values Q_e (defined in JPEG 2000 Part 1 Table C-2) must be updated according to the following rule, which correspond to multiply the decimal probability values by (1-FS probability)

$$Q_e = Q_e - Q_e * Q_f /(0x8000 * 4/3) = Q_e - (Q_e * Q_f * 3) >> 17$$

Note that the evaluation of previous expression requires to multiply the 16 bit variables Q_e, Q_f with sufficient bit precision. The Q_f value can be defined/overridden at component, tile and layer level and consequently the LPS probabilities table must be kept synchronized.

The coding redundancy introduced by the FS is $R_f = -\log_2(1 - Q_f / 0x8000 * 4/3)$ bit per input symbol. Finally, it is worth pointing out that MQF is fully compliant with MQ in the case Q_f =0x0000.

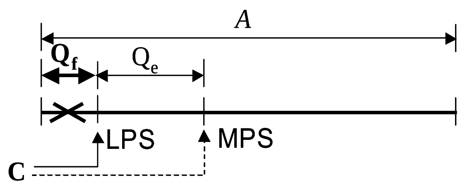

Figure H.1 – MQF probability intervals.

H.3.2 Symbol encoding

The presence of the FS implies a slight modification of the JPEG 2000 Part 1 arithmetic encoding steps. In particular, the CODELPS and CODEMPS procedures must be modified as shown in 0. The grey boxes represent add-on to JPEG 2000 Part 1 arithmetic encoding procedures, the dash boxes represent the point where the JPEG 2000 Part 1 procedure must be used. At each symbol encoding Q_f must be subtracted from the A register, in order to obtain the MPS interval amplitude, and added to the C register, in order to skip the FS interval. The two gray boxes in the figure substitute the one box labeled [A=A-Qe(I(CX))] in Fig. C-6 of [1]. The dashed box indicates that the procedure than continues from the solid box following [A=A-Qe(I(CX))] in Fig. C-6 of [1].

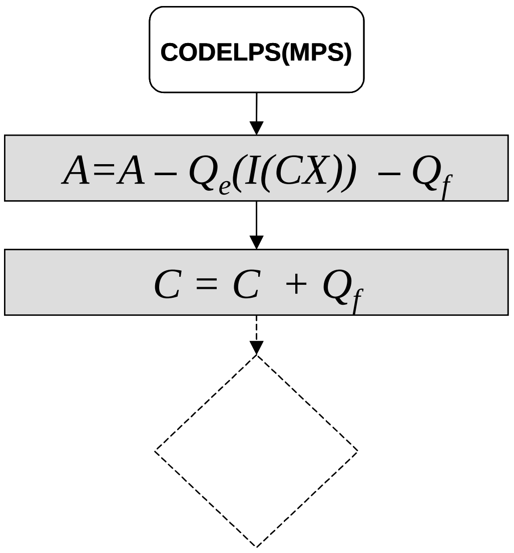

Figure H.2 – MQF coder: CODELPS (MPS) procedures

H.4 Error resilience segmentation symbols

The error detection and synchronization markers addition is defined as follows.

- SEGMARK : original markers addition scheme : 1010 at the end of each bit-plane, i.e. at the end of each cleanup pass, arithmetically encoded with uniform probability.

- SEGMARKPASS : addition of 1010 at the end of the significance propagation and magnitude refinement passes, arithmetically encoded with uniform probability.

- SEGMARKSTRIPE *n* : addition of a marker at the end of each stripe, arithmetically encoded with uniform probability. If *n* is 1 the marker is 10, else if *n* is 2, the marker is 1010. Other values of *n* are not necessary.

The presence of SEGMARKPASS and SEGMARKSTRIPE synchronization symbols is indicated through the SSP marker of Table H.14. The c bit is set to one to indicate the use of SEGMARKPASS option. Its default value is 0. The SEGMARKSTRIPE 2 option is indicated by setting a and b bits to 1. If only one of those two bits is set to one, the SEGMARKSTRIPE 1 option is used. The default value of these two bits is 00. The other bits of the SSP marker are reserved. Examples of SSP values are provided in Table H.14.

Table H.14 – SSP example values

SSP value	option
0000 0001	SEGMARKPASS
0000 0010	SEGMARKSTRIPE 1
0000 0110	SEGMARKSTRIPE 2
0000 0111	SEGMARKPASS + SEGMARKSTRIPE 2

H.5 Error detection

H.5.1 Decoding in presence of errors

The resilience tools described in this annex can be employed along with those adopted in JPEG 2000 Part 1 to provide the decoder with error detection capabilities. The error detection allows one to properly truncate the decoding of erroneous coding passes for a given code-block, thus preventing from error propagation at the transformed coefficients level (see JPEG 2000 Part 1 SectionJ.7). In the following, the error detection strategies based on MQF and error resilience segmentation symbols are described.

H.5.2 MQF error detection

MQF decoding requires that the standard JPEG 2000 Part 1 DECODE procedure is modified as shown in 0 The modified MQF interval evaluation $A = A - Q_e - Q_f$ must be used. FS decoding allows for error detection. In fact, if the received code string happens to fall into the forbidden interval $Chigh < Q_f$, transmission errors are detected and concealment or correction strategies can be adopted. On the contrary, if no FS detection takes place, the C register is moved to the base of the LPS interval $Chigh = Chigh - Q_f$, and standard MQ decoding can be used.

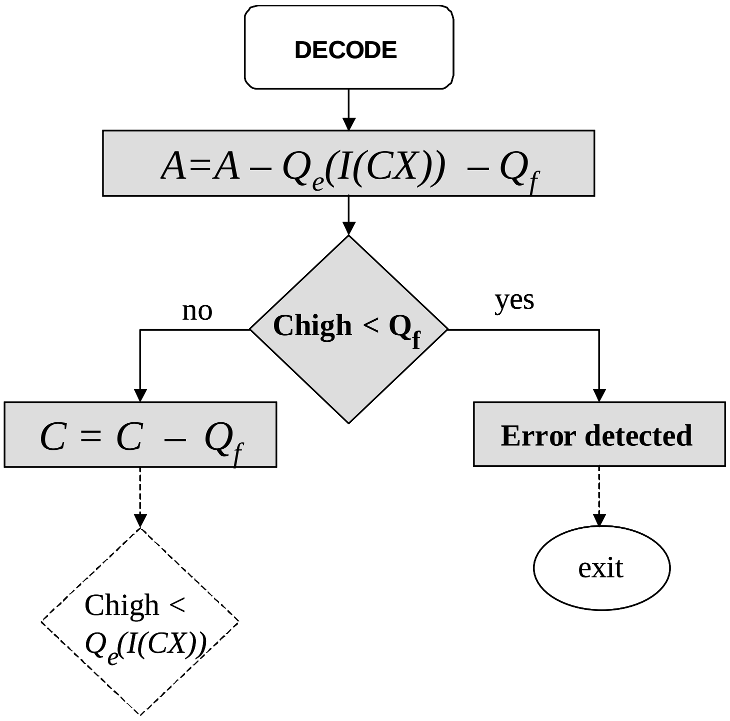

Figure H.3 – MQF decode procedure

H.5.3 Segmentation symbols error detection

The correct decoding of the segmentation symbol confirms the correctness of the decoding up to this point in the codestream. If the segmentation symbol is not decoded correctly, then bit errors occurred and proper countermeasures can be adopted.

H.6 Error correction

The MQF and error resilience segmentation symbols allow to implement a JPWL decoder able to correct bit error at bitstream level.

H.6.1 Bit-clock decoding

The MQ decoder is described in 0. The compressed data **CD** and a context CX are input to the decoder to allow the output of a binary decision D. More precisely, in order to output the i^th^ binary decision d[i] in the code-block, it is necessary to input the associated context CX[i] and a certain number of bits **CD**[n_{i-1}+1;n_i] from the compressed data, where *i* is a generic index representing the scanning order of symbols into the code-block and n_i is the total number of bits that have been read when the decision d[i] has been decoded. The index i is referred to as the *symbol clock* and the decoder is said to be *symbol clock driven*.

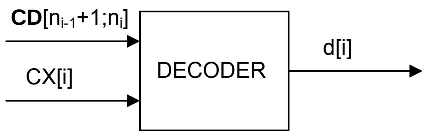

Figure H.4 – Symbol clock based JPEG 2000 Part 1 arithmetic decoder.

For the purpose of error correction, it is convenient to switch to a *bit clock* driven decoder. This change affects only the interface and not the decoder properties. The bit clock based arithmetic decoder module accept as input a single bit CD[n], where n represents the position in the bitstream and a variable number of contexts **CX** corresponding to a variable number of output binary decisions **D**. This new representation is shown in 0, where i_n is the total number of decisions that have been decoded when n bits have been read from the compressed data.

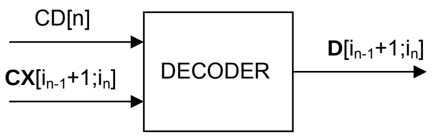

Figure H.5 – Bit clock based JPEG 2000 Part 1 arithmetic decoder.

In order to implement such a decoder, the functions DECODE, RENORMD and INITDEC, shown respectively in Fig. C-15, C-18 and C-20 of JPEG 2000 Part 1 are modified, as described in 0,0 and 0.

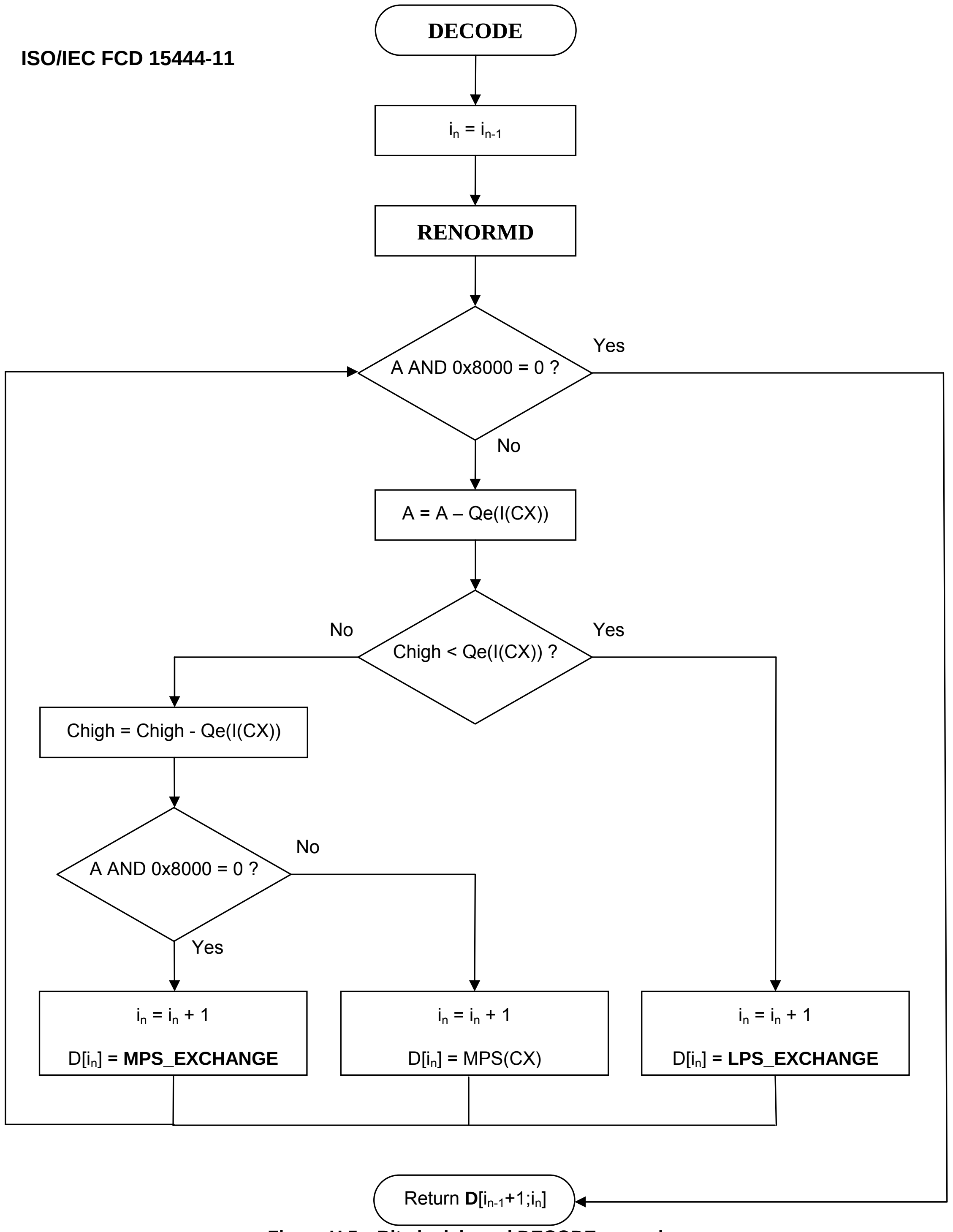
DECODE
$i_n = i_{n-1}$
RENORMD
A AND 0x8000 = 0 ?
Yes
No
A = A – Qe(I(CX))
Chigh < Qe(I(CX)) ?
No
Yes
Chigh = Chigh - Qe(I(CX))
A AND 0x8000 = 0 ?
No
Yes
$i_n = i_n + 1$ D[i_n] = MPS_EXCHANGE
$i_n = i_n + 1$ D[i_n] = MPS(CX)
$i_n = i_n + 1$ D[i_n] = LPS_EXCHANGE
Return D[i_{n-1}+1;i_n]

Figure H.5 – Bit clock based DECODE procedure

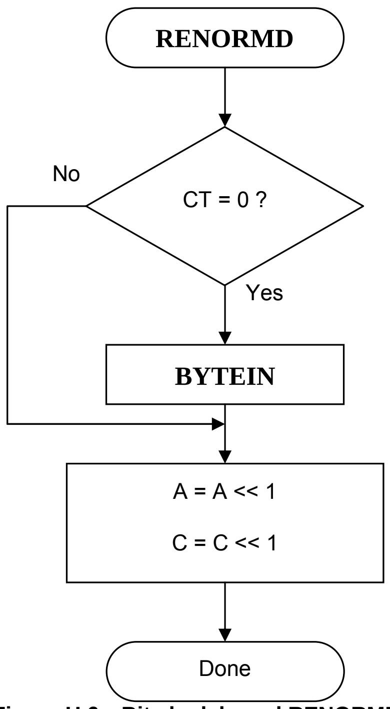

Figure H.6 – Bit clock based RENORMD procedure.

Actually all the possible symbols which can be identified, given the input bit CD[n], are decoded. This approach allows to perform sequential decoding on a bit clock basis. It can be modeled as a state transition automaton described in 0. A state σ[n] may contain all necessary internal state information, for example arithmetic decoder states. The transition between state σ[n-1] and σ[n] is triggered by the bit CD[n]. A variable number of output binary decisions **D** is associated to this transition.

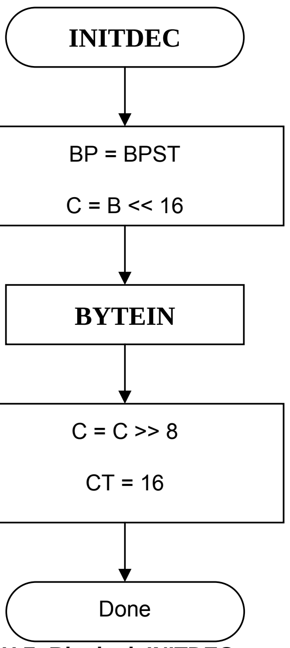

Figure H.7: Bit clock INITDEC procedure

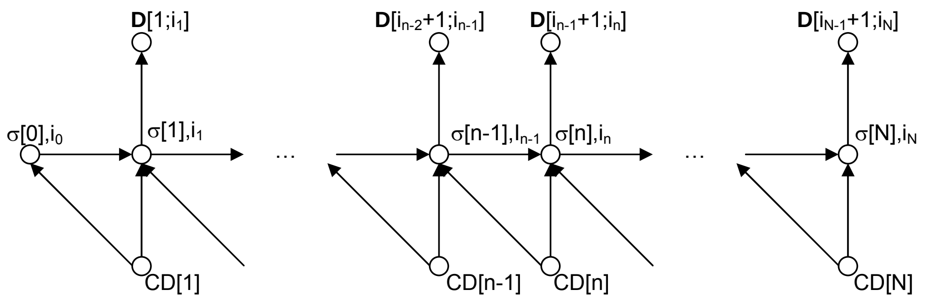

Figure H.8: State transition automaton representation of the decoding process

H.6.2 Bitstream error correction

When the encoded bitstream **CD**, output by the arithmetic coder, is transmitted across a noisy channel the JPWL decoder observes a corrupted version of the bitstream $\underline{\boldsymbol{CD}}$. The error detection tools previously described are used to identify the presence of bit errors. In such a case JPWL decoder attempts error correction by means of bit-clock decoding and sequential search techniques. At each bit depth *n* in the decoding automaton, shown in 0, a set of possible candidate bitstreams $\boldsymbol{CD}_k$ is considered. Some memory space addressed by k is used to store the set of candidate bitstreams. Each bitstream candidate $\boldsymbol{CD}_k$ with its corresponding decoded decisions $\boldsymbol{D}_k$, is ranked according to a proper metric $M_k(n)$, which allows to select the most probable candidate bitstream ***CD***, corresponding to the correct decisions ***D***.

H.6.3 Metrics

H.6.3.1 MAP metric

The *maximum a posteriori* (MAP) probability for the bitstream $\boldsymbol{CD}_k$ with bit depth *n* is defined as

$$P(D_k[1;i_n] \mid \underline{CD}[1;n]) \propto P(D_k[1;i_n]) \cdot P(\underline{CD}[1;n] \mid \mathrm{CD_k}[1;n]).$$

The JPWL decoder uses the following MAP metric

$$M_k(n) = \log\left[P(D_k[1;i_n]) \cdot P(\underline{CD}[1;n] \mid \mathrm{CD_k}[1;n])\right]$$

In presence of a memoriless channel and assuming an order-1 Markov model for the decision bits, the metric $M_k(n)$ can be computed according the state transition automaton as follows:

$$\begin{cases} M_k(0) = 0 \\ M_k(n) = \quad M_k(n-1) + \sum_{j=i_{n-1}}^{i_n} \log\left[P(D_k[j] \mid D_k[1;j-1])\right] + \log\left[P(\underline{CD}[n] \mid \mathrm{CD_k}[n])\right] \end{cases}$$

The term $P(D_k[i] \mid D_k[1;i-1])$ represents the *a priori* probability of the decision bits and it is estimated by means of the binary contextual model of the Coefficient bit modeling, which approximates the LPS probability with the Q_e values, defined by the arithmetic coder. The log of the source model probabilities can be pre-computed and stored in a table in order to speed up the metric evaluation. The term $P(\underline{CD}[n] \mid \mathrm{CD_k}[n])$ represents the channel transition probability. Clearly, the defined metric requires the definition of a channel model, whose state must be available at the receiver; nevertheless, when this information is not available, the simplified metrics described in sections H.6.3.2, H.6.3.3, can be used.

H.6.3.2 Hamming distance

The Hamming metric is defined as the Hamming distance between the received ***CD*** and the candidate ***CD**_k* bitstreams. The Hamming additive metric is defined as $M_k(n) = M_k(n-1) - \underline{CD}[n] \oplus CD_k[n]$, where $\oplus$ represents modulo 2 summation.

This simple distance metric can be employed when the bitstream is transmitted across a binary input/ binary output channel, and no feedback information (channel model, bit error rates, ecc.) is available at the decoder.

H.6.3.3 Euclidean distance

The Euclidean metric can be used when the bitstream is transmitted across a channel with binary input and real output. In this case, the additive metric is $M_k(n) = M_k(n-1) - \left|\underline{CD}^S[n] - \mathrm{soft}(CD_k[n])\right|$, where $\underline{CD}^S[n]$ is the received soft value corresponding to bit $CD[n]$, and $\mathrm{soft}(CD_k[n])$ is the transmitted soft value corresponding to bit $CD_k[n]$.

H.6.4 Sequential search example

In this section an example of sequential search approach is described. The sequential search is based on the decoding tree shown in 0. Each node in the tree represents a bitstream candidate ***CD_k***, decoded up to the bit depth *n*. For each depth a maximum number of candidates MEM are stored for future recursions. At each iteration, all the stored candidates are extended one bit forward. In case of error detection the candidate is pruned (see ***CD_3*** at bit depth *n*=2 in 0). On the contrary, as long as the candidate bitstreams appear as correct, the decoding metrics *$M_k(n)$* are updated and only the best MEM candidates are stored for the next iteration. When the maximum bit depth for the current bitstream is reached, the best candidate in terms of the decoding metric is considered as the most likely bitstream ***CD***.

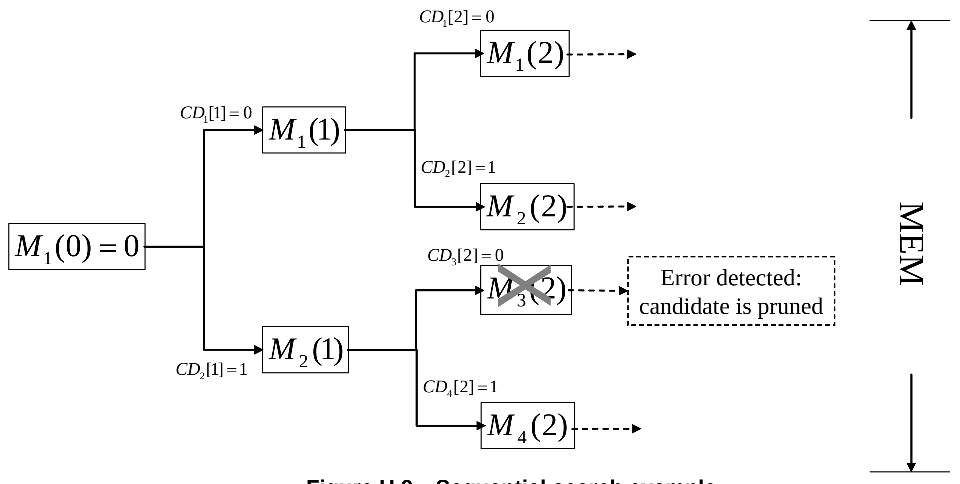

Figure H.9 – Sequential search example.

Annex I: Unequal Error Protection

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

I.1 Introduction

The objective of this Annex is to explain how it is possible using the normative tools of JPWL to apply Unequal Error Protection (UEP) on a JPEG 2000 codestream. This UEP can take advantage of the Error Sensitivity Descriptor information, for selecting the most appropriate technique for protecting the different parts of a JPEG 2000 codestream. UEP can be applied with different manners, inside the codestream thanks to the flexible structure of the Error Protection Block, or it can be done by separating the JPEG 2000 codestream in different parts, each of them being protected differently, and sent to different error prone environments.

I.2 Use of Error Sensitivity Descriptor as input information to Unequal Error Protection systems

The Error Sensitivity Descriptor, by signaling the different error sensitivity of a JPEG 2000 codestream allows to try to select the most appropriate technique for protecting each of the different parts. The most important parts of the codestream are then protected with a larger redundancy than the less important parts of the codestream. This error protection can be applied by a process outside the scope of this Recommendation | International Standard, or using the Error Protection Block as defined in the next section.

I.3 Use of Error Protection Block (EPB) for Unequal Error Protection

The LDPepb parameter of the EPB marker segment that may be present in a Tile-part header, can address data which are outside the Tile-part header bound. This allows to include the JPEG 2000 bitstream in the error protection data range, including or not the packet headers, depending on the use of packed packet feature of JPEG 2000 Part 1.

The Pepb parameter of each EPB marker segment can be used to describe what is the error correction technique used to protect the different part of the bitstream. Each successive EPB marker segment can use different Pepb configuration, being a selection of a code within the same error correction code family or to describe the use of several techniques. As a matter of fact each EPB marker segment redundant data allow protecting differently to errors the different parts of the bitstream they refer to. In the example given in Figure I.1, EPB0 is protecting the Tile-part header marker segments, and EPB1 to EPBn are protecting parts L1 to Ln of the bitstream.

The predefined codes as well as the default codes can be used for this purpose. If other error correction code are used, they must be signaled in the EPC marker segment.

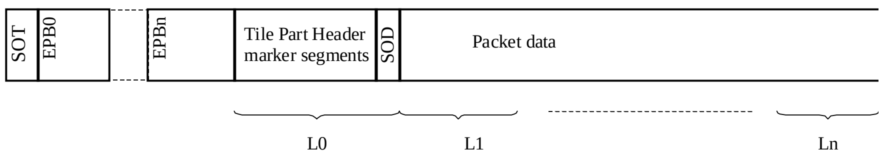

Figure I.1 – Use of EPB for Unequal Error Protection

Annex J: Interoperability with ISO/IEC 15444

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

J.1 Interoperability with ISO/IEC 15444-1

JPWL tools are fully backward compatible with JPEG 2000 Part 1 in the sense of backward compatible and backward compatible with extensions as defined in section 3 of this International Standard | Recommendation

J.2 Interoperability with ISO/IEC 15444-3

All the JPWL tools, acting at the codestream level, can be used for enhancing the robustness of Motion JPEG 2000 in the presence of errors. A possible use of JPWL is to protect each of the individual codestreams.

J.3 Interoperability with ISO/IEC 15444-8 (JPSEC)

Secure JPEG 2000 or JPSEC (ISO/IEC 15444–8) extends the baseline JPEG 2000 specification to provide a standardized framework for secure imaging. This framework enables the efficient integration and use of the tools needed to secure digital images, such as content protection, data integrity check, authentication, and conditional access control. The framework is open and flexible, hence ensuring a straight path for future extensions.

JPSEC enables the use of security tools supporting a number of security services, including:

- confidentiality,
- integrity verification,
- source authentication,
- conditional access,
- secure scalable streaming and secure transcoding,
- registered content identification.

JPSEC defines two marker segments: SEC and INSEC.

The SEC marker segment is present in the Main Header and is mandatory. It gives overall information about the security tools which have been applied to secure the image. More specifically, SEC indicates the JPSEC tools used to secure the image, along with some parameters referring to the technique used. Among other things, these parameters can indicate which parts of the codestream have been secured.

The INSEC marker segment provides with an additional mean to transmit parameters for one of the security tools declared in SEC, in order to complement the information in Main Header. It can be placed in the codestream data and is optional. It uses the fact that the arithmetic decoder in JPEG 2000 stops reading bytes when it encounters a termination marker (i.e. two bytes with a value greater than 0xFF8F).

J.3.1 General Relationship between JPWL and JPSEC

The combination of JPWL and JPSEC is required whenever JPEG 2000 images need to be secured and transmitted over an error-prone wireless channel.

At the transmitter side, JPWL error sensitivity is typically generated during JPEG 2000 encoding. JPSEC tools are then applied to the codestream in order to secure it. Finally JPWL encoding tools are used to make the codestream more robust to transmission errors.

At the receiver side, JPWL decoding tools are first applied to correct possible transmission errors. During this step, JPWL may also generate residual errors information. Finally, JPSEC tools are applied in order to fulfil the selected security services

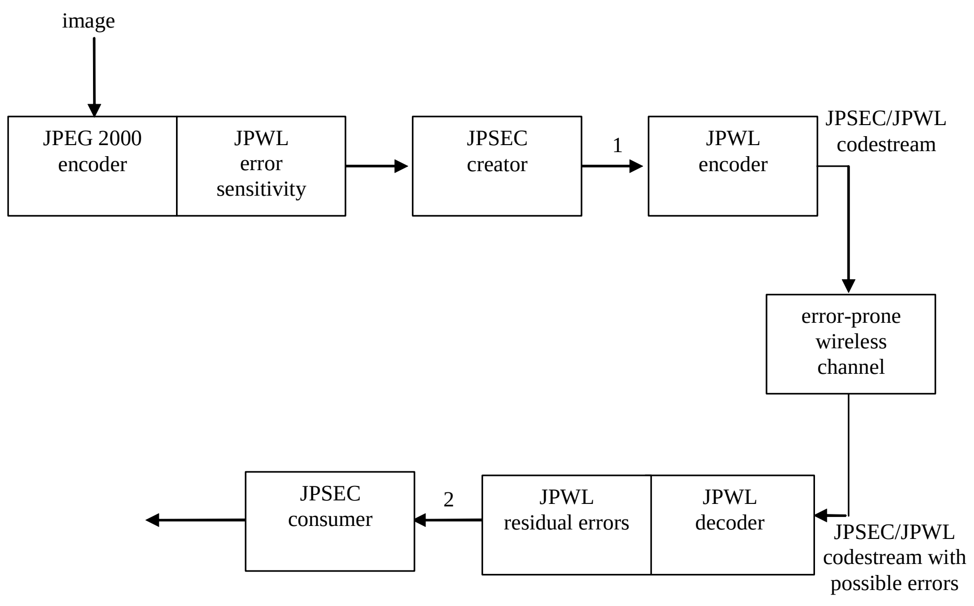

Figure J.5 — Typical JPWL and JPSEC combination

J.3.2 Specific Issues on interoperability between JPWL and JPSEC

A number of issues have to be considered for interoperability between JPWL and JPSEC, as detailed hereafter:

1. JPWL Error Protection Capability (EPC): the presence of this marker segment will affect byte ranges. Note that this marker segment is mandatory in a JPWL codestream.

2. JPWL Error Protection Block (EPB): this marker segment is typically added as the last step at the transmitter and removed as the first step at the receiver. In principle, it should not affect JPSEC.

3. JPWL Error Sensitivity Descriptor (ESD): this marker segment is typically added during JPEG 2000 part 1 encoding, in which case it will be transparent to subsequent JPSEC operations. However, JPSEC could adversely affect the use of ESD in JPWL. In particular, JPSEC should not change byte ranges whenever ESD uses byte ranges. In addition, the JPSEC operations should not affect distortion values; otherwise the information carried by ESD becomes irrelevant. In the latter case, the JPSEC creator has the option to remove the ESD marker segment.

4. JPWL Residual Error Descriptor (RED): this marker segment can be inserted after JPWL decoding. It may therefore affect JPSEC byte ranges. It may also impact JPSEC integrity authentication techniques. In case of a corrupted codestream, the RED information can be useful for a JPSEC consumer to appropriately handle it.

5. JPSEC SEC: the presence of this marker segment will affect byte ranges. Note that this marker segment is mandatory in a JPSEC codestream.

6. JPSEC INSEC: the presence of this marker segment will affect byte ranges. Note that this marker segment appears in the codestream data.

In the case when there are no residual errors, the JPWL encoder and decoder should ideally be transparent. In other words, in this case, the streams at points 1 and 2 in the above figure should be strictly identitical.

As a general recommendation, when used in combination with JPWL, it is preferable for JPSEC to use byte ranges beginning after SOD marker in order to minize problems with byte ranges. In addition, it is preferable to restrict the presence of JPWL marker segments to the Main header and to avoid their presence in the Tile-part headers.

Annex K: Registration Authority

(This Annex is informative only and is not an integral part of this Recommendation | International Standard.)

K.1 General introduction

This description makes reference to JPWL but can be extended to any other JPEG 2000 existing or new work item.

As previously stated, JPWL addresses a number of techniques from which use cases can be defined and applications developed. In order to have a reference site where information about the different JPWL items is available, it has been decided to create a Registration Authority (RA). JPWL items refer to tools for error resilience.

The JPWL RA should allow for registration, identification and dissemination of essential information about JPWL items. A central authority is the proposed solution to make sure that each of the items has a unique identifier linking to the registered descriptive information.

While the proposed solution is based on a unique RA, several intermediate sites interfacing the unique RA may also exist, in particular for linguistic and/or character set usage purposes. However, in this case an automatic translation will have to make it possible to submit the registration with the same conditions as if the submission was directly addressing the unique RA.

K.2 JPWL registration authority

A number of error resilience tools can be used in the JPWL specifications, including the following classes of techniques:

- Unequal Error Protection (UEP)
- Robust entropy coding
- Data partitioning
- Data interleaving

Temporary numbers to be defined can be allocated to items undergoing the registration process if and when applicable. These values may be used until assignment of a permanent JPWL item identification number from the JPWL Authority.

K.3 Purpose of JPWL tool registration

The registration mechanism has been defined to allow for well defined methods used in various JPWL operations; it does not allow for registration of application specific methods and/or technologies. Its purpose is also to allow for addition of other standardized JPWL methods to the initial list above. The addition of new JPWL items will be strictly controlled. Applicants may suggest standardized technologies to be included in the JPWL reference list.

Note that the usage of JPWL items is signaled with a JPWL marker segment present in the codestream. The registration process will create a link to the technology provider in order to obtain further information. When an application finds an unknown JPWL ID, it can query the JPWL RA and get all registered information about this item. When applicable it can also get the appropriate tool to exploit the particular function of the JPWL item. In some use cases, this can lead to automatic operation.

K.4 Criteria

A proposed new JPWL item must meet the following criteria:

- Unique - it must not duplicate a JPWL item already defined.
- Correct - a syntactically and technically correct submission along with all appropriate explanations must be submitted.
- Useful - the new JPWL item should demonstrate usefulness to the user and give examples of use cases when relevant.

Upon fulfiling these criteria, the new JPWL item will be allocated an identification (ID) number and considered as referenced. The ID number can then be used for signaling in the JPWL codestream.

K.4.1 Contents of the submission

The submission should conform to a normative form and make any possible reference to existing standards document describing similar processes.

The submission may also include an informative section that describes the reason for inclusion of this new JPWL item. It should also explain and demonstrate proper usage of this new JPWL item in the JPEG 2000 environment.

K.4.2 Normative description

The normative form has three different parts:

- Identification of the registrant, technology provider, owner or right holder

- Identification of the new JPWL item

- Automatically generated registration data.

K.4.3 Example of new JPWL item registration documents

XYZ unequal error protection algorithm

Identifier: 275 (hex 0113)

Part 1 : identification of the registrant

Name (mandatory)

Company, postal address (optional)

Email link (mandatory)

Occupation, title … (optional)

Part 2 : identification of the item

Registrant identification (mandatory)

Category (optional)

IPR status, e.g. owner, right holder, mandated dealer… (mandatory)

Title or name (optional)

Short technical description (optional)

Operational example use case description (optional)

Parameters table, number and specification, including possible values (optional)

Restrictions of use (optional)

Guidelines for optimum usage (optional)

Downloads when relevant (optional)

Additional comments, motivation, references … (optional)

Part 3 : automatically generated registration data

Identifier

Time stamp

Version when relevant

Last modification when relevant

Part 4 : additional registration information

Special access control conditions

Validity (temporary or final version)

Update and erase procedure (profiles)

Email confirmation

K.5 Operating conditions for the Registration Authority - Submission, review and appeal process

This section provides requirements for the submission and approval of new or updated JPWL item as defined previously. All the registration processes have a common set of requirements for the registrants as well as requirements and processes based on the category of submission, as described in detail below.

K.5.1 Submission process

The registration submission process conforms to the specifications described above in this document and is graphically shown in Figure 6 below, showing the process flow for submission of a new item to the JPWL Authority. This process flow is focused on access based on an Internet submission and approval system, but could be applied to other (non-electronic) processes going for example through the Authority Administrator assistance.

The registrant (individual or institution) shall submit a request for registration of a new item to the JPWL Authority, possibly via Internet access. The first operation is the identification of the registrant, which takes the steps shown on the block diagram: either the registrant is unknown to the JPWL Authority, or he has previously registered an item and is listed in the JPWL Authority registers. If it is the first time the registrant provided a submission to the Authority, he must complete a general identification form and get a personal password for access controlled operations.

In order to provide some level of security to the registration process, a password is set by the registrant and must be used every time the registrant contacts the JPWL Authority for an active operation (this excludes consultation of the contents but concerns registration and update of registered material). Only the registrant will be allowed modification of the password and all the registered information regarding his personal identification (name, company, etc.). The registrant may also request, under certain conditions to be defined by the Authority and registration submission form, that part of the input information remains undisclosed.

It is possible to modify the submitted content. For example, the registrant may want to correct the information in the submission form. However, it is highly desirable that the modified item is backward compatible with the original submission. A modification can be made directly by the registrant under his own responsibility. Alternatively, a modification can be made by a third party; in this case the modified submission must be approved by either the registrant or the administrator of the RA.

K.5.2 Review process

Following validation of the individual’s right to access the Authority’s Registration System, the actual registration of the desired item begins. The JPWL Authority verifies (manually or automatically) the conformity of the request to the specifications, and notifies the registrant of a positive or negative response to the registration request. The Authority must respond within one month to the registrant’s request. If no response is received within one month, the probationary period shall begin.

The response of the JPWL Authority shall be positive if the item has not been registered (i.e. is new and unique), and if the contents of the submission conforms to the applicable forms. This positive response, however, is not final. It does allow the submitter to implement the proposed item for a probationary “validation” period to be defined by the Authority. This probationary period may last for three to six months, for example. It is during this time that the content of the submission is made accessible to the public for comments. This will be done via the direct link available from the JAWS web site, and will be coordinated and validated by the Expert technical advisory group.

During the validation period, the registration is temporary, and the JPWL Authority may require more information from the registrant regarding the submission. During this period, omissions or errors in the submission, lack of implementation support, or related problems, may cause the rejection of the submission. However, if after the probationary period no inconsistencies or concerns are raised, the submission shall be considered accepted and formally approved by the Authority and will be accordingly disseminated on the public register. Any individual reviewing the public register shall be clearly informed if a submission has successfully completed its probationary period based on a “status” field or similar notification method.

The response of the JPWL Authority may also be negative if the submission is not acceptable as received, due to errors, inconsistencies or technical concerns with the proposed content. Examples of why the Authority would reject a submitted item include:

- If an approved, registered item already exists that contains the identical content of the submission.
- The Authority considers that there is not enough originality in the proposed item which could easily be implemented with an existing, approved item.
- If the submission contains errors or is not compliant with the specifications or standard it is based on.

K.5.3 Notification and appeal process

A negative response may be appealed if the registrant believes that there was an error made in the rejection, or when further information is required to clarify issues or concerns. If the registrant requires additional review beyond the Authority’s process, he may submit his case for review by ISO/IEC JTC1 SC29 WG1 Committee at the next appropriate WG1 group meeting. He may then be required to provide additional information at the request of the experts, who, under the authority of WG1, will provide a final, definitive response of acceptance or rejection. A refused item may still be used by the registrant, but it will not be allowed to claim standards compliance and hence, may become de facto a proprietary item. In order to have a rejected item reviewed by the WG1, the registrants must re-submit the proposal through their National Body, specifying why the submission requires consideration by WG1.

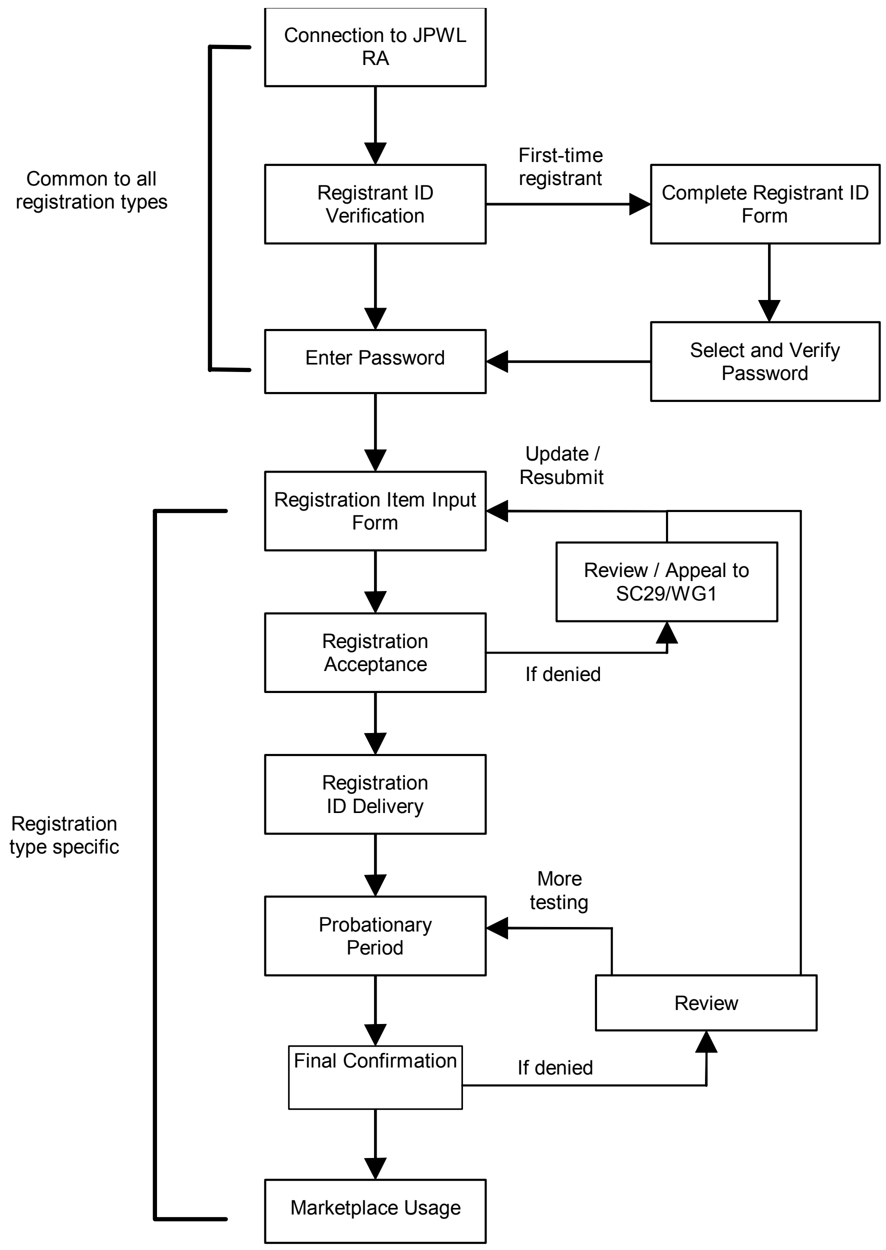

Figure 6 — JPWL RA Registration Process

Annex L: Patent Statement

(This Annex is informative only and is not an integral part of this International Standard.)

There is the possibility that, for some of the processes specified in this International Standard, conformance or compliance may require use of an invention covered by patent rights. By publication of this 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 J-15 Received intellectual property rights statements

Number	Company
1	Thales
2	INRIA
3	
4	

Bibliography

[1] ISO/IEC 15444-1 / ITU-T T.800 JPEG 2000 image coding system Part 1

[2] Poulliat C., Vila P., Pirez D., Fijalkow I., “Progressive JPEG2000 Image Transmission over noisy channel”, *Eusipco 2002*, Toulouse, France, 3rd-6th September 2002.

[3] Moccagatta I., Soudagar S., Liang J., and Chen. H., ”Error-Resilient Coding in JPEG-2000 and MPEG-4”, *IEEE Journal on Selected Areas in Communications* , Vol. 18, No. 6, pp. 899-914, June 2000.

[4] Hagenauer J., “Rate-Compatible Punctured Convolutionnal Codes (RCPC Codes) end their applications”, *IEEE Transactions on Communications*, Vol. 36, No. 4, pp. 389-400, April 1988.

[5] Morelos-Zaragoza R. H. , Fossorier M. P.C., Lin S., Imai H., “Multilevel Coded Modulation for Unequal Eror Protection and Multistage Decoding - Part I: Symmmetric Constellations”, *IEEE Transactions on Communications*, Vol. 48, No. 2, February 2000.

[6] A. Natu and D. Taubman “Unequal Protection of JPEG 2000 Code-Streams in Wireless Channels”, *Proceedings of IEEE GLOBECOM’02*, vol. 1, pp. 534-538, Taipei, China, 17-21 Nov. 2002

[7] V. Sanchez and M.K. Mandal, “Robust transmission of JPEG 2000 images over noisy channels, ” Proceedings of IEEE ICCE'02, pp. 80-81, 2002

[8] D. Nicholson, C. Lamy-Bergot, X. Naturel and C. Poulliat, “JPEG 2000 backward compatible error protection with Reed-Solomon codes”, IEEE Transactions on Consumer Electronics, November 2003

[9] F.J. MacWilliams and N.J.A. Sloane, The Theory of Error-Correcting Codes, North-Holland: New York, NY, 1977.

[10] M.Grangetto, E.Magli, G.Olmo, "Robust video transmission over error-prone channels via error correcting arithmetic codes", *IEEE Communications Letters*, Vol. 7, No. 12, pp. 596-598, Dec. 2003

[11] T. Guionnet, C. Guillemot, "Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels", *IEEE Transactions on Image Processing*, v. 12, Nr. 12, pp. 1599-1609, Dec. 2003

[12] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling, “Numerical Recipes in C: The Art of Scientific Computing, Second Edition”, Cambridge University Press, Chapter 20, pp. 896-903

[13] F. Frescura, C. Feci, M. Giorni, S. Cacopardi, “JPEG2000 and MJPEG2000 Transmission in 802.11 Wireless Local Area Networks”, IEEE Transactions on Consumer Electronics, November 2003.