ISO/IEC JTC 1/SC 29/WG1 N xxxx
2 November 2012

ISO/IEC JTC 1/SC 29/WG 1
(& ITU-T SG16)

Coding of Still Pictures

JBIG	**JPEG**
Joint Bi-level Image Experts Group	Joint Photographic Experts Group

TITLE:	Text of ISO/IEC WD2 XXXXX-1
SOURCE:	JPEG HDR Ad Hoc Group Thomas Richter, Germany Walt Husak, USA Chaker Larabi, France Touradj Ebrahimi, Switzerland Alessandro Artusi, UK Massimiliano Agostinelli, UK
PROJECT:	ISO/IEC XXXXX-1 (JPEG HD Part 1 – JPEG High Dynamic Rnage Image coding – Specification)
STATUS:	WD1
REQUESTED ACTION:	For WD1 ballot
DISTRIBUTION:	For WG1 review

INTERNATIONAL ORGANISATION FOR STANDARDISATION

INTERNATIONAL ELECTROTECHNICAL COMMISSION

Information technology –

JPEG digital imaging systems integration –

JPEG High Dynamic Range Image Coding Systems – Core Coding System Specification

Draft ITU-T Recommendation | International Standard

CONTENTS

Page

Foreword

This Recommendation | International Standard covers compression and representation of high dynamic range images backwards compatible to ITU.T Rec.81 | ISO/IEC 10918-1. That is, legacy applications conforming to ITU.T Rec. 81 | ISO/IEC 10918-1 will be able to reconstruct streams generated by an encoder conforming to this Recommendation | International Standard, though will possibly not be able to reconstruct such streams in full dynamic range or full quality.

The aim of this standard is to provide a migration path for legacy applications to support, potentially in a limited way, high dynamic range images. Existing tools depending on the existing standards will continue to work, but will only work on a low dynamic range version of the image. This Recommendation | International Standard specifies a coded file format, referred to as JPEG HD, which is designed primarily for storage and interchange of continuous-tone photographic content.

Introduction

This Recommendation | International standard specifies a coded codestream format for storage of continous-tone high and low dynamic range photographic content. JPEG HD is a scalable image coding system supporting multiple component images in bitranges from eight up to XXX bits per sample; the codestream is composed in such a way that legacy applications conforming to ITU.T 81 | ISO/IEC 10918-1 are able to reconstruct a lower quality, low dynamic range, eight bits per sample version of the image.

JPEG HD is primarily designed to represent images of high dynamic range content, even though images of low dynamic range using only eight bits per pixel are supported. The goal is to provide a backwards compatible coding spefication that allows legacy applications and existing toolchains to continue to operate on codestreams conforming to this Recommendation|International Standard.

Today, the most widely used digital photography format, a minmal implementation of JPEG (specified in ITU-T Recommendation T.81 | ISO/IEC 10918-1), uses a bit depth of 8; each of the three channels that together compose an image pixel is represented by 8 bits, providing 256 representable values per channel or 16 777 216 representable color values. For more demanding applications, it is not uncommon to use a bit depth of 16, providing 65 536 representable values to describe each channel within a pixel, resulting on over 2.8 x 10^{14} representable color values. In some less common scenarios, even greater bit depths are used. In scenarios when memory or processing power is at a premium, as few as five bits may be used, providing only 32 representable values per channel.

Most common photo and image formats use an 8-bit or 16-bit unsigned integer value to represent some function of the intensity of each color channel. While it might be theoretically possible to agree on one method for assigning specific numerical values to real world colors, doing so is not practical. Since any specific device has its own limited range for color reproduction, the device's range may be a small portion of the agreed-upon universal color range. As a result, such an approach is an extremely inefficient use of the available numerical values, especially when using only 8 bits (or 256 unique values) per channel.

To represent pixel values as efficiently as possible, devices use a numeric encoding optimized for their own range of possible colors or gamut.

JPEG HD has been designed to be backwards compatible to legacy applications while at the same time have a small coding complexity; JPEG HD uses, whenever possible, functional blocks of ITU.T 81|ISO/IEC 10918-1 to represent high dynamic range images. It is optimized for good image quality and compression efficiency while also enabling low-complexity encoding and decoding implementations. The design objectives include:

- Backwards compatible representation of images of various bitdepths, embedded system friendly compression
  - Small memory footprint
- High compression quality

The algorithm uses a reversible lapped biorthogonal transform. The transform requires at most 3 non-trivial (multiply plus addition) and 7 trivial (addition or shift) operations per sample (with no divisions) at the highest quality level. In the lowest complexity mode, 1 non-trivial and 4 trivial operations per sample are required for transform. The image is processed in 16x16 macroblocks, allowing a minimal memory footprint for embedded implementations.

The algorithm provides native support for both RGB and CMYK by converting these color formats to an internal luma-dominant format through the use of a reversible color transform. In addition, YUV, monochrome and arbitrary n-component color formats are supported.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY –
INFORMATION TECHNOLOGY – JPEG HDR IMAGE CODING SYSTEM:
CODING OF HIGH DYNAMIC RANGE IMAGES

1 Scope

This Recommendation | International standard specifies a coding format, referred to as JPEG HD, which is designed primarily for continuous-tone photographic content using a sample precision of above eight bits per sample.

2 Normative references

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

2.1 Identical Recommendations | International Standards

ITU.T XXX

2.2 Paired Recommendations | International Standards equivalent in technical content

To be determined.

2.3 Additional references

ITU-T Rec. T.81 | ISO/IEC 10918-1

3 Definitions, Abbreviations and Symbols

3.1 Definitions

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

AC coefficient: Any DCT coefficient for which the frequency is not zero in at least one dimension.

binary decision: Choice between two alternatives.

bit stream: Partially encoded or decoded sequence of bits comprising an entropy-coded segment.

block: An 8 × 8 array of samples or an 8 × 8 array of DCT coefficient values of one component.

byte: A group of 8 bits.

coder: An embodiment of a coding process.

coding: Encoding or decoding.

coding model: A procedure used to convert input data into symbols to be coded.

(coding) process: A general term for referring to an encoding process, a decoding process, or both.

compression: Reduction in the number of bits used to represent source image data.

component: A two-dimensional array of samples having the same designation in the output or display device. An image typically consists of several components, e.g. red, green and blue.

continuous-tone image: An image whose components have more than one bit per sample.

data unit: An 8 × 8 block of samples of one component in DCT-based processes; a sample in lossless processes.

DC coefficient: The DCT coefficient for which the frequency is zero in both dimensions.

(DCT) coefficient: The amplitude of a specific cosine basis function – may refer to an original DCT coefficient, to a quantized DCT coefficient, or to a dequantized DCT coefficient.

decoder: An embodiment of a decoding process.

decoding process: A process which takes as its input compressed image data and outputs a continuous-tone image.

dequantization: The inverse procedure to quantization by which the decoder recovers a representation of the DCT coefficients.

discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform.

downsampling: A procedure by which the spatial resolution of a component is reduced.

encoder: An embodiment of an encoding process.

encoding process: A process which takes as its input a continuous-tone image and outputs compressed image data.

entropy-coded (data) segment: An independently decodable sequence of entropy encoded bytes of compressed image data.

entropy decoder: An embodiment of an entropy decoding procedure.

entropy decoding: A lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder.

entropy encoder: An embodiment of an entropy encoding procedure.

entropy encoding: A lossless procedure which converts a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols.

extended (DCT-based) process: A descriptive term for DCT-based encoding and decoding processes in which additional capabilities are added to the baseline sequential process.

forward discrete cosine transform; FDCT: A mathematical transformation using cosine basis functions which converts a block of samples into a corresponding block of original DCT coefficients.

frequency: A two-dimensional index into the two-dimensional array of DCT coefficients.

grayscale image: A continuous-tone image that has only one component.

high dynamic range: An image or image data comprised of more than eight bits per sample.

Huffman decoder: An embodiment of a Huffman decoding procedure.

Huffman decoding: An entropy decoding procedure which recovers the symbol from each variable length code produced by the Huffman encoder.

Huffman encoder: An embodiment of a Huffman encoding procedure.

Huffman encoding: An entropy encoding procedure which assigns a variable length code to each input symbol.

Joint Photographic Experts Group; JPEG: The informal name of the committee which created this Specification. The "joint" comes from the ITU-T and ISO/IEC collaboration.

legacy decoder: An embodiment of a decoding process conforming to ITU.T Rec.T.81|ISO/IEC 10918-1.

lossless: A descriptive term for encoding and decoding processes and procedures in which the output of the decoding procedure(s) is identical to the input to the encoding procedure(s).

lossless coding: The mode of operation which refers to any one of the coding processes defined in this Specification in which all of the procedures are lossless (see Annex H).

lossy: A descriptive term for encoding and decoding processes which are not lossless.

low-dynamic range: An image or image data comprised of data with no more than eight bits per sample.

marker: A two-byte code in which the first byte is hexadecimal FF and the second byte is a value between 1 and hexadecimal FE.

marker segment: A marker and associated set of parameters.

minimum coded unit; MCU: The smallest group of data units that is coded.

pixel: A collection of sample values in the spatial image domain having all the same sample coordinates, e.g. a pixel may consist of three samples describing its red, green and blue value.

point transform: Scaling of a sample or DCT coefficient.

precision: Number of bits allocated to a particular sample or DCT coefficient.

procedure: A set of steps which accomplishes one of the tasks which comprise an encoding or decoding process.

quantization value: An integer value used in the quantization procedure.

quantize: The act of performing the quantization procedure for a DCT coefficient.

residual scan: An additional pass over the image data invisible to legacy decoders which provides additive and/or multiplicative correction data of the legacy scans to allow reproduction of high-dynamic range data.

refinement scan: An additional pass over the image data invisible to legacy decoders which provides additional least significant bits to extend the precision of the DCT transformed coefficients.

sample: One element in the two-dimensional array which comprises a component.

sample grid: A common coordinate system for all samples of an image. The samples at the top left edge of the image have the coordinates (0,0), the first coordinate increases towards the right, the second to the bottom.

scan: A single pass through the data for one or more of the components in an image.

scan header: A marker segment that contains a start-of-scan marker and associated scan parameters that are coded at the beginning of a scan.

table specification data: The coded representation from which the tables used in the encoder and decoder are generated and their destinations specified.

transcoder: A procedure for converting compressed image data of one encoder process to compressed image data of another encoder process.

(uniform) quantization: The procedure by which DCT coefficients are linearly scaled in order to achieve compression.

upsampling: A procedure by which the spatial resolution of a component is increased.

vertical sampling factor: The relative number of vertical data units of a particular component with respect to the number of vertical data units in the other components in the frame.

zero byte: The X’00’ byte.

zig-zag sequence: A specific sequential ordering of the DCT coefficients from (approximately) lowest spatial frequency to highest.

3.2 Symbols

X	Width of the sample grid in positions
Y	Height of the sample grid in positions
Nf	Number of components in an image
$s_{i,x}$	Subsampling factor of component i in horizontal direction
$s_{i,y}$	Subsamplng factor of component i in vertical direction
H_i	Subsampling indicator of component i in the frame header
V_i	Subsampling indicator of component i in the frame header
$v_{x,y}$	Sample value at the sample grid position x,y

3.3 Abbreviations

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

ASCII	American Standard Code for Information Interchange
DC	Lowpass
AC	Highpass
LSB	Least Significant Bit
MSB	Most Significant Bit
HDR	High Dynamic Range
LDR	Low Dynamic Range
TMO	Tone Mapping Operator

DCT	Discrete Cosine Transformation

4 Conventions

4.1 Conformance language

This Recommendation | International Standard consists of normative and informative text.

Normative text is that text which expresses mandatory requirements. The word "shall" is used to express mandatory requirements strictly to be followed in order to conform to this Specification and from which no deviation is permitted. A conforming implementation is one that fulfils all mandatory requirements.

Informative text is text that is potentially helpful to the user, but not indispensable and can be removed, changed or added editorially without affecting interoperability. All text in this Recommendation | International Standard is normative, with the following exceptions: the Introduction, any parts of the text that are explicitly labeled as "informative", and statements appearing with the preamble "NOTE" and behavior described using the word "should". The word "should" is used to describe behavior that is encouraged but is not required for conformance to this Specification.

The keywords "may" and "need not" indicate a course of action that is permissible in a conforming implementation.

The keyword "reserved" indicates a provision that is not specified at this time, shall not be used, and may be specified in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be specified in the future.

4.2 Operators

NOTE – Many of the operators used in this Recommendation | International Standard are similar to those used in the C programming language.

4.2.1 Arithmetic operators

+	Addition
–	Subtraction (as a binary operator) or negation (as a unary prefix operator)
*	Multiplication
/	Division without truncation or rounding.
mod	x mod a is defined as the modulus operator for $x \geq 0$, $a>0$
	x mod a = –((–x mod a)) for $x< 0$, $a>0$

4.2.2 Logical operators

||	Logical OR
&&	Logical AND
!	Logical NOT
$\in$	$x \in \{A, B\}$ is defined as (x == A || x == B)
$\notin$	$x \notin \{A, B\}$ is defined as (x != A && x != B)

4.2.3 Relational operators

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

4.2.4 Bit-wise operators

When operating on integer arguments, bit-wise operators operate on a two's complement representation of the integer value using a number of bits sufficient to represent the integer value (with a bit equal to 0 in the MSB of non-negative integer value representations and otherwise with a bit equal to 1 in the MSB). When interpreting the result of a bit-wise operator as an integer, the result is interpreted as a two's complement representation of the integer value. The following bit-wise operators are defined:

&	AND. When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than the other argument, the shorter argument is extended by adding more significant bits equal to the MSB of the shorter argument.
|	OR. When operating on integer arguments, operates on a two's complement representation of the integer values. When operating on a binary argument that contains fewer bits than the other argument, the shorter argument is extended by adding more significant bits equal to the MSB of the shorter argument.
x >>b	Arithmetic right shift of a two's complement integer representation of x by b binary digits. Bits shifted into the MSBs as a result of the right shift shall have a value equal to the MSB of x prior to the shift operation.
x << b	Arithmetic left shift of a two's complement integer representation of x by b binary digits. Bits shifted into the LSBs as a result of the left shift shall have a value equal to 0.

4.2.5 Assignment operators

=	Assignment operator

4.2.6 Precedence order of operators

Operators are listed below in descending order of precedence. If several operators appear in the same line, they have equal precedence. When several operators of equal precedence appear at the same level in an expression, evaluation proceeds according to the associativity of the operator either from right to left or from left to right.

Operators	Type of operation	Associativity
(), [ ], .	Expression	Left to Right
–	Unary negation	
*, /	Multiplication	Left to Right
mod	Modulo (remainder)	Left to Right
+, –	Addition and Subtraction	Left to Right
< , >, <=, >=	Relational	Left to Right
&, |, ^	Bitwise operator	Left to Right

4.2.7 Mathematical functions

$\lceil x \rceil$	Ceil of x. Returns the smallest integer that is greater than or equal to x.
$\lfloor x \rfloor$	Floor of x. Returns the largest integer that is lesser than or equal to x.
$\lvert x \rvert$	Absolute value, is –x for x < 0, otherwise x.
sign(x)	Sign of x, zero if x is zero, +1 if x is positive, -1 if x is negative.
clamp(x,min,max)	Clamps x to the range [min,max]: returns min if x < min, max if x > max or otherwise x.

5 General

The purpose of this clause is to give an informative overview of the elements specified in this Specification. Another purpose is to introduce many of the terms which are defined in clause 3. These terms are printed in *italics* upon first usage in this clause.

There are three elements specified in this Specification:

a) An *encoder* is an embodiment of an *encoding process*. An encoder takes as input *digital source image data* and *encoder specifications*, and by means of a specified set of *procedures* generates as output *a codestream*.

b) A *decoder* is an embodiment of a *decoding process*. A decoder takes as input a codestream, and by means of a specified set of procedures generates as output *digital reconstructed image data*.

c) The *codestream* is a compressed image data representation which includes all necessary data to allow a (full or approximate) reconstruction of the sample values of a digital image. Additional data might be required that define the interpretation of the sample data, such as color space or the spatial dimensions of the samples.

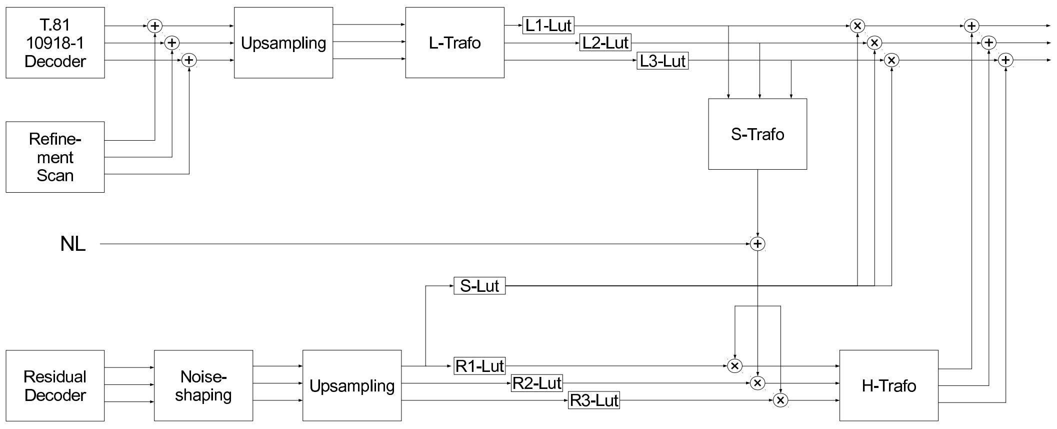

Figure 1 – Overview on the decoding process of high-dynamic range images

Kommentar [tr1]: Behind L, a round to integer and clip operation should be inserted. Behind upsampling in the R-branch, also a round-clip should be added. Does Dolby has an additional color transformation after upsampling? An additional "cast to float" should be added for the final output.

Figure 1 illustrate the general case for which the *continuous-tone* source and reconstructed image data consist of multiple *components*. (A *colour* image consists of multiple components; a *grayscale* image consists only of a single component.) A significant portion of this Specification is concerned with how to handle multiple-component images in a flexible, application-independent way.

5.1 Low and High Dynamic Range Images

Kommentar [tr2]: This description should be extended to include alpha components. Still unclear how to do that.

This Specification distinguishes between low and high dynamic range images. The first type of images consists of unsigned integer samples of a sample precision of exactly eight bits per sample. Such images are represented in a codestream that is compatible to ITU.T Rec. T.81|ISO/IEC 10918-1, even though this Recommendation | International Standard consists only of a subset of the tools available in this earlier Recommendation|International Standard. High dynamic range images consist of samples that are either floating point, or use bit precisions between nine and 16 bits per sample. Decoding of such images requires additional tools defined in this Recommendation | International Standard. These additional tools are implemented in such a way that legacy decoders are still able to decode a low-dynamic range version of the original image, but will not have access to additional data that comprises the original image.

Specifically, this Recommendation | International Standard depends only the DCT coding process defined in Section 4.3 of ITU.T Rec. T.81|ISO/IEC 10918-1, and the Baseline, Sequential and Progressive *Scan Types* defined in Annexes F and G thereof. Additional restrictions apply such as images in this Recommendation | International Standard may only use up to four components and chroma subsampling factors of one and two. The process of chroma subsampling is defined in detail in Annex A.

Kommentar [tr3]: or three?

The additional tools defined here first generate a tone mapped version of the high dynamic range image and encode this image by means already defined in the earlier standard. The encoder also computes a *residual image* between the low dynamic and high dynamic range image and transmits this data, along with all necessary metadata in side channels. The side channel is build on top of *Application Markers* defined in Annex B of the earlier standard, and thus would remain invisible to legacy decoders.

5.2 Functional overview on the decoding process

A high-level overview on the decoding process is given by Figure 1: A low dynamic range representation of the image is represented by a ITU-T Rec.T 81|ISO/IEC 10918-1 compliant codestream is decoded into a low dynamic range representation (top left). An additional scan type, defined in Annex XXX may refine this data in the DCT domain. The resulting samples are then upsampled on a common sample grid. The upsampling process is defined in Annex A. Following that, the output data is processed by an inverse decorrelation transformation, the L-Transformation, which is either the ICT, the FCT or the identity transformation. The inverse decorrelation transformation is defined in Annex C. The ouptut of this transformation is then processed by an inverse tone mapping operation denoted by L1, L2 and L3, resulting in an approximate version of the HDR image. The S-Transformation computes the luminance of the approximate HDR image, resulting in a single value for each pixel. The S-Transformation can be either the forwards ICT transformation, or a null-transformation that ouptuts zero for any input. A noise-floor value, encoded in the JPEG Extensions markers, is added to the luminance value.

The residual image to the high-dynamic range image is represented either by an ITU.T-Rec.T 80|ISO/IEC 10918-1 codestream, or by an additional scan type denoted "Residual Scan" defined in Annex G. The reconstructed residual sample values are optionally processed by an inverse noise shaping algorithm removing quantization noise before performing an upsampling step that is identical to the low-dynamic range upsampling process defined in Annex A. An additional inverse tone mapping step is applied to the residual signals as well, denoted in the figure by R1 through R3. The output luminance and chroma residuals are multiplied by the LDR luminance computed from the S-Transformation, and then undergo a second inverse decorrelation transformation called the H-Transformation. This transformation may either be the inverse ICT, the RCT or the identity transformation, all of which are defined in Annex C.

The residual luma signal is in addition mapped by a third tone mapping operation into a scale value by which the approximate high-dynamic range sample values are multiplied. Finally, the output of the H-transformation is added to the approximate high-dynamic sample values resulting in the decoded image.

5.3 Example Applications

A **purely additive** HDR decoding process can be implemented by setting the S-Transformation to the null-tansform, the noise floor value to one and the R1 though R3 transformations to identity. The S-tonemapper is in this application scenario a constant table that generates the output one. The L1 through L3 tables are arbitrary and define here an inverse tone mapping process from from the LDR to the HDR regime. If the H-Transformation is identical to the L-Transformation, the residual stream represents a simple additive error residual that is added to the decoded, inversely tonemapped LDR signal. It is suggested to use the residual scan defined in Annex XXX for encoding the residual signal.

Kommentar [tr4]: Fix the reference.

A **lossless or near lossless** image coder can be realized by a slight modification of the above process, namely by setting the H-transformation to the inverse RCT transformation and using the lossless residual scan for encoding the residual data. Furthermore, the FCT must be selected as L-Transformation, the DCT in the 10918-1 decoder decoding the LDR signal must be the exact fixpoint DCT defined in Annex XXX and the R-Tables must be identities. In this setting, the data represented by the residual scan is selected to be the error signal created by the specified FCT and fixpoint DCT transformation. Lossless coding of the residual ensures that the decoded residual signal matches exactly the error created by the FCT and FDCT, and since these two transformations are fully specified, the residual signal can be selected to match the error signal exactly, creating an overall lossless process.

A **multiplicative** HDR image decoding process in which the residual image represents an exponent signal that scales the LDR image can be realized as follows: The L-Transformation should here be selected as either the FCT or the ICT, the L1 through L3 curves implement here a mapping from sRGB to linear gamma and may be selected to be identical to the sRGB nonlinearities. The S-Transformation extracts the luminance of the approximate HDR signal and scales by that the chroma signals of the residual signal. The R1 through R3 curves should here be selected such that the luminance signal is completely suppressed, i.e. the luma R-table is identical to zero, and the chroma curves may implement a simple linear function. The H-Transformation is here identical to the inverse ICT transformation, creating additive chroma residuals. The S-Table imlements in this setting an inverse logarithm, i.e. an exponential function, reconstructing the exact scale of the approximate HDR signal. The advantage of this approach is that the residual image behaives close to a natural image and can be compressed by legacy 10918-1 encoders, the drawback is that this scheme does not allow lossless coding.

In addition to these three application profiles many other profiles can be configured. The parameters specifying the L,S and R transformations and curves are recorded in JPEG Extension markers invisible to legacy decoders and so is the residual signal and the refinement signal. A legacy decoder would only reconstruct the low dynamic range portion of the image recorded in a codestream that is compliant to the interchange format specified in ITU.T Rec.T 80|ISO/IEC 10918-1.

5.4 Encoder requirements

An encoding process converts source image data to compressed image data. This includes first generating a low dynamic range image, and representing it by a coding process specified in Annex F or Annex G of ITU.T Rec.T 81 | ISO/IEC 10918-1, and then generating a residual image which is encoded by one of the processes defined in this Recommendation|International Standard.

In order to comply with this Specification, an encoder shall satisfy at least one of the following two requirements. An encoder shall with appropriate accuracy, convert source image data to compressed image data which comply with the codestream format syntax specified in Annex B for the encoding process(es) embodied by the encoder. If lossless coding is required, the FCT and fixpoint DCT specified in Annexes XXX and YYY must be implemented in a way that exactly reproduces exactly the output of the algorithms specified in the above Annexes, i.e. full accuracy is then required. For lossly coding, a limited accuracy sufficient to match the error bounds specified in the compliance tests is acceptable. Compliance tests for the above requirements are specified in this Recommendation | International Standard.

Kommentar [tr5]: This, of course, still has to be done.

NOTE – There is **no requirement** in this Specification that any encoder which embodies one of the encoding processes specified here shall be able to operate for all ranges of the parameters which are allowed for that process. An encoder is only required to meet the compliance tests and to generate the compressed data format according to Annexes B for those parameter values which it does use.

5.5 Decoder requirements

A decoding process converts compressed image data to reconstructed image data. For that, it has to follow the decoding operation specified in ITU.T Rec.T 81 | ISO/IEC 10918-1 with sufficient accuracy, using either the Baseline, Sequential or Progressive Scan process of the earlier Recommendation | International Standard defined there in Annexes F and G, and merge the resulting sample values with decoded residual values reconstructed from data included in the side channel represented by one or multiple application markers. The actual merging process is driven by parameters also recorded in Application Markers, the format of which is specified in this Recommendation | International Standard.

In order to comply with this Specification, a decoder shall satisfy all three of the following requirements. A decoder shall

a) with appropriate accuracy, convert a codestream conforming to this Recommendation | International Standard **without considering the side channel** into to a low dynamic range digital image.

b) **may additionally**, with appropriate accuracy, convert a conforming codestream including the side information included in the JPEG Extensions Markers, into a high dynamic range continuous tone image.

c) **may additionally** implement the lossless decoding process by using algorithms for the backwards DCT and backwards FCT that reproduce the output of the algorithms specified in Annexes XXX and YYY exactly. Decoders unable to reproduce exactly the output of these algorithms shall not claim to support the lossless profile.

• Annex A
Component Subsampling and Expansion of Subsampling

Kommentar [tr6]: This should probably be moved into one of the additional parts as subsampling expansion was not really part of the original JPEG, though it is part of any (software!) decoder I'm aware of. Walt would prefer to exclude it in part-1. I'm personally open for all options.

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

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

5.6 Component dimensions and subsampling factors (normative)

An image is defined to consist of Nf components, each of which is identified by a unique identifier C_i defined in the frame frame header of the codestream format specified in Annex B. The number of components Nf shall be either one, three or four. or four. A component consists of a rectangular array of samples x_i wide and y_i samples high. The component dimensions are dimensions are derived from the image dimensions X and Y, also parameters recorded in the frame header. These two parameters define a sample grid of X grid points wide and Y grid points high, where the left topmost grid coordinate is (0,0) (0,0) and coordinates increase from left to right and from top to bottom. However, not every grid point need to carry a sample sample value. For each component, two subsampling factors $s_{i,x}$ and $s_{i,y}$ define the spacing between sample points of component i relative to the sample grid and the size of the component array. If X and Y are the dimensions of the sample grid, sample grid, the size of component i with subsampling factors $s_{i,x}$ and $s_{i,y}$ is

Kommentar [tr7]: Allow alpha channels?

$$\lceil X/s_{i,x} \rceil \quad \text{and} \quad \lceil Y/s_{i,y} \rceil$$

Namely only those sample grid points whose horizontal coordinate is an integer multiple of $s_{i,x}$ and whose vertical coordiate is coordiate is an integer multiple of $s_{i,y}$ carry sample values that are represented in the codestream. Sample grid points whose whose sample value is not recorded in the codestream shall to be interpolated from their surrounding samples by mechanisms mechanisms specified in subclause A.2.

Kommentar [tr8]: This is actually not how JPEG was originally designed. Samples are not co-located as in JPEG 2000, but centered. It is unfortunately how one popular implementation works. Shall we change that?

The subsampling factors $s_{i,x}$ and $s_{i,y}$ are not directly represented in the binary codestream or any of its markers, but shall be derived from the parameters H_i and V_i recorded in the frame header. If Nf equals one, i.e. the image consists of a single component, H_1 and V_1 shall be one, and $s_{1,x}$ and $s_{1,y}$ are both one. If Nf equals three, Table A-1 defines the relation between H_i, V_i and $s_{i,,x}$ and $s_{i,y}$. No other combinations of H_i and V_i than those listed in this table shall be used. If Nf equals four, the only allowable values for $s_{4,,x}$ and $s_{4,y}$ are one, and thus H_4 and V_4 are identical to H_1 and V_1. The allowable values for $s_{i,,x}$ and $s_{i,y}$ for i<4 shall be again only those listed in Table A-1.

H_1	V_1	H_2	V_2	H_3	V_3	$s_{1,x}$	$s_{1,y}$	$s_{2,x}$	$s_{2,y}$	$s_{3,x}$	$s_{3,y}$
1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	1	2	1	1	1	1	2	1	2
2	2	1	2	1	2	1	1	2	1	2	1
2	2	1	1	1	1	1	1	2	2	2	2
All other values reserved for ITU/ISO purposes											

Table A-1: Subsampling Values

Kommentar [tr9]: Do we actually need this?

Kommentar [tr10]: Ditto – Walt would prefer to not include it. No problem with me.

NOTE – ISO/IEC 10918-1 allowed other component arrangements and relations between grid positions and sample positions that are not valid in this Recommendation | International Standard. However, the definitions given here are special cases of the more general relations provided in 10918-1 and both definitions agree whenever both are defined.

5.7 Expansion of subsampled components (normative)

Whenever the subsampling factors $s_{i,x}$ and $s_{i,y}$ are not both one, sample values at sample grid positions not populated by actual samples shall be interpolated from surrounding positions. The precise algorithm how this interpolation has to be performed is not specified in this Recommendation | International Standard and various implementation choices exist. This subclause provides an example for a simple bi-linear interpolation that is sufficient for most applications, though more sophisticated choices might provide better subjective image quality.

5.8 Bilinear expansion of subsampled components (informative)

Kommentar [tr11]: Actually, the algorithm shown here is probably not the best given that samples in JPEG are actually not co-located, but centered. It is however the method implemented by IJG. Any better options?

In a first step, check for each component i whether $s_{i,y}$ is two or one. If $s_{i,y}$ is one, perform no activity. If $s_{i,y}$ equals two, interpolate samples v at odd lines from even lines as follows:

$$v_{x,2y+1} \quad = \quad \lfloor (v_{x,2y}+v_{x,2y+2}+(y \bmod 2))/2 \rfloor$$

If 2y+2 is larger or equal Y, the horizontal sample grid dimension, set $v_{x,2y+1}$ equal to $v_{x,2y}$.

In a second step, check for each component i whether $s_{i,x}$ is two o one. If $s_{i,x}$ is one, no further activity is required and the algorithm terminates. Otherwise, if $s_{i,x}$ is equal to two, interpolate samples at odd positions from samples at even positions accordingly:

$$v_{2x+1,y} \quad = \quad \lfloor (v_{2x,y}+v_{2x+2,y}+(x \bmod 2))/2 \rfloor$$

Again, if 2x+2 is larger or equal than X, the vertical sample grid dimension, set $v_{2x+1,y}$ to $v_{2x,y}$.

5.9 Downsampling of components (informative)

This Recommendation|International Standard does not define a normative procedure by which the resolution of components whose $s_{i,x}$ and $s_{i,y}$ factors are not both one shall be reduced. Any procedure that generates components of the size $X/s_{i,x}$ and $Y/s_{i,y}$ is acceptable as long as it is compatible with the upsampling procedure defined above. A very simple downsampling filter is given in the next subclause.

5.10 Downsampling by a box filter (informative)

Kommentar [tr12]: This is surely a lousy choice.

The box filter is the simplest possible downsampling filter and provides only poor quality. Even though better alternatives exist, the box filter is nevertheless presented here as an example. The input of the box filter is a X×Y component array of samples, where the sample value at position x,y is denoted by $v_{x,y}$. Set now

$x_{min} := s_{i,x} \times x_s \quad x_{max} := \min(s_{i,x} \times x_s + s_{i,x} - 1, X-1) \qquad y_{min} := s_{i,y} \times y \quad y_{max} := \min(s_{i,y} \times y_s + s_{i,y} - 1, Y-1)$

The output of the box filter at position x_s,y_s is then defined as

$$v^s_{x,y} := \left(\Sigma_{x=xmin..xmax}\ \Sigma_{y=xmin..ymax} v_{x,y}\right) / \left((x_{max}-x_{min}-1) \times (y_{max}-y_{min}-1)\right)$$

i.e. the average over the box x_{min},y_{min} to x_{max},y_{max}. The array of downsampled sample values $v^s_{x,y}$ is then subject to further processing, e.g. DCT transformation and entropy coding.

Annex B
Codestream Syntax

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

This annex defines the compressed bitstream syntax which, structurally, consists of an ordered collection of marker segments and entropy coded data segments. Marker segments specify parameters necessary to reconstruct the sample values from the entropy coded data segments. Because all of these constituent parts are represented with byte-aligned codes, each compressed data format consists of an ordered sequence of 8-bit bytes. For each byte, a most significant bit (MSB) and a least significant bit (LSB) are defined.

NOTE – The codestream syntax defined here agrees mostly with the "interchange format" defined in ITU.T Rec. T.81| ISO/IEC 10918-1, with some additional constraints on the parameters in the marker segments and some additional markers carrying information that is irrelevant for the older standard. Such additional data is encapsulated in a way that will be ignored by legacy decoders complying to the older standard.

5.11 Parameters

Parameters are integers, with values specific to the encoding process, source image characteristics, and other features selectable by the application. Parameters are assigned either 4-bit, 1-byte, 2-byte or 4-byte codes. Except for certain optional groups of parameters, parameters encode critical information without which the decoding process cannot properly reconstruct the image. The code assignment for a parameter shall be an unsigned integer of the specified length in bits with the particular value of the parameter.

For parameters which are 2 bytes (16 bits) in length, the most significant byte shall come first in the compressed data's ordered sequence of bytes. The same holds for parameters that are 4 bytes (32 bit) in length, where bits are ordered in the codestream in decreasing significance. Parameters which are 4 bits in length always come in pairs, and the pair shall always be encoded in a single byte. The first 4-bit parameter of the pair shall occupy the most significant 4 bits of the byte. Within any 32-, 16-, 8-, or 4-bit parameter, the MSB shall come first and LSB shall come last. This encoding is commonly known as "big endian" representation of unsigned integers.

5.12 Markers

Markers serve to identify the various structural parts of the compressed data formats. Most markers start marker segments containing a related group of parameters; some markers stand alone. All markers are assigned two-byte codes: an 0xff byte followed by a byte which is not equal to 0x00 or 0xff. Any marker may optionally be preceded by any number of fill bytes, which are bytes assigned code 0xff.

NOTE – Because of this special code-assignment structure, markers make it possible for a decoder to parse the compressed data and locate its various parts without having to decode other segments of image data.

5.13 Marker assignments

All markers shall be assigned two-byte codes: a 0xff byte followed by a second byte which is not equal to 0x00 nor 0xff. The second byte is specified in Table B-1 for each defined marker. An asterisk (*) indicates a marker which stands alone, that is, which is not the start of a marker segment. Most of the marker segments used by this Recommendation | International Standard are defined in ITU.T Rec.T.81 | ISO/IEC 10918-1, though some marker segments defined there shall not be used in this Recommendation|International standard. For completeness, these markers are also included in Table B-1. Furthermore, care must be taken that the parameters of some markers are more constraint than in the earlier Recommendation | International Standard, and the meaning of several markers changed slightly. Such changes are also indicated in the table, and a full specification of the changes follows.

Code Assignment	Symbol	Description	Defined in
Start of Frame markers valid in this Recommendation | International Standard			
0xFFC0	SOF_0	Baseline DCT Process	This Recommendation | Internatinonal Standard
0xFFC1	SOF_1	Extended Sequential DCT Process	
0xFFC2	SOF_2	Progressive DCT Process	
Start of Frame markers defined in T.81|10918-1 that shall not be used in this Recommendation | International Standard			
0xFFC3	SOF_3	Additional Start of Frame Markers, shall not be used.	ITU.T Rec.T81 | ISO/IEC 10918-1
0xFFC5	SOF_5		
0xFFC6	SOF_6		
0xFFC7	SOF_7		
0xFFC9	SOF_9		
0xFFCA	SOF_{10}		
0xFFCB	SOF_{11}		
0xFFCD	SOF_{12}		
0xFFCE	SOF_{14}		
0xFFCF	SOF_{15}		
Table Specifications			
0xFFC4	DHT	Define Huffman Tables	ITU.T Rec.T81 | ISO/IEC 10918-1
0xFFDB	DQT	Define Quantization Tables	ITU.T Rec.T81 | ISO/IEC 10918-1
Restart Interval Termination			
0xFFD0 through 0xFFD7	RST_m^*	Restart modulo 8 count 'm'	ITU.T Rec.T81 | ISO/IEC 10918-1
0xFFDD	DRI	Define Restart Interval	ITU.T Rec.T81 | ISO/IEC 10918-1
Other Markers			
0xFFD8	SOI^*	Start of Image	ITU.T Rec.T81 | ISO/IEC 10918-1
0xFFD9	EOI^*	End of Image	
0xFFDA	SOS	Start of Scan	
0xFFFE	COM	Comment	
JPEG Extension Markers			
0xFFE0	APP_0	JFIF Application Marker	ISO/IEC 10918-5 and ISO/IEC 10918-4
0xFFE1	APP_1	EXIF Application Marker	JEITA CP-3451 and ISO/IEC 10918-4
0xFFE2	APP_2	ICC Profile Marker	ISO/IEC 10918-6 and ISO/IEC 10918-4
0xFFE3 through 0xFFE8	APP_3 through APP_8	Application Markers 3 through 8	ISO/IEC 10918-4
0xFFE9	APP_9	JPEG Extensions Marker	This Recommendation | International Standard
0xFFEA	APP_{10}	Application Marker 10	ISO/IEC 10918-4
0xFFEB	APP_{11}	JPEG High Dynamic Range	This Recommendation |

		Extensions Marker	International Standard
0xFFEC	APP$_{12}$	Application Marker 12	ISO/IEC 10918-4
0xFFED	APP$_{13}$	Component Decorrelation Control Marker	This Recommendation | International Standard
0xFFEE through 0xFFEF	APP$_{14}$ through APP$_{15}$		
Markers defined in other specifications that shall not be used in this Recommendation | International Standard			
0xFFC8	JPG	Reserved for JPEG Extensions	ITU.T Rec.T81 | ISO/IEC 10918-1
0xFFCC	DAC	Define Arithmetic Coding Conditions	
0xFFDC	DNL	Define Number of Lines	
0xFFDE	DHP	Define Hierarchical Progression	
0xFFDF	EXP	Expand Reference Components	
All other values		Reserved for ITU/ISO purposes, shall not be used in this Recommendation | International Standard	

Table B-1: Markers and Marker Segments

Kommentar [tr13]: For consistency, I would prefer if all entropy coded data would go into a consistent APP9 marker space.

5.14 Marker segments

A marker segment consists of a marker followed by a sequence of related parameters. The first parameter in a marker segment is the two-byte length parameter. This length parameter encodes the number of bytes in the marker segment, including the length parameter and excluding the two-byte marker. The marker segments identified by the SOF and SOS marker codes are referred to as headers: the frame header and the scan header respectively.

5.15 Entropy-coded data segments

An entropy-coded data segment contains the output of an entropy-coding procedure. It consists of an integer number of bytes created by the Huffman coding procedure. An entropy coded data segment stands either for itself and is then started by an SOS marker, or it is contained in the APP$_9$ or APP$_{11}$ JPEG Extensions marker. In the latter cases, the size of the marker segment containing the entropy coded data includes the size of the entropy coded data, the size of the Common Identifier (CI) and the JPEG Extensions Type Identifier (TI) and the size of the 16-bit marker size. The marker itself is not included in the count.

Kommentar [tr14]: I would prefer to use a consistent model for entropy coded data where the Dolby HDR extensions also go into APP9.

Kommentar [tr15]: CI and TI look different for the Dolby codestream, hence I would prefer to have a common definition.

Entropy coded segments started by an SOS marker shall be always a multiple of eight bits long, and shall be padded by one bits at the end of the segment to fill an entire byte. In order to ensure that a marker does not occur within an entropy-coded segment, the encoder shall insert a zero byte into the output bitstream following any any 0xff byte generated by either the Huffman encoder or by the above termination algorithm. Decoders shall remove such "stuffed" zero bytes before feeding the resulting bitstream into the Huffman decoder.

Since APP$_9$ markers have a limited capacity of 2^{16}-1 bytes, entropy coded data may be partitioned over several segments with with identical Common Identifier and Type Identifier. Prior decoding, the decoder shall remove the marker length field, and field, and the CI and TI fields and shall concatenate the remaining contents of the markers in the order they appear in the the codestream. The resulting data stream is then an entropy coded segment in the above sense.

Kommentar [tr16]: The Dolby proposal encloses an entire (selfcontained) JPEG codestream in APP11. I'm not doing this here and would prefer not to do it. Comments welcome.

NOTE – Note that by the above paragraph, byte stuffing also applies to entropy coded data contained in APP$_9$ markers, and padding applies to the last byte of the concatenated stream only. Note further that due to the segmentation of entropy coded data in application markers, it may happen that the last byte of an APP$_9$ marker segment is 0xff, and that the corresponding "stuffed" zero byte is part of a subsequent application marker segment. This does not cause a problem for legacy decoders since they are required to skip over unknown application marker segments in first place, without interpreting their content.

5.16 High-level syntax

The high-level syntax of the codestream defined in this Recommendation|International Standard shall follow the syntax of the Interchange Format specified in Subclauses B.2 and B.3 of ITU.T Rec.T 81 | ISO/IEC 10918-1 where only the subset of the markers specified in Table B-1 shall be used.

Specifically, conforming codestreams shall start with an SOI marker, followed by a single frame, followed by a single EOI marker. The high level syntax is depicted in Figure B-1.

SOI Marker	Frame	EOI

Figure B-1: High-level syntax of the codestream

NOTE – ITU.T Rec.T.81|ISO/IEC 10918-1 allowed multiple frames for the hierarchical coding mode, though this mode is not applicable in this Recommendation | International Standard.

A frame consists of zero or more table definitions containing the coding parameters of the LDR and residual image data, and optionally also JPEG Extension markers including the residual image, followed by the frame header, followed by one or more scans over the LDR image. The scan type of the LDR data is indicated by the type of the SOF (Start of Frame) marker starting the frame header, see Table B-1. The frame header syntax is defined in Subclause B-XXX. The syntax of a frame is indicated in Figure B-2.

[Tables/JPEG Extensions Markers/Misc]	Frame header	Scan_1	…	$\text{Scan}_{\text{Last}}$

Figure B-2: Syntax of a Frame

The Tables consist of zero or more of the following marker segments:

- Zero or more DHT marker segments defining the Huffman code required for decoding the entropy coded segment,
- zero or more DQT marker segments defining the quantization matrix required to reconstruct the DCT coefficients,
- at most one DRI marker segment defining the restart interval,
- zero or more JPEG Extension marker segments containing additional coding parameters and/or entropy coded data segments defining the residual image,
- an optional Component Decorrelation Control marker, defining the inverse multi-component decorrelation transformation for the LDR bitstream,
- optionally, additional application specific data encoded in APP_n markers
- optionally, one or more COM markers including additional codestream comments.

NOTE – Unlike ITU.T Rec.T.81|ISO/IEC 10918-1, DNL markers are not valid in this Recommendation|International Standard. The size of the image is thus to be completely defined by the frame header.

A scan consists of zero or more table definitions including coding parameters or JPEG Extension markers, followed by a scan header starting with an SOS marker, and one or more entropy coded segments. If restart markers are disabled, a scan shall only contain a single entropy coded segment. Otherwise, the scan may contain multiple entropy coded segments separated by restart markers. The syntax of the restart markers shall follow the definitions found in ITU.T Rec.T.81|ISO/IEC 10918-1. The syntax of a scan is indicated in Figure B-3.

The Tables/Extensions/Misc segment of a scan shall follow the same requirements as the Tables/Extensions/Misc segment in the frame. DHT tables arriving later in the codestream, i.e. in the first or any subsequent scan, shall replace the Huffman tables with the same table class and table destination defined earlier in the codestream. A DRI marker arriving later in the codestream shall replace the restart interval definition found earlier.

Kommentar [tr17]: All this does not yet define how to handle markers contained in the residual scans. Currently, the Stuttgart implementation uses a common database for all markers, though arguably this is a bad choice.

There shall be at most one DQT table for each table destination in the codestream, replacing an already defined quantization matrix by inserting a DQT marker segment with a table destination already defined earlier in the codestream is invalid. Furthermore, the codestream shall include sufficient information to be self-contained, i.e. to allow reconstruction of the sample values with information contained in the codestream only.

Additional requirements may apply to other marker segments found in the Tables/Extensions/Misc segment, they are defined together with the corresponding marker segment in this Annex.

[Tables/ JPEG Extensions Markers/ Misc]	Scan header	ECS_0	RST_0	…	ECS_{Last-1}	RST_{Last}	ECS_{Last}

Figure B-3: Syntax of a Scan

5.17 Frame header syntax

Table B-2 specifies the frame header which shall be present at the start of a frame. Frame headers always start with a SOF_n marker, where n defines the type of the LDR scans contained in the frame. See Table B-1 for scan types supported by this Recommendation|International Standard.

The frame header specifies the dimensions of the image, the components in the frame, and the sampling factors for each component, and specifies the destinations from which the quantized tables to be used with each component are retrieved.

Parameter	Size (in Bits)	Value	Meaning
SOF_n	16	0xFFC0 through 0xFFC2	Start of Frame Marker, indicating this marker segment and the type of the LDR scans
Lf	16	8+3*Nf	Size of the marker segment not including the marker
P	8	8	Precision of the low dynamic range representation of the image contained in the legacy codestream
Y	16	1-65535	Height of the sample grid in lines
X	16	1-65535	Width of the sample grid in lines
Nf	8	1 or 3	Number of visible components in the image
C_i	8	0-255	Component identifiers
H_i	4	1 or 2	Specification of the component subsampling factors, see Table A-1 for allowable values and the relation between H_i V_i and $s_{i,x}$ and $s_{i,y}$
V_i	4	1 or 2	
Tq_i	8	0-3	Quantization Table Destination selector

Table B-2: Frame Header Parameters and values

NOTE – The frame header syntax defined here is identical to the syntax defined in ITU.T Rec.T 81| ISO/IEC 10918-1 except that some some parameters are restricted. Specifically, the bit precision of the components must always be eight, the height must be greater than

than zero indicating that the number of lines of the image is known, the number of components must be either one or three and the horizontal and vertical sampling factors are constraint to the values listed in table A-1.

5.18 Scan Header Syntax

The scan header defines the parameters necessary to decode a single scan over the LDR image. For the baseline or sequential frame syntax, this is the only scan that represents the low dynamic range version of the image. Multiple scans are possible in the progressive mode. Residual data necessary to recover the high dynamic range representation of the image is recorded in one or several JPEG extensions markers; encoding or decoding this image representation may require additional scans over the image that are not represented by the scan header defined in this subclause.

The syntax of the scan header follows the definition for the same marker in ITU.T Rec. T81 | ISO/IEC 10918-1, though some of the parameters are here more constrainted than in the older Recommendation | International Standard. The parameters and sizes of the Scan header are defined in Table B-3.

Parameter	Size (in Bits)	Values			Meaning
		Baseline	Extended Sequential	Progressive	
SOS	16	0xFFDA			SOS marker, identifies the start of the scan
Ls	16	6+2*Ns			Size of the marker segment (not including the marker)
Ns	8	1 or 3 or 4		1 or 3 or 4 (if the progressive scan is a subsequent scan and includes AC components, only 1 is allowed here)	Number of components in the scan
Cs_j	8	0-255 (shall be a subset of the component identifiers defined in the frame header)			Component identifiers of the components defined by the scan
Td_j	4	0-1	0-3		Huffman DC table specification
Ta_j	4	0-1	0-3		Huffman AC table specification
Ss	8	0		0-63	Start of Spectral Selection
Se	8	63		Ss-63	End of Spectral Selecation

Kommentar [tr18]: Still unclear how to include alpha here. Just setting this to four is probably a bad choice.

Ah	4	0	0-9 If nonzero, must be equal to Al+1	Successive Approximation high bit position
Al	4	0	0-8	Successive Approximation low bit position

Table B-3: Scan Header Parameters and Values

NOTE – The scan header syntax defined here is identical to the syntax defined in ITU.T Rec.T 81| ISO/IEC 10918-1, though a scan may contain at most three components.

Kommentar [tr19]: or four? Discuss!

5.19 Component Decorrelation Control marker

Kommentar [tr20]: This should possibly not go into part-1, even though all software implementations I am aware of support it, and support it exactly in the way described here. Walt would be happy not to, I do not mind too much, though I believe it would make sense here on the rationale of standardizing a living specification. Maybe make it informative here and normative later.

There is one corner point, namely if both this marker and the JFIF marker is present. Then, it seems, IJG ignores this marker. Should be specifiy this corner case?

The APP_{13} controls whether the Multi-Component Decorrelation Transformation described in Annex C is applied to the components before subsampling and entropy coding. This transformation typically improves the compression performance by representing data in an RGB type colorspace in a luma/chroma format that is close to YCbCr. This marker shall only be inserted into the codestream as parts of the Table/misc segment if the number of components in the frame (Nf) is three, and there shall be at most one such marker in the codestream. The marker parameters and sizes are defined in Table B-4, the organization of the marker is shown in Figure B-1.

NOTE – This marker is identical to the "Adobe Color Management" marker defined in ISO/IEC 10918-6, though the possible values the parameters inside the marker segment are more restricted. The intended use is also slightly different: Whereas the marker in ISO/IEC 10918-6 defines an output color space of the encoded samples, it controls here the decorrelation transformation from the internal representation to the external color format, though not necessarily the color space itself. Legacy software, however, will assume an RGB-type output colorspace in the absence of any additional information and will implement a color transformation close to the decorrelation transformation specified in Annex C, though with the intent to change the colorspace and not to undo a component decorrelation transformation. The presence of this marker, with suitable parameters, will then suppress this change in the colorspace, and thus implement the desired functionality, namely to disable the multi-component decorrelation transformation.

0xFFED	La	ACid	ver	f1	f2	cc

Figure B-1 : Organization of the Component Decorrelation Control marker

Parameter	Size (bits)	Value	Meaning
APP_{13}	16	0xFFED	APP_{13} marker encapsulating the component decorrelation control information.
La	16	14	Size of the marker segment (not including the marker)
ACid	40	0x41 0x64 0x6f 0x62 0x65 (ASCII encoding of "Adobe")	Identifies the use of the APP_{13} marker segment for the purpose of the Component Decorrelation Control marker. Readers shall ignore the APP_{13} marker for the purpose of color decorrelation if this value does not match.
ver	16	0x64	A version number. Only version 100 is defined.
f1	16	0	Flags 1. Only the value 0 is defined in this Recommendation | International Standard.
f2	16	0	Flags 2. Only the value 0 is

			defined in this Recommendation | International Standard.
cc	8	0 or 1	Color Decorrelation Control Byte. If this value is zero, the L-Transformation shall be replaced by the identity transformation. If this value is one, or the marker is not present, the L-Transformation shall be employed as specified by the JPEG Extensions parameter marker. All other values are reserved and their meaning is not defined by this Recommendation | International Standard.

Table B-4: Component Decorrelation Control marker parameters and sizes

5.20 JPEG Extensions Marker Segment

The APP_9 marker is reserved by this Recommendation | International Standard to include additional parameters and entropy coded data that remain invisible to legacy decoders. All extensions share a common syntax that is defined in this and the following subclauses.

A JPEG Extensions marker segment consists of the marker, the marker length that includes the marker length, all parameters and entropy coded data but not the marker itself, a two-byte common identifier and a four byte type field, plus all parameters and data encapsulated by the marker. Figure B-4 shows the organization of the marker, Table B-4 defines the fields, the field lengths and the contents of a JPEG Extensions marker segment.

0xFFE9	Le	Common Identifier	Type Identifier	Payload Data

Figure B-4: Organization of the JPEG Extensions Marker Segment

Parameter	Size (bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies all JPEG Extension Marker Segments.
Le	16	8..65535	Length of the marker segment, including the size, the common identifier, the type identifier and the payload, but not including the marker itself.
CI	16	0x4A50 (ASCII encoding of "JP")	The special value 0x4A50 (ASCII: 'J' 'P') allows readers to to distinguish the JPEG Extensions marker segment from other uses of the APP_9

			marker. Readers shall ignore the APP_9 marker for the purpose of decoding JPEG extensions if this value does not not match.
TI	32	See subclauses B.XX through B.YY	The Type Identifier describes the type of the payload data that follows. Subclauses B.XX through B.YY define the payload data used in this Recommendation|International Standard.
Data	Varies	Varies	Defined in subclauses B.XX through B.YY.

Table B-5: JPEG Extensions Marker Parameters and Sizes

5.20.1 JPEG Extensions Marker Segment: Entropy Coded Data for Refinement Coding

Entropy coded segments from refinement coding are encapsulated in this variant of the JPEG Extensions marker. This data extends the precision of the DCT transformed coefficients by additional least significant bits that improve the precision of the DCT coefficients beyond the sample depths available for legacy decoders. To form the entropy coded bitstream, the contents of the Entropy Coded data from all JPEG Extensions markers of this type shall be concatenated by a decoder in the order the markers appear in the codestream, and then form an input for the refinement entropy decoding process defined in Annex XX.

The structure of this marker is defined in Figure B-3, the parameters and sizes in Table B-6.

0xFFE9	Le	Common Identifier	Type Identifier	Entropy Coded Data

Figure B-3: Organization of the JPEG Extensions Marker Segment for Refinement Coding

Parameter	Size (bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies this marker as JPEG Extensions marker.
Le	16	8..65535	Length of the marker segment, including the size, the common identifier, the type identifier and the payload, but not including the marker itself.
CI	16	0x4A50 (ASCII encoding of "JP")	Identifies this marker segment as JPEG Extensions marker
TI	32	0x46494e45 (ASCII encoding of "FINE")	Identifies this marker as carrying entropy coded data for refinement coding.
Data	Varies	Varies	Entropy coded data segment of variable lengths.

Table B-6: JPEG Extensions Marker for Refinement Coding, Parameters and Sizes

5.20.2 JPEG Extensions Marker Segment: Entropy Coded Data for Residual Coding

Entropy coded segments from residual coding are encapsulated in this variant of the JPEG Extensions marker. This data provides entropy coded data that, when decoded and merged with the low-dynamic range data, describes a high-dynamic range image. The algorithm for decoding residual data is defined in Annex XX, the operation for merging the decoded low-dynamic range data with the high-dynamic range data in Annex YY.

To form the entropy coded residual bitstream, the contents of the Entropy Coded data from all JPEG Extensions markers of this type shall be concatenated by a decoder in the order the markers appear in the codestream, and then form an input for the residual entropy decoding process defined in Annex XX.

The structure of this marker is defined in Figure B-4, the parameters and sizes in Table B-7.

0xFFE9	Le	Common Identifier	Type Identifier	Entropy Coded Data

Figure B-3: Organization of the JPEG Extensions Marker Segment for Residual Coding

Parameter	Size (bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies this marker as JPEG Extensions marker.
Le	16	8..65535	Length of the marker segment, including the size, the common identifier, the type identifier and the payload, but not including the marker itself.
CI	16	0x4A50 (ASCII encoding of "JP")	Identifies this marker segment as JPEG Extensions marker
TI	32	0x52455349 (ASCII encoding of "RESI")	Identifies this marker as carrying entropy coded data for residual coding.
Data	Varies	Varies	Entropy coded data segment of variable lengths.

Table B-6: JPEG Extensions Marker for Residual Coding, Parameters and Sizes

5.20.3 JPEG Extensions Marker Segment: Inverse Tone Mapping Marker

This marker segment defines an inverse tone mapping curve in the form of a look-up table. This look-up table may be used to implement the L_i, S or R_i inverse tone mapping operations, which are part of the merging process for combining the low-dynamic range and residual image information to reconstruct a high-dynamic range image. An additional method to define an inverse tone mapping curve is by a parametric description, defined in subclause B-XXX. The merging process is defined in more detail in Annex YYY.

For each tone mapping destination identifier at most one tone mapping information, either in the form of an Inverse Tone Mapping Marker or a Parametric Inverse Tone Mapping Marker shall be present in the codestream.

Inverse tone mapping is applied by using an input value in the range of 0 to 2^{8+h}-1 as an index in the D_k table recorded in the marker. The output of the inverse tone mapping operation is then defined as $D_k[v]/65535\times ts_i$, where ts_i is the scaling value defined in the JPEG Extensions parameter marker.

The marker segment organization is defined in Figure B-4, the parameters and sizes in Table B-7.

0xFFE9	Le	Common Identifier	Type Identifier	M	h	D_i

Figure B-3: Organization of the JPEG Extensions Marker: Inverse Tone Mapping Curve

Parameter	Size (in Bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies this marker as JPEG Extensions marker.
Le	16	$9+2*2^{h+8}$	Length of the marker segment (not including the marker)
CI	16	0x4A50 (ASCII encoding of "JP")	Identifies this marker segment as JPEG Extensions marker
TI	32	0x544f4e45 (ASCII encoding of "TONE")	Identifies this marker as carrying inverse tone mapping information in table form.
M	4	0..15	Tone mapping table destination. Up to 16 tone mapping curves can be defined.
h	4	0..4	Number of hidden DCT bits represented by refinement coding.
D_k	16	0..65535	Output high dynamic range sample value for low dynamic range sample value i. The marker includes 2^{h+8} table entries each 16 bits wide.

Table B-7: JPEG Extensions Marker containing an Inverse Tone Mapping Curve

5.20.4 JPEG Extensions Marker Segment: Parametric Inverse Tone Mapping Marker

This marker segment defines an inverse tone mapping curve in the form of a parametrized curve, which is for some common settings for efficient than a look-up table. This parametric curve may be used to implement the L_i, S or R_i inverse tone mapping operations, which are part of the merging process for combining the low-dynamic range and residual image information to reconstruct a high-dynamic range image. An additional method to define an inverse tone mapping curve is by a parametric description, defined in subclause B-XXX. The merging process is defined in more detail in Annex YYY.

For each tone mapping destination identifier at most one tone mapping information, either in the form of an Inverse Tone Mapping Marker or a Parametric Inverse Tone Mapping Marker shall be present in the codestream.

The parametric inverse tone mapping marker with curve types different than t=0 or t=1 shall not be used for the lossless coding process.

The marker segment organization is defined in Figure B-4, the parameters and sizes in Table B-7.

0xFFE9	Le	Common Identifier	Type Identifier	M	t	P_1	P_2

Figure B-3: Organization of the JPEG Extensions Marker: Parametric Inverse Tone Mapping Curve

Parameter	Size (in Bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies this marker as JPEG Extensions marker.
Le	16	$9+2*2^{h+8}$	Length of the marker segment (not including the marker)
CI	16	0x4A50 (ASCII encoding of "JP")	Identifies this marker segment as JPEG Extensions marker
TI	32	0x43555256 (ASCII encoding of "CURV")	Identifies this marker as carrying a parametric inverse tone mapping curve.
M	4	0..15	Tone mapping table destination. Up to 16 tone mapping curves can be defined.
t	4	0..15	Curve type. Curve types are defined in Table B-XX.
P_1	32	Depends on t	Curve parameter 1, encoded as big-endian IEC 60559 single precision floating point value
P_2	32	Depends on t	Curve parameter 2, encoded as big-endian IEC 60559 single precision floating point value
P_3	32	Depends on t	Curve parameter 3, encoded as big-endian IEC 60559 single precision floating point value
P_4	32	Depends on t	Curve parameter 4, encoded as big-endian IEC 60559 single precision floating point value

Table B-7: JPEG Extensions Marker defining a parametric Tone Mapping Curve

The tone mapping curve to be applied is defined by the t parameter. Depending on t, the parameters P_1 and P_2 further specify the curve. Note that both parameters are always present, regardless of the actual curve type. They are, however, eventually ignored. Table B-XXX specifies the available parametric curve types. The input value x is here normalized to the maximum amplitude of the channel, i.e. $x=v/(2^{8+h}-1)$ for the LDR image, where h is the number of hidden DCT bits and v the reconstructed sample value, and the output is again scaled by ts_i, where ts_i is the scaling value defined in the JPEG Extensions parameter marker.

Curve Type, Value of t	Inverse tone mapping to be performed	Remarks
0	$f(x) = 0$	The zero tone mapping, the output is deleted. This is useful for selectively disabling parts of the reconstruction and to allow a fully additive or fully multiplicative reconstruction process. P_1 through P_4 are ignored.
1	$f(x) = 1$	The constant-one tone mapping curve. This is a useful choice for the S-Lut for implementing a fully additive decoding process. P_1 through P_4 are ignored.
2	$f(x) = x$	Identity transformation, input data is passed through. Note that due to input and output scaling, this is equivalent to a linear scaling of the input by $ts_i/(2^{8+h} - 1)$.
3		Reserved for ITU|ISO purposes
4	$f(x) = (1+P_3)x^{P2}-P_3$ for $x > P_1$ $f(x) = P_2(1+P_3)P_1^{P2-1}x$ for $x \leq P_1$	This is an inverse gamma mapping with parameters P_1 through P_3. P_4 is ignored. For P_1=0.0031308, P_2=0.416667, P_3=0.055, this is the sRGB nonlinearity.
5	$f(x) = x \times (P_2-P_1)+P_1$	Linear ramp with start value P_1 and end value P_2. Note that P_3 and P_4 are ignored. Note further that the input x and the output f(x) are not directly the sample value.
6	$f(x) = \exp(x \times (P_2-P_1)+P_1)$	Exponential function interpolating between $\exp(P_1)$ and $\exp(P_2)$
7..15		Reserved for ITU|ISO purposes.

5.20.5 JPEG Extensions parameter marker segment: Specifications of the Merging Process

This type of the JPEG extensions marker segment defines the process and parameters for merging low dynamic range image with the residual image to reconstruct the high dynamic range image. It shall be present in the Tables section of the frame or scan whenever a residual image or a refinement scan is included in the codestream. The details of the merging process of residual/refinement data with LDR data is defined in Annex XX. Figure B-4 describes the organization of this marker segment, Table B-8 the parameters and parameter sizes.

0xFFE9	Le	CI	TI	Lcod	Rcod	Scod	h	Lf	Oc	A	Rb	Rq_i	NL

Figure B-4: Organization of the JPEG Extensions Parameter Marker: Specification of the Merging Process

Parameter	Size (in bits)	Value	Meaning
APP_9	16	0xFFE9	Identifies this marker as a JPEG Extensions marker
Le	16	TBD	Length of the marker segment (not including the marker)
CI	16	0x4A50 (ASCII encoding of "JP")	Identifies this marker segment as JPEG Extensions marker
TI	32	0x53504543 (ASCII encoding of "SPEC")	Identifies this marker as carrying entropy coded data for residual coding.
Lcod	40	See Table B-xxx	Defines the L-Lookup tables and the L Transformation
Rcod	40	See Table B-xxx	Defines the R-Lookup tables and the H-Transformation
Scod	40	See Table B-xxx	Defines the S-Lookup table and the S-Transformation
h	4	0..4	Number of hidden DCT bits represented by refinement coding. The value of h here shall be identical to the h parameter in all 'TONE' JPEG Extension markers. The bit precision of the LDR decoding plus refinement decoding is computed as 8+h.
Lf	1	0 or 1	Lossless flag. If set to one, lossless decoding is desired, and the decoder shall pick the fixpoint DCT for decoding the legacy codestream. If zero, the DCT for reconstruction has only to follow the looser constraints of ITU.T Rec.T.81 | ISO/IEC 10918-1.
Oc	1	0..1	If 1, output data shall be converted to floating point

			by the procedure indicated in in Annex XXXX, otherwise otherwise output data is integer.
A	2	0,1 or 2	Reserved for alpha component definition. Suggested use: 0: No alpha components, 1 alpha component present, 2 premultiplied alpha component present.
Rb	8	0..7	Number of additional bits available for high dynamic range data. The bit precision of the high dynamic range data shall be computed as 8+Rb.
Rq_i	8	See Table B-XXX	Four fields identifying the inverse quantization and DCT process for residual and low dynamic range data. This parameter set is defined in Table XXX.
NL	32	variable	Encodes the noise floor value NL as a big endian IEC 60559 single precision floating point value.

Kommentar [tr21]: Still to be written

Table B-8: JPEG Extensions Marker containing an Inverse Tone Mapping Curve

The lookup table and transformation matrix definitions for the L,R and S tables and transformations are defined by the following parameters, all of which are part of the JPEG Extensions parameter marker as Lcod, Scod or Rcod field. These parameters define the lookup table to use, where for each component a lookup table destination index is given. The lookup table for component i=0..3, the decoder shall locate the JPEG Extensions Inverse Tone Mapping marker or the JPEG Extensions Inverse Parametric Tone Mapping Cuve whose M value is equal to td_i. The decoded component values are then used as either an index or an input into the table and generate an output between 0 and 1. This output is then to be scaled according to the ts_i parameter.

Parameter	Size (in bits)	Value	Meaning
ts_0	4	0..2	Postscaling value for the lookup table applied to output component 0. If zero, the output is zero and hence inverse tone mapping is disabled. If 1, the output remains unscaled in the range [0,1]. If 2, the output is scaled by 2^{8+Rb}, i.e. the full range of the high-resolution data.
td_0	4	0..15	Selects the lookup table or parametric inverse tone

			mapping curve whose M value is equal to td_0. This inverse tone mapping is then then applied to component 0. 0.
ts_1	4	0..2	Postscaling method for component 1. Shall be ignored for the Scod parameters.
td_1	4	0..15	Inverse tone mapping curve for component 1. Shall be ignored for the Scod parameters.
ts_2	4	0..2	Postscaling method for component 2. Shall be ignored for the Scod parameters.
td_2	4	0..15	Inverse tone mapping curve for component 2. Shall be ignored for the Scod parameters.
ts_3	4	0	Reserved for ITU|ISO use.
td_3	4	0	Reserved for ITU|ISO use.
Rs	4	0 or 1	Reserved for ITU|ISO use.
Xt	4	0..5	Selects the inverse decorrelation transformation for the L, S or H transformation. See Table B-XXX for the possible values.

Table B-8: JPEG Extensions Marker parameter marker Lcod,Scod and Rcod parameter fields

The Lcod, Scod and Rcod parameters also specify type of the inverse component decorrelation transformation by the Xt parameter. It can be either the ICT, the FCT, the RCT, the identity transformation or the zero transformation generating only zero outputs. If the cc parameter of the Component Decorrelation Control marker is zero, it overrides the Xt value in the Lcod parameter field and replaces the L-transformation by the identity transformation. The Component Decorrelation Control marker does not modify the interpretation of the Rcod or Scod parameters.

The encoding of the Xt parameter is given by the following table:

Value	Transformation to be used
0	The zero transformation, the output is zero regardless of the input. This effectively disables the R or S transformation. Xt shall not be zero for the Lcod parameter set.
1	The identity transformation. The output components shall be identical to the input components.

2	The ICT or the FCT. Both transformations are within a reasonable tolerance identical and the implementation may pick either one. Lossy decoding is sufficient. This transformation shall not be used if the number of components, Nf, equals one.
3	The FCT shall be used. A precise fix-point implementation of the FCT following precisely the guidelines of Annex XXX shall be used for inverse component decorrelation. Lossless coding is desired, and no implementation freedom exists for the decoders. This option shall not be used if Nf is one.
4	The RCT shall be used. This is an integer to integer transformation with no implementation freedom available. Lossless coding is desired. This value shall only be applicable in the Rcod parameter value, and shall not be used if Nf equals one.
5..15	All other values reserved for ITU|ISO purposes.

Table B-8: Interpretation of the Xt parameter fields of the JPEG Extensions parameter marker.

The Rq_i parameter set defines for each component the DCT process to be applied, whether lossless coding is desired, and which quantization matrix is to be used to reconstruct samples. Four Rq_i parameter sets exist of which only the first one or three are used, depending on the value of Nf in the frame header.

The encoding of the Rq_i field is specified in Table B-9:

Parameter	Size (in bits)	Value	Meaning
dct_i	2	0..2	Specifies the type of the DCT transformation to be used. If zero, no DCT transformation is to be applied on the residual data. If one, a lossy DCT transformations following ITU.T Rec.T.81|ISO/IEC 10918 shall be used. If two, the DCT shall be the fixpoint DCT defined in Annex XXX and lossless decoding is desired.
Ns_i	1	0 or 1	Defines whether inverse noise shaping shall be applied to component i. If zero, the inverse noise shaping filter shall be disabled, if one, the filter is enabled. This bit shall only be one if the dct_i bit is zero, i.e. the DCT transformation is disabled.
fq_i	1	0 or 1	Defines where the quantization table for the residual data is located. If one, quantization matrices for the residual data are

			included in the residual stream and qt_i shall be ignored. Otherwise, the quantization matrix for residual component i shall be found in the DQT marker marker in the legacy codestream whose table index is given by qt_i.
tq_i	4	0..3	If fq_i is zero, this parameter defines which quantization matrix of the legacy codestream shall be used to dequantize the residual data of component i.

Table B-9: Interpretation of the Rq_i parameter sets of the JPEG Extensions parameter marker.

Annex C
Multi-Component Decorrelation

Kommentar [tr22]: This should go, at least in parts, into a separate part. What might remain in part-1 is probably the ICT.

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

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

This Annex describes several multiple component transformations that may be deployed as options for the L,S and H transformation. These multiple component transformations are used to improve compression efficiency. They are not related to multiple component transformations used to map colour values for display purposes. The FCT transformation is a fully specified fix-point transformation that may be used for lossy or lossless coding. The ICT is an irreversible approximation of the first and may only be used for lossy coding, though may be more efficient to implement. The RCT is an exactly reversible transformation that, depending on the settings defined in the codestream, acts on the residual data only and thus is only available for the H-transformation; it is used for lossy and lossless encoding in one of the profiles of this Recommendation | International Standard.

Additionally, the L, S and H transformation may be either the zero-transformation, or the identity transformation. The zero transformation generates a zero output regardless of its input values. The identity transformation does not implement a color transformation, but transforms the input samples to integer and clips them to the component range.

5.21 Irreversible inverse multi component transformation (ICT) (normative)

This transformation is applied for reconstructing a low-dynamic range version of the image from the data encoded in the entropy coded data segment following SOS markers. The ICT shall be used to inversely decorrelate samples as L-Transformation

- the number of components in the frame, Nf, equals three,
- and the Component Decorrelation control marker is absent its cc value is equal to one
- and the JPEG Extensions parameter marker is absent or the Xt control parameter in the Lcod parameter set equals two

It shall be used as the S- or H-Transformation in case the Xt parameter of the JPEG Extensions parameter marker segment in the Scod and/or Rcod parameter set equals two.

NOTE – An S or H transformation is not required if the JPEG extensions parameter marker is not present as then no residual data is available.

In the following, set I_0,I_1 and I_2 be the sample values reconstructed from the upsampled components 0, 1 and 2, and let L_0,L_1 and L_2 be the sample values of the reconstructed low dynamic range version of the image. Let h be the number of hidden DCT bits as defined by the h parameter of the JPEG Extensions Specification ('SPEC') marker, or zero if this marker is not present. Then:

$$L_0 = \mathrm{clamp}(I_0+1.40200*(I_2-2^{7+h}),0,2^{8+h}-1)$$

$$L_1 = \mathrm{clamp}(I_0-0.7141362859*(I_2-2^{7+h})-0.3441362861*(I_1-2^{7+h}),0,\ 2^{8+h}-1)$$

$$L_2 = \mathrm{clamp}(I_0+1.772*(I_1-2^{7+h}),0,\ 2^{8+h}-1)$$

NOTE – This transformation is intentionally identical to the YUV to RGB transformation specified in ISO/IEC 10918-6, though its purpose is here slightly different.

5.22 Inverse fixpoint multi component transformation (FCT) (normative)

This transformation is applied in the lossless process for reconstructing a low-dynamic range version of the image from the data encoded in the entropy coded data segment following SOS markers. It shall be used as L-Transformation if:

- the number of components in the frame, Nf, equals three,
- and the Component Decorrelation control marker is absent its cc value is equal to one
- and the JPEG Extensions parameter marker is absent or the Xt control parameter in the Lcod parameter set equals three

It shall be used as the H-Transformation in case the Xt parameter of the JPEG Extensions parameter marker segment in the Scod and/or Rcod parameter set equals three.

The FCT assumes that the reconstructed untransformed sample values are the output of the fixpoint DCT approximation defined in Annex G and thus are integer values that are premultiplied with the factor 16 – or equivalently, the input data to this transformation is given in fixpoint with four fractional bits.

In the following, set I_0,I_1 and I_2 be the sample values reconstructed from the upsampled components 0, 1 and 2, and let L_0,L_1 and L_2 be the sample values of the reconstructed low dynamic range version of the image that are preshifted by 4 bits, i.e. premultiplied by 16. Let h be the number of hidden DCT bits as defined by the h parameter of the JPEG Extensions Specification ('SPEC') marker, or zero if this marker is not present. Then:

$$L_0 = \mathrm{clamp}(\lfloor(2^{13}*I_0+11495*(I_2-2^{7+4+h})+2^{16})/2^{17}\rfloor,0,\ 2^{8+h}-1)$$

$$L_1 = \mathrm{clamp}(\lfloor(2^{13}*I_0-5850*(I_2-2^{7+4+h})-2819*(I_1-2^{7+4+h})+2^{16})/2^{17}\rfloor,0,2^{8+h}-1)$$

$$L_2 = \mathrm{clamp}(\lfloor(2^{13}*I_0+14516*(I_1-2^{7+4+h})+2^{16})/2^{17}\rfloor,0,\ 2^{8+h}-1)$$

NOTE – The fixpoint inverse multi component transformation is an implementation of the irreversible inverse multi component transformation when the input data is represented in fixpoint with four fractional bits and the transformation coefficients are represented with 13 fractional bits. The number of fractional bits has been selected such that 32 bit integer arithmetic is sufficient to implement the transformation. Implementations may substitute the ICT with the FCT where, additionally, input samples I_0 through I_2 are multiplied by 16 before the FCT is applied. The resulting transformations are identical within the desired implementation precision and implementations need not to provide code for both.

~~5.23 Irreversible inverse high-dynamic range multiplicative residual multi component transformation (HCT) (normative)~~

Kommentar [tr23]: This is what is found in the Dolby code currently.
I have no idea why that should not be identical to the ICT, but it isn't. Can't we fix this?

Isn't there an offset shift missing? Is this part of the tone mapping?

Dolby, please add and clarify.

~~This transformation shall be used for the reconstruction of the high-dynamic range chroma components in the multiplicative merging process. It is almost identical to the irreversible multi component transformation, though requires only two chroma components of the inversely tonemapped chroma signals, here denoted by C_1 and C_2. The output signals M_0, M_1 and M_2 are added to the gamma-corrected low-dynamic range image and scaled by the inversely tone-mapped luma signal. For details of the multiplicative process, consult Annex XXX.~~

~~M_0 = 1.3984 * C_2~~

~~M_1 = -0.3054 * C_1 – 0.7038 * C_2~~

~~M_2 = 1.7969 C_1~~

5.24 Reversible inverse multi component transformation (RCT) (normative)

The RCT is a lossless integer to integer transformation that is only available as the H-transformation. It shall be applied if the number of components, Nf, is equal to three and the Xt parameter of the Rcod parameter set of the JPEG Extensions parameter marker segment equals four. The output of this transformation is then added to the output of the inverse tone mapping process to reconstruct the high-dynamic range data.

NOTE – This transformation is applied regardless of whether the component decorrelation control marker is present, or the cc parameter within this marker.

In the following, let I_0,I_1 and I_2 be the reconstructed residual data generated by the residual scan as defined in Annex XXX, and R_0,R_1 and R_2 the reconstructed residual color data to be added to the inversely tonemapped low dynamic range data. Then set:

$$R_1 = I_0 - \lfloor (I_1+I_2)/4 \rfloor$$

$$R_0 = R_1 + I_2$$

$$R_2 = R_1 + I_1$$

NOTE – This transformation is a rough integer approximation of the irreversible inverse multi component transformation that has an exact inverse. It is identical to the RCT defined in ISO/IEC 15444-1. Note further that the RCT does not include a clipping process.

5.25 Inverse identity transformation (normative)

The inverse identity transformation maps its input values to the output without applying an inverse linear decorrelation step. However, it includes transforming data from fixpoint to integer, if applicable, and clamps the output to range. The inverse identity transformation shall be applied as L-Transformation whenever:

- the number of components in the frame, as given by the Nf parameter, is equal to one,
- or a Component Decorrelation control marker is present and cc value is equal to zero
- or a JPEG Extensions parameter marker segment is present and the Xt parameter in the Lcod parameter set is equal to one.

The Inverse identity transformation shall be used as S-Transformation and/or H-Transformation if the Xt parameter of the Scod, resp. Rcod parameter set in the JPEG Extensions parameter marker is equal to one.

Denote the input value of the identity transformation with I_0 and the output value by L_0. The inverse identity transformation comes in two forms, the first of which is picked if the input values are floating point and a floating point DCT implementation is used. In this case,

$$L_0 = \mathrm{clamp}(I_0,0,2^{8+h}-1)$$

In case input samples are represented as fixpoint samples with four fractional bits, as generated by the fixpoint DCT required for lossless coding,

$$L_0 = \mathrm{clamp}(\lfloor (I_0+2^3)/2^{14} \rfloor,0, 2^{8+h}-1)$$

If more than one component has to be transformed, the above equation(s) apply identically to all components.

5.26 Inverse zero transformation (normative)

The zero transformation sets its output to zero regardless of its input. It is only available as an option for the H and S transformation and shall there be used whenever the Xt parameter of the Rcod resp. Scod parameter set of the JPEG Extensions parameter marker segment is zero.

5.27 Irreversible forward multi component transformation (ICT) (normative)

This transformation is used to decorrelate the low-dynamic range versions of the image before encoding the components by a legacy 10918-1 conforming encoder. This transformation is applied in the encoding process to create the legacy codestream if Nf is equal to three and component decorrelation is not disabled by the Component decorrelation control marker. It is also applicable as the encoding equivalent of the R transformation or, for decoding, the S-Transformation.

The forward ICT transformation shall be used as S-transformation if the number of components, Nf, equals three, and the Xt parameter of the Scod parameter set of the JPEG Extensions parameter marker equals two.

The input signals L_0 L_1 and L_2 are already expected to be tone-mapped and hence in the low-dynamic range regime, all three in the range $[0,2^{8+h}-1]$ where h is the number of hidden DCT bits recorded in the JPEG Extensions parameter marker. The output components I_0,I_1 and I_2 are the input of the legacy 10918-1 process.

$$I_0 = 2.9900 * L_0 + 0.58700 * L_1 + 0.11400*L_2$$

$$I_1 = -0.1687358916 * L_1 - 0.3312641084 * L_1 + 0.5*L_2 + 2^{7+h}$$

$$I_2 = 0.5\ L_2 - 0.4186875892*L_1 - 0.08131241085*L_2+2^{7+h}$$

NOTE – This transformation is, up to rouding errors, identical to the inverse of the irreversible inverse multi component transformation and thus identical to the transformation specified in ISO/IEC 10918-5.

5.28 Fixpoint forward multi component transformation (FCT) (informative)

The transformation given in this subclause provides an implementation choice for the ICT, and by that – within the precision of a fixpoint representation – the forward transformation of the FCT defined in subclause C-XX. It then creates fixpoint samples with four fractional bits, directly applicable to the forward fixpoint DCT. It will be applied to generate the legacy codestream data if the number of components in the frame is three and the component decorrelation transformation is not disabled by the Component decorrelation control marker. The forwards FCT can also optionally be used as the encoding equivalent of the H transformation.

The forward FCT shall be used as S-transformation if the number of components, Nf, is three and the Xt parameter of the Scod parameter set of the JPEG Extensions parameter marker is three.

The main purpose of the FCT is to provide a fully specified component decorrelation transformation for lossless coding that is compatible with the ICT used by legacy implementations. Because it is very close to the ICT, encoders may optionally replace the ICT completely by the FCT, and thus are only required to implement one instead of two algorithms.

Denote by L_0,L_1 and L_2 the low-dynamic range sample values of the tone mapped high-dynamic range source image, represented by integer values in the range $[0,2^{8+h}-1]$ where h is the number of hidden DCT bits recorded in the JPEG Extensions parameter marker. Denote by I_0, I_1 and I_2 the output components, represented in a fixpoint format that is preshifted by four bits, i.e. four fractional bits are present. This format is required as input for the fixpoint DCT transformation that is used for the lossless encoding process. Then:

$$I_0 = \lfloor(2449 * L_0 + 4809 * L_1 + 934 * L_2 + 2^8)/2^9\rfloor$$

$$I_1 = \lfloor(-1382 * L_0 - 2714 * L_1 + 4096 * L_2 + 2^{13+7+h} + 2^8)/2^9\rfloor$$

$$I_2 = \lfloor(4096 * L_0 - 3430 * L_1 - 666 * L_2 + 2^{13+7+h} + 2^8)/2^9\rfloor$$

NOTE – This transformation is not exactly the inverse of the inverse FCT, though the additive residual coding process not only includes the residuals due to the finite implementation precision of the DCT, but also due to the precision loss in the FCT.

~~5.29 Irreversible forward high-dynamic range multiplicative residual multi component transformation (HCT) (normative)~~

Kommentar [tr24]: Needs to be clarified by Dolby.

~~TO BE SUPPLIED BY DOLBY.~~

5.30 Reversible forward multi component transformation (RCT) (informative)

The transformation supplied by this subclause decorrelates the additive error residuals of the lossless and lossy additive coding process and is the exact inverse of the RCT defined in subclause XXX. It is applied if the Xt parameter in the Rcod parameter set of the JPEG Extensions parameter marker is four.

Let I_0, I_1 and I_2 be the (integer) error residuals of the additive coding process, and R_0,R_1 to R_2 the output of the transformation which is then quantized and coded by the residual coding process of Annex XXX. Then set:

$$R_0 = \lfloor (I_0 + 2*I_1 + I_2) / 4\rfloor$$

$$R_1 = I_2 - I_1$$

$$R_2 = I_0 - I_1$$

NOTE – This transformation is the inverse of the RCT defined in subclause XXX and, unlike the FCT, exactly invertible. It is a rough approximation of the RCT and identical to the forward RCT defined in ISO/IEC 15444-1.

5.31 Forward identity transformation (normative)

This transformation is used to generate the input of the legacy coding process when the number of components in the frame is one, or the component decorrelation transformation is disabled. It may also be used as the encoding equivalent of the S or R transformation. The transformation is trivial as it simply replaces the output by its input. If the fixpoint DCT transformation, required for lossless coding, is employed, it will additionally multiply its output values by 16 to follow the conventions of this transformation.

This transformation shall be used as S-transformation if the number of components, Nf, is one, and the Xt parameter of the Scod parameter set is one.

Annex D
Tone Mapping Operations

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

Kommentar [tr25]: All this has to go into a separate part once we have the request for subdivision.

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

5.32 Inverse Tone mapping (ITMO) (normative)

This subclause defines the L_i,S and R_i inverse tone mapping processes used by the decoding process. The purpose of the L_i curves is to provide a reasonable prediction of the HDR image based on the legacy codestream, the S-curve generates a luminance exponent for it based on the residual data, and the R curves provide means to adapt the HDR residual data to the output.

For applying an inverse tone mapping operation on component i, the following steps shall be performed:

- Find the JPEG Extensions parameter marker segment in the codestream and within it, locate either the Lcod, Rcod or Scod parameter set, depending on whether an L, S or R inverse tone mapping is to be applied,
- Find the ts_i and td_i parameters within the parameter set, where i is the component index. The S transformation is always applied on the luminance data of the residual image, hence i will be zero for S curves.
- From ts_i, compute the output multiplier μ of the tone mapping curve. This multiplier is zero if ts_i is zero, is one if ts_i is one and 2^{8+Rb} if ts_i is. R_b is the number of additional high-dynamic range bits, this parameter is also found in the JPEG Extensions parameter marker.
- Locate the JPEG Extensions inverse tone mapping marker or JPEG Extensions inverse parametric tone mapping curve marker whose M parameter is equal to td_i. The codestream specifications ensure that there is one and only one such marker.
- If the found marker is a JPEG Extensions inverse tone mapping marker, the output of the inverse tone mapping process is defined to be

  $$H_i \quad = \mu\times D_k[I_i] / 65535$$

  where D_k is the lookup table included in the marker segment, I_i is the input sample value and μ the multiplier computed above.
- If the found marker is a JPEG Extensions parametric inverse tone mapping curve, locate the inverse tone mapping function f by interpreting the curve type parameter t in this marker. Also locate the parameters P_1 through P_4 to fully define f. The applicable values of t are defined in Table B-XXX. The output of the inverse tone mapping curve is then defined as

  $$H_i \quad = \mu\times f(I_i/2^{8+h})$$

  where μ is the output scale parameter derived from the ts_i parameter and h is the number of hidden DCT bits, specified in the JPEG Extensions parameter marker.

5.33 Forward Tone mapping (FTMO) (informative)

This Recommendation | International Standard does not define a way how to find a forward tone mapping curve, nor how to generate its inverse required for the process described in subclause XXX. In general, the forward tone mapping process generates for each high-dynamic range sample value X_i a low dynamic sample value L_i, which is then encoded by a coding process defined in ITU.T Rec. T.80 | ISO/IEC 10918-1. It is, however, recommended for optimum coding efficiency in the lossless process that the algorithm for determining the sample value L_i and the inverse tone mapping curve are selected such that the error residual $R_i = X_i\text{-}D_k[L_i]$ is as small as possible.

A suitable algorithm would, for example, first construct a forward tone mapping curve F_k depending on the rendering intent of the image to be encoded. In a second step, an approximate inverse of F_k would be encoded in the JPEG Extensions Inverse Tone Mapping Marker, i.e. D_k would be selected such that $|D_k[F_k[x]] - x|$ is as small as possible. The table D_k would then be encoded in the codestream, and the sample values L_i would be computed by $L_i=F_k[X_i]$.

TODO: Tone Mapping in the Multiplicative (Dolby) process.

Kommentar [tr26]: To be provided by Dolby.

Annex E
Splitting and Merging Low Dynamic Range and Residual Image

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

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

5.34 Merging LDR and residual image to a HDR image (normative)

This subclause specifies the reconstruction process of a high-dynamic range image from a LDR image and a residual image decoders shall follow. This process consists of the following steps, see also Figure XXX:

- Reconstruction of an image from the ITU.T Rec.T.81|ISO/IEC 10918-1 legacy codestream and the refinement codestream if the number of hidden DCT bits h is larger than zero. The h parameter is found in the JPEG Extensions parameter marker, the process how to reconstruct a common LDR image from the legacy codestream and the refinement extensions is specified in Annex XXX. If lossless decoding is desired, the fixpoint DCT specified in Annex YYY shall replace the DCT process specified in ITU.T Rec.T.81|ISO/IEC 10918-1. The output of this step consists of one or three components.
- The upsampling process specified in Annex YYY shall be followed to generate samples for all positions on the sample grid. Upsampling is only valid for lossy coding. For lossless coding, no upsampling shall take place and the subsampling factors of all components shall be (1,1).
- The L-Transformation shall be applied to inversely decorrelate the image components. This transformation is either the identity transformation, the ICT or the FCT. The specific transform to be used depends on the number of components, of the Component decorrelation control marker if present, and the Lcod parameter set of the JPEG Extensions parameter marker. The output of this process are one or three components of integer samples in the range $[0,2^{8+h}\text{-}1]$.
- Each of the output components shall be inversely tonemapped according to the ts_i and td_i parameters of the Lcod parameter set of the JPEG Extensions parameter marker, implementing the L_i Luts of Figure XXX. The process of inverse tone mapping is defined in Annex XXX. The output of this process are one or three sample value(s) per sample grid point, denoted by L_i.
- The decoder shall compute for each set of L_i the value Lu. Lu is the output I_o of the S-Transformation, which is specified by the Xt parameter of the Scod parameter set in the JPEG Extensions parameter marker. Outputs I_1 and I_2 are dischared and not used.
- The noise floor shall be added to the value Lu, giving the scale Lp. The noise floor value NL is encoded in the JPEG Extensions parameter marker:

$$Lp = Lu + NL$$

- The residual image shall be reconstructed following the specifications of Annex XXX. The output of this process is optionally inversely DCT transformed, depending on the dct_i parameter of the Rq_i parameter set of the JPEG Extensions marker, and dequantized according to the fq_i and qt_i parameters of Rq_i.
- Depending on the Ns_i marker, an inverse noise shaping filter is applied to the residual component i.
- Residual data is upsampled to the common sample grid. Upsampling factors shall be (1,1) if lossless coding is desired, hence if the dct_i parameters are zero.
- The output of the residual decoding process are one or three integer sample values R_i per sample grid point.
- An inverse tonemapping process shall be applied to the first sample R_0 using the S-inverse tone mapping curve, specified by the Scod parameter set of the JPEG Extensions marker. The result of this process is the scale factor μ.

- Inverse tone mapping by the R-Luts are applied to the reconstructed residual sample values. The R-inverse tonemapping operation is specified by the Rcod parameter set of the JPEG Extensions parameter marker. The output sample values of this process are called O_i.
- The output of the inverse residual tone mapping shall be multiplied with the luminance prediction value Lp computed from the low dynamic range and refinement data giving the scaled residual data P_i:

$$P_i \quad = \quad Lp \times O_i$$

- Inverse color decorrelation shall be applied to the P_i data by means of the H-transformation. This transformation is either the zero transformation, the identity transformation, the ICT, FCT or RCT. It is specified by the Xt parameter of the Rcod parameter set of the JPEG Extensions parameter marker. The output of this process are the inversely decorrelated prediction errors Q_i.
- The high dynamic range output F_i, i.e. the final output of the decoding process, is reconstructed from H_i, the predicted high dynamic range signal, the multiplier µ and the inversely decorrelation prediction errors Q_i as follows:

$$F_i \quad = \quad \mu \times H_i + Q_i$$

- If the Oc parameter of the JPEG Extensions parameter marker is one, the F_i sample values are converted to floating point values according to the specification of Annex XXX. Otherwise, F_i are the final output of the decoding process.

Profiles may restrict the choice of some of the parameters, and decoders are only required to support those subsets of parameter values that correspond to the profiles they claim to support. Profiles and the corresponding restrictions of the parameter set are defined in Annex XXX.

5.35 Splitting HDR data into LDR and residual data (informative)

Kommentar [tr27]: This should probably go into a separate "Recommendations and Guidelines" section.

This Recommendation|International Standard does not define a normative algorithm for computing a low dynamic range and a residual image from a given high dynamic range image. Nevertheless, some guidelines will be given that demonstrate how the coding mechanisms defined by this Recommendation|International Standard can be deployed to implement image coding in the high dynamic range regime.

An image coding process in which a low dynamic range image is extended to high dynamic range by means of an additive residual error can be realized by disabling the S and R tone mapping processes, setting the S-transformation to zero and selecting a noise floor of 1.0. This process has the advantage that the absolute error in the high-dynamic range process is easily controllable and may even be zero, i.e. lossless coding is possible. The following steps should be taken to implement this coding process:

- The Xt value of the Scod parameter is set to 0, disabling the S transformation.
- The NL parameter is set to 1.0 indicating a unit noise floor. By this choice, the multiplier Ld will always be constant 1.0.
- The ts_0 parameter of the Scod parameter set is set to one, and the td_0 parameter of the Scod parameter set selects a parametric tone mapping curve whose t value is one. This setting generates an S-tonemapping whose output is constant 1.0 multiplier µ.
- The ts_i parameters of the Rcod parameter set are all set to two, and the td_i parameters are set to the M value of a JPEG Extensions parametric tone mapping curve whose t value is two. This ensures that the R-tone mapping implements an identity map.
- The L-lookup tables are chosen as the inverses of a suitable tone mapping process. For example, the encoder might apply the Reinhard tone mapping operator [insert reference here] to generate a low-dynamic range version of the input HDR image, and compute a numerical inverse of the tone mapping operation which is then stored as lookup table in the JPEG Extensions Inverse Tone Mapping marker segment. The td_i parameters of the Lcod parameter set will then include the M value of this marker, and the ts_i parameters are set to two, indicating that the output of the L-tables are scaled to the full range of the residual image.
- The low dynamic range image generated by forwards tone mapper is directly compressed by a legacy ITU.T Rec.T.81 | ISO/IEC 10918-1 encoder.
- The residual image is generated by inversely dequantizing the low-dynamic range DCT coefficients $C_{i,j}$ and applying an inverse fixpoint DCT transformation to them. The resulting components are then upsampled and inversely decorrelated. The difference between these samples and the original image defines then the residual signal Q_i. This essentially requires reproducing parts of the decoding process at the encoder.

- The difference signal Q_i is decorrelated by the ICT or RCT implementing the (approximate) inverse of the H transformation, generating the R_i signal. The selected transformation is indicated in the Xt parameter of the Rcod parameter set.
- The R_i signal is quantized and encoded by the residual coding process defined in Annex XXX. While coding this signal by the legacy ITU.T Rec.T.81|ISO/IEC 10918-1 specifications is possible, it is not recommended since the signal consists of already decorrelated noise which does not compress well by the legacy coding mechanism.
- The output rate can be controlled by adjusting the quantization matrix of the LDR coding process and the quantization parameters qt_i in the Rq_i parameter set of the residual coding process. For low qualities, it is typically most efficient to disable the residual signal completely and defining the quality of the HDR image entirely by the legacy quantization matrix. For higher qualities, the legacy quantization matrix should be left constant and should define relatively fine quantization, i.e. the entries in the quantization matrices should be small. Quality is then controlled by the high-dynamic range quantization parameter.

Lossless coding can be implemented by a refinement of the above coding process, namely

- The H transformation must be the RCT, i.e. the Xt parameter of the Rcod parameter set is set to four.
- The DCT to compute the decoder generated LDR data at encoder side is the fixpoint DCT. For that, the Lf parameter of the JPEG Extensions parameter marker is set to one.
- No DCT process is used for coding the residual data, i.e. the dct_i parameters in the Rq_i parameter sets are set to zero.
- The L-Transformation is the FCT or the identity transformation. For that, the Xt parameter of the Lcod parameter is set to three or one. If Xt is equal to one, an Component decorrelation control marker with cc=0 is also written to the stream, indicating legacy decoders that no component decorrelation is desired.
- The qt_i values of the Rq_i parameter set is set to a quantization table that contains at least at its entry #63 a one, indicating that residual coefficients are not to be quantized.

NOTE – the choice of the forwards DCT, the L-curves or the quantization matrix of the legacy codestream is irrelevant as far as lossless decoding is concerned. It influences only the compression performance of the encoder, though not the ability to reconstruct the original image samples without loss.

By disabling the residual coding completely and additionally setting the L-lookup tables to identity, an extension of the legacy DCT process for bit depths up to 12 bits per sample is possible. This allows a lightweight, minimum effort coder for images of up to 12 bits per sample. Coding performance of this extension is not quite as good as the performance of the residual process defined above. For this, only refinement coding is used.

EDITOR's NOTE: Description of Dolby's encoding process should go here, additionally with its advantages and when to pick it.

Kommentar [tr28]: TBD by Dolby. Need to understand how coding works here.

~~Annex F~~
~~Lossy Coding of Residual Data~~

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

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

~~TO BE SUPPLIED BY DOLBY. This is likely only a 10918-1 codestream encapsulated in an APP11-marker. Can we change the marker?~~

Kommentar [tr29]: Strike this. Not required and captured by Annex G (then Annex F).

Annex G
Lossless, near lossless and lossy coding of residual data

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

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

This Annex describes methods for coding and decoding residual data; a lossy process that encapsulates residual data in a codestream that follows the earlier Recommendation|International Standard, and a lossless or near-lossless process that works most effective on additive residual data. Since the residual coding process specified here can be implemented without loss, it shall be used for lossless coding of still images. The near-lossless process for coding of high-dynamic range images and is then combined with the quantization of the residual data specified in Annex XXX and, optionally, noise shaping, specified in the same Annex.

Entropy coded residual data is encapsulated in JPEG Extensions Residual Data markers. Whenever such markers are present, additional residual scans over the image provide a correction signal that enhances the high-dynamic range image or enables lossless coding by encoding the residual error caused by the non-ideally invertible FDCT and FCT decorrelation transformation.

The residual codestream shall be formed by concatenating the payload data of all JPEG Extensions residual markers in the order they appear in the codestream. The resulting residual stream consists optionally of a frame header, followed by at least one scan over the residual data. Depending on whether a frame header is present, the entropy decoding proceeds either by one of the processes defined in ITU.T Rec.T.81|ISO/IEC 10918-1, or uses a modified progressive decoding algorithm defined in sublcause XXX-1.

If the frame header is present, it shall be identical to the frame header of the legacy codestream and the scan type and thus the decoding process is defined by the type of the SOF marker included in the residual stream. The residual stream then follows the syntax of the abbreviated codestream format specified in ITU.T Rec.T.81|ISO/IEC 10918-1, though the scan type is restricted to either the baseline sequential, extended sequential or progressive scan, i.e. only SOF_0 through SOF_2 markers shall be used to start the frame header. If no Huffman tables are specified by DHT marker segments in the residual codestream, those defined in the legacy codestream shall be used for decoding. If no quantization tables are specified in the residual codestream, the Rq_i parameter sets in the JPEG Extensions parameter marker define the quantization matrices in the legacy codestream to be used. In the case a frame header is present, the residual codestream decodes to 8-bit sample values, as indicated in the frame header.

Kommentar [tr30]: Is this correct or can it be different?

Kommentar [tr31]: This excludes refinement scans for residual scans.

If no frame header is present, then the dimensions and subsampling factors of the residual stream are identical to that of the frame defined by the legacy codestream, but a modified decoding process is used for decoding the entropy coded data in the scan. As for legacy scan patterns, Huffman tables and quantization matrices may be omitted and are then taken from the legacy codestream, where in specific the Rq_i parameter set defined which quantization matrix to use. The modified scan process for decoding the data is defined in subclause G-2. The residual scan decodes in this case to 8+Rb sample values, thus to the full bitdepth of the HDR image.

Unlike the legacy coding process or the refinement coding process, residual coding may or may not transform its coefficients by a DCT; the DCT for residual coding is enabled by the dct_i field of the Rq_i parameter set of the JPEG Extensions parameter marker. If a frame header is included in the residual codestream, the DCT process shall be enabled by setting all dct_i parameter to one. If no frame header is included in the residual scan and the modified decoding process of subclause G-2 is used, the DCT shall be disabled by setting dct_0 to zero for all components included in the scan.

The reason for optionally disabling the DCT is that it is not exactly invertible and is thus unsuitable for lossless coding. The second reason is that the residual signal may, depending on the coding process, consist mostly of noise and additional transformations do not provide compression gains. If the DCT is disabled, entropy coding and decoding operates directly in the spatial domain on the residual signal, and so does the quantization and noise shaping specified in Annex XXX. Nevertheless, the residual signal is also separated into 8×8 blocks that are aligned with the blocks the DCT transformation, and these blocks are scanned in the order defined in ITU.T Rec.T.81|ISO/IEC 10918-1.

5.36 Modified decoding process of residual error signals (normative)

This decoding process shall be used if the residual codestream included in the JPEG Extensions residual markers does not include a frame header. The structure of the codestream included in the residual markers is depicted in Figure XXX-1.

[Tables/ misc]	Scan header	ECS_0	$[RST_0]$	…	ECS_{last}	$[RST_{last}]$	[Tables /misc]	Scan header	ECS_0	$[RST_0]$	…	ECS_{last}	$[RST_{last}]$

Figure G-1: Syntax of the Contents of the Residual Data Continuous Codestream

Data in the entropy coded segments ECS_0 to ECS_{last} shall be decoded by a modified progressive or successive approximation scan algorithm as defined in Annex G of ITU.T Rec.T.81|ISO/IEC 10918-1:

- The DC decoding step shall be skipped, i.e. subclause G.1.2.1 of ITU.T Rec.T.81|ISO/IEC 10918-1 is not effective.

- The AC decoding algorithms defined in subclauses G.1.2.2 and G.1.2.3., specifically figures G.3 and G.7 also apply if Ss = 0, i.e. if the start of the spectral selection is zero. For example, if Ss = 0, the flow charts G.3 and G.7 start with K=.-1 and the first coefficient decoded is ZZ(0), the top left block edge.
- Consequently, only the AC Huffman tables shall be used for decoding the entropy coded data segment. The DC Huffman table indices in the scan header are irrelevant. If no Huffman table definition is found in the residual codestream, the Huffman tables defined in the frame or scan header of the legacy codestream apply.
- As no frame header is written for this scan type, the dimensions of the frame and the subsampling factors are defined in the frame header of the legacy codestream. The precision of the samples reconstructed by this stream is 8+Rb, the number of bits in the HDR image. The Rb parameter is found in the JPEG Extensions parameter marker.
- The dct_i parameters of the Rq_i parameter field in the JPEG Extensions parameter marker shall be zero, indicating that no DCT transformation is applied to reconstruct the sample values.

NOTE – Since the scan type defined in this Annex decodes residual data that has not been transformed by a DCT, no specific DC coefficient that would require special treatment is present; instead, all residual error coefficients follow an AC statistics that allows a consistent coding mechanism. The progressive encoding mechanism of AC coefficients defined in ITU.T Rec.T.81 | ISO/IEC 10918-1, especially the block-skip mode, allows efficient encoding of data that contains many zeros. A similar block skip mode is not available for DC coefficients which was removed for this reason. Note that the modified progressive coding mode is used for residual data regardless of the frame type indicated by the legacy codestream.

5.37 Coding of residual error signals (informative)

Encoding of non-DCT transformed residual data consists of two steps: First, the data is encoded by one or several modified progressive or subsequent approximation scans. The resulting codestream, with or without Huffman table definitions and scan headers, but without the frame header, is split into one or several data chunks encapsulated into JPEG Extensions Residual Data markers which are then embedded into the legacy codestream.

In the first step, the residual data is first separated into 8×8 blocks that are aligned to the DCT blocks of the LDR image. Note that the residual data is untransformed, but potentially quantized and noise-shaped. This data is then encoded by one or many modified progressive scans, and optionally additional subsequent approximation scans, forming one or many entropy coded segments. The modifications made to the entropy coding algorithm are identical to those discussed in subclause XXX, namely DC coding is skipped and AC coding also applies to the residual coefficient at block position 0,0. Optionally, the modified progressive and subsequent approximation scans may redefine Huffman tables to improve coding efficiency of the residual data, otherwise the Huffman tables of the legacy codestream apply.

The output of the first step is a continuous codestream that optionally includes one or several DHT marker segments, and one or many scans each consisting of a SOS marker followed by an entropy coded segment. The layout of this codestream follows then Figure H-1 in the previous subclause.

In the second step, this continuous codestream is separated into chunks, each of which no longer than 2^{16}-9 bytes. These chunks are inserted as payload data into JPEG Extensions Residual data marker segments which are injected into the legacy codestream.

NOTE – It is recommended, though not required, to define Huffman tables that are specific to the residual coding scans as this helps to improve the coding efficiency. Since spectral selection does not select any part of a spectrum, but rather selects the spatial position of the residual data within an 8×8 block in the image, using this feature of progressive or subsequent approximation scans is not particularly useful, though valid. Successive approximation scans may provide a subsequent refinement of the residual data, though.

Encoding of DCT transformed residual data follows in its first step the coding procedure specified in ITU.T Rec.T.81|ISO/IEC 10918-1, except that only the baseline sequential, extended sequential and progressive scan types are available. Unlike the above DCT-less coding mechanism, the residual codestream shall here include a frame header, though need not to include DQT or DHT markers; if DQT or DHT markers are missing, the corresponding markers from the legacy codestream are used, where the Rq_i parameter set of the JPEG Extensions parameter marker selects the effective quantization matrices then.

The resulting codestream is then separated into chunks no longer than 2^{16}-9 bytes each, and packaged into JPEG Extensions residual markers. These markers are finally injected into the Tables/Misc section of the legacy codestream.

Annex H
Entropy Coding of Refinement Data

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

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

This Annex defines the procedures for decoding and encoding of images using an extended bitdepths in the DCT domain. If this coding method is used, the entropy coded data visible to legacy decoders describes an image with a precision of eight bits per sample, which is extended by a refinement coding procedure described in this Annex to up to 12 bits. Refinement coding can be combined with other coding methods, for example the residual coding method defined in Annex G, then allowing even higher bit depths of up to 16 bits per sample.

Refinement coding is signaled by a non-zero value of the h parameter in the JPEG Extensions Specification marker; the value of h describes the additional bit precision becoming available due to the refinement scan described in this Annex. It may take values from zero – refinement scan disabled – to four, resulting in 12 bits precision for the output of the DCT process.

If the refinement scan is enabled by setting $h \neq 0$, the quantization and inverse quantization process defined in ITU.T Rec. T.81 has to be modified to include the additional bits, and the entropy coding and decoding procedures includes additional scans over the image to represent such bits. The DCT transformation and multi component decorrelation transformation remain unchanged except that the DC level shift has to be modified to include the additional bits. That is, while subclauses F.1.1.3 and F.2.1.5 of ITU.T Rec.T.81|ISO/IEC 10918-1 specify a level shift of 2^7 for a sample precision of eight bit, the level shift shall be replaced by a shift of 2^{7+h} if the refinement scans are included in the decoding process. A decoder that chooses to ignore refinement scans will continue to operate with the level shift as indicated in the legacy standard, but will not take advantage of the enhanced bit precision.

NOTE – Additional care must be taken to avoid overflow conditions as the range of the input and output data is extended. These considerations were included in the fixpoint DCT transformation defined in Annex XXX which shall be used for lossless reconstruction. Refinement coding may also be used for lossy reconstruction of images and then provides a good compromise between speed and precision.

5.38 Modifications of the inverse quantization process for residual decoding (normative)

The regular dequantization process in the absence of refinement scans multiplies the entropy decoded DCT coefficient value $S_{i,j}$ at block position i,j with the quantization matrix entry $q_{i,j}$ to get the DCT coefficient $C_{i,j}$. In the presence of a residual scan, h additional bits $T_{i,j}$ are decoded by the refinement scan that shall be included as h least significant bits in the reconstruction of the DCT coefficient $C_{i,j}$ as follows:

$$C_{i,j} \quad = \quad (2^h \times S_{i,j} + T_{i,j}) \times q_{i,j}$$

NOTE – An equivalent implementation strategy for the above inverse quantization procedure is to decode the data from the legacy scans represented by the entropy coded data following the SOS markers as if the point transformation parameter would be set to Al+h (or 0+h) instead of Al (or zero). Al is the point transformation parameter of the scan header, see subclause A.4 and B.2.3 of ITU.T Rec.T.81| ISO/IEC 10918-1. Data decoded by the refinement scan uses an unmodified decoding procedure and is then automatically placed in the least significant bits of $S_{i,j}$.

5.39 Modifications of the quantization process for residual coding (informative)

The quantization procedure generates in the presence of refinement scans from the DCT coefficients $C_{i,j}$ two data sets, the legacy quantized data $S_{i,j}$ which will be encoded by the entropy coding methods provided by ITU.T Rec.T.81|ISO/IEC 10918-1, and the refinement data $R_{i,j}$ which undergoes refinement coding. $S_{i,j}$ and $T_{i,j}$ are computed from the DCT coefficients $C_{i,j}$ and the quantization matrix $q_{i,j}$ as follows:

$$X_{i,j} = \lfloor C_{i,j}/q_{i,j} + 0.5 \rfloor$$

$$S_{0,0} = \lfloor X_{0,0}/2^h \rfloor \qquad T_{0,0} = X_{0,0} - 2^h \times S_{0,0}$$

$$S_{i,j} = \mathrm{sign}(X_{i,j}) \times \lfloor |X_{i,j}|/2^h \rfloor \qquad T_{i,j} = X_{i,j} - 2^h \times S_{i,j} \qquad \text{for } (i,j) \neq (0,0)$$

NOTE – The division and rounding algorithm for computing $S_{i,j}$ generates a "fat zero" for AC coefficients, i.e. a dead zone of twice the size of a regular quantization bucket. DC coding does not follow this convention to avoid drift errors when encoding. The rounding conventions picked here are intentionally identical to the rounding procedure used when encoding data with the successive approximation of the progressive scan defined in ITU.T Rec.T.81|ISO/IEC 10918-1, and this is, in fact, one possible implementation strategy for refinement coding: The legacy codestream is created by an encoder as if the successive approximation parameter would be Al+h (or h) instead of Al (or zero), the remaining bits are then encoded by the refinement scan defined in this Annex.

5.40 Decoding process for entropy coded refinement data (normative)

This subclause defines the decoding procedure for the refinement data $T_{i,j}$ that extends the precision of the DCT coefficients by h bits. Entropy coded refinement data is present whenever the h parameter of the JPEG Extensions Specification marker is non-zero.

In a first step, the decoder shall concatenate the payload data of all JPEG Extensions refinement ('FINE') markers in the order they appear in the bitstream to form a single consecutive bitstream. The payload data is defined by the marker body without the marker segment length, the JPEG Extensions Common Identifier and the JPEG Extensions Type Identifier.

The resulting stream consists of a series of scan headers and entropy coded data following the syntax of ITU.T Rec.T.81|ISO/IEC 10918-1, except that neither a frame header nor DQT markers shall be present. Restart markers, if indicated in the legacy codestream, shall however be included, following the syntax defined in the legacy standard, and DHT markers re-defining the Huffman code may be present as well. The scans included in this stream are decoded by a modification of the successive approximation decoding algorithm using Huffman decoding defined in Annex G of ITU.T Rec.T.81|ISO/IEC 10918-1, regardless of the frame type indicated in the legacy codestream (i.e. regardless of whether the frame type is baseline, extended sequential or progressive).

[Tables/ misc]	Scan header	ECS_0	$[RST_0]$	…	ECS_{last}	$[RST_{last}]$	[Tables /misc]	Scan header	ECS_0	$[RST_0]$	…	ECS_{last}	$[RST_{last}]$

Figure G-1: Syntax of the Contents of the Refinement Data Continuous Codestream

The entropy coding algorithm used for refinement decoding is identical to the successive approximation decoding defined in Annex G of ITU.T Rec.T.81|ISO/IEC 10918-1 except that the scan coefficients ZZ'(K) have to be initialized before starting the first refinement scan by the entropy decoded data of the legacy codestream shifted by 2^h bits, i.e.

$$ZZ'(K) \quad = \quad 2^h \times ZZ(K)$$

where ZZ(k) is the k-th position of the zig-zag scan defined in Figure 5 of ITU.T Rec.T.81|ISO/IEC 10918-1. The refinement data $T_{i,j}$ after decoding is then defined by the output ZZ'(K) of all successive approximation scans included in the JPEG Extensions refinement markers:

$$T_{(i,j)(k)} \quad = \quad ZZ'(K) - 2^h \times ZZ(K)$$

NOTE – An alternative implementation strategy would first decode the legacy portion of the codestream using either the baseline, sequential or progressive decoding procedure, but with a point transformation/successive approximation parameter of Al+h (or h) for each scan included in the legacy codestream. The refinement scan extends these scans on the same scan buffer by using a successive approximation scan on the same scan buffer while using the successive approximation parameter exactly as inidicated in the scan headers included in the JPEG Extensions Refinement marker. Note further that the refinement scan always uses a successive approximation scan pattern, regardless of the frame type of the legacy codestream.

5.41 Encoding of refinement data (informative)

This Annex specifies an encoding process of refinement data that is compatible to the decoding process defined in the above subclause. In the first step, the scan pattern ZZ'(K) is initialized with the quantized data encoded in the legacy codestream, $S_{i,j}$, plus the refinement data $T_{i,j}$:

$$ZZ'(K) \quad = \quad 2^h \times ZZ(K) + T_{(i,j)(k)}$$

where ZZ(K)(k) is the k-th position in the zig-zag scan over the data in the legacy codestream. Then, one or several successive approximation scans following the specifications in Annex G of ITU.T Rec.T.81|ISO/IEC 10918-1 encode the h least significant bits of the data, i.e. the Al parameter of the scans generated by the refinement scans will vary between h-1 and zero.

The resulting entropy coded data, together with the scan headers defining the scan pattern, but without any frame header or DQT markers, are split into chunks of at most 2^{16}-9 bytes each and inserted as payload data of the JPEG Extensions Refinement markers into the legacy bitstream.

NOTE – An alternative implementation would first use a legacy scan process with a point transformation of Al+h (or h) bits to encode the LDR data, and would then run a successive approximation scan on the same data buffer with an unmodified parameter of Al. Note further that the refinement encoding procedure always uses the successive approximation scan of the progressive coding mode, regardless of the frame type of the legacy codestream.

Annex I
Discrete Cosine Transformation

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

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

This Annex defines a fixpoint version of the inverse DCT transformation implementations of the lossless profile shall follow. If lossless reconstruction is not required, alternative implementations of the DCT process are possible; nevertheless, the algorithm described here define a DCT implementation that is conforming to ITU.T Rec.T.81|ISO/IEC 10918-1 and that may even be used for the lossy decoding process.

5.42 Overview on the Inverse Fixpoint DCT Transformation (normative)

The Inverse Fixpoint DCT Transformation takes integer samples $Q_{i,j}$ in the form of an 8×8 block, and generates from that fixpoint samples $Y_{i,j}$ also organized in an 8×8 block. The input samples $Q_{i,j}$ are integers generated by the dequantization procedure described in ITU.T Rec.T.81|ISO/IEC 10918-1, the output samples $Y_{i,j}$ are fixpoint numbers with four fractional bits, i.e. integers premultiplied by 16, that are suitable for the Inverse Fixpoint Decorrelation Transformation (FCT) defined in Annex C. The DC level shift specified in subclauses F.1.1.3 and F.2.1.5 of the legacy standard are included in the DCT algorithm specified by this Annex, thus forwards and inverse level shift need not to be performed if the algorithms shown here are followed. Modifications for refinement coding are also included.

The steps of the algorithms are as follows:

- Level-shift the DC coefficient $Q_{0,0}$ and scale all coefficients to the precision of the FCT; this step implements the inverse DC level shift of subclause F.2.1.5 of ITU.T Rec.T.81|ISO/IEC 10918-1 as part of the DCT transformation:

  $X_{0,0} = 2^4 \times Q_{0,0} + 2^{7+h+3+4}$ where h is the number of hidden DCT bits specified by the h parameter of the JPEG Extensions marker.

  $X_{i,j} = 2^4 \times Q_{i,j}$ for $(i,j) \neq (0,0)$
- Run a one dimensional inverse fixpoint DCT process over all columns of the 8×8 block
  - by setting $A_k = X_{k,l}$ where k is the row and l is the column index
  - following the procedure of the one-dimensional DCT process taking the vector A as input and generating the vector B as output
  - backwards scaling B and generating one column of the 8×8 block Z from B

    $Z_{k,l} = \lfloor (B_k + 2^8)/2^9 \rfloor$
  - repeating the above steps for all columns $l \in [0,7]$
- Run a one dimensional inverse fixpoint DCT process over all rows of the 8×8 block
  - By setting $A_k = Z_{j,k}$ where j is the row and k is the column index
  - Following the procedure of the one-dimensional DCT process taking the vector A as input and generating the vector B as output
  - Backwards scaling the output B and inserting this as one row into the output matrix Y

    $Y_{j,k} = \lfloor (B_k + 2^{8+3})/2^{9+3} \rfloor$
  - Repeating the above steps for all rows $j \in [0,7]$

NOTE – the DCT process defined in this subclause is designed such that 32 bit wide variables are sufficient to implement all steps for $h \leq 4$, i.e. for at most 12 bits sample precision without causing overflows.

5.43 One dimensional inverse fixpoint DCT transformation (normative)

This subclause defines a one dimensional inverse fixpoint DCT process that is a building block of the two-dimensional fixpoint DCT transformation defined in the subclause above. It takes an eight-dimensional vector A_k of fixpoint numbers prescaled by 4 bits as input, and generates an eight-dimensional vector B_k of fixpoint numbers prescaled by 9+4=13 bits as output.

- Set $Z_1 = (A_2 + A_6) \times 277$
- Set $T_2 = Z_1 - A_6 \times 946$
- Set $T_3 = Z_1 + A_2 \times 392$

- Set T_0 = $(A_0+A_4) \times 512$
- Set T_1 = $(A_0-A_4) \times 512$
- Set T_{10} = $T_0 + T_3$
- Set T_{13} = $T_0 - T_3$
- Set T_{11} = $T_1 + T_2$
- Set T_{12} = $T_1 - T_2$
- Set Z_4 = $A_7 + A_3$
- Set Z_5 = $A_5 + A_1$
- Set Z_6 = $(Z_4 + Z_5) \times 602$
- Set Z_7 = $(A_7 + A_1) \times -461$
- Set Z_8 = $(A_5 + A_3) \times -1312$
- Set Z_9 = $Z_4 \times -1004 + Z_6$
- Set Z_{10} = $Z_5 \times -200 + Z_6$
- Set T_{30} = $A_7 \times 153 + Z_7 + Z_9$
- Set T_{31} = $A_5 \times 1051 + Z_8 + Z_{10}$
- Set T_{32} = $A_3 \times 1573 + Z_8 + Z_9$
- Set T_{33} = $A_1 \times 769 + Z_7 + Z_{10}$
- Set B_0 = $T_{10} + T_{33}$
- Set B_7 = $T_{10} - T_{33}$
- Set B_1 = $T_{11} + T_{32}$
- Set B_6 = $T_{11} - T_{32}$
- Set B_2 = $T_{12} + T_{31}$
- Set B_5 = $T_{12} - T_{31}$
- Set B_3 = $T_{13} + T_{30}$
- Set B_4 = $T_{13} - T_{30}$

NOTE – The above algorithm implements an (unscaled) one dimensional DCT transformation that is, by itself, not exactly invertible. The lossless process will correct errors from the forwards transformation by encoding residuals between this DCT and the ideal transformation in a residual scan.

5.44 Overview on the Fixpoint DCT Transformation (informative)

This subclause provides an overview and implementation guideline for a forwards reversible fixpoint DCT. Many other implementation choices are possible, even for the lossless profile, provided that the residual is computed according to inverse fixpoint DCT specified above.

The forwards fixpoint DCT transform takes an 8×8 block of sample values $Y_{i,j}$ as input and generates from that a DCT transformed signal $X_{i,j}$ which is also organized in an 8×8 block. In the implementation presented here, the input samples are fixpoint numbers premultiplied by 16, i.e. have 4 fractional bits. Level shifting as specified in subclause F.1.1.3 of the legacy standard is included in the described algorithm and need not to be performed. The output coefficients $Q_{i,j}$ are also represented in fixpoint and have 9+4+3=16 fractional bits and form the input for the quantization procedure defined in ITU.T Rec.T.81|ISO/IEC 10918-1. Quantization has then to be modified and/or adapted to work with fixpoint input.

The number of fractional bits in both the input and output samples have been selected such that the DCT transformation can be implemented on 32 bit architectures without causing overflow conditions for input samples of up to 12 bits precision. This corresponds to the maximum allowable choice of h = 4.

The forwards fixpoint DCT consists of the following steps:

- Extract row k from the source signal $Y_{k,l}$ and denote this row-vector by B_l.

  $B_l = Y_{k,l}$

- Run the one dimensional forwards fixpoint DCT algorithm on B_k described in the next subclause resulting in an output vector A_k.
- Scale the output vector A_k back into a fixpoint representation and insert it as k^th row into the intermediate block $Z_{k,l}$:

  $Z_{k,l} = \lfloor (A_k+2^8)/2^9 \rfloor$

- Repeat the above steps for all rows $l \in [0,7]$
- Extract column j from the intermediate block $Z_{i,j}$ and use this as input vector B_i for the one dimensional fixpoint DCT

  $B_i = Z_{i,j}$

- Run the one dimensional forwards fixpoint algorithm on B_i giving A_i.
- Insert the output vector as column j into the intermediate output DCT block $X_{i,j}$

  $X_{i,j} = A_i$

- Repeat the above steps for all columns $i \in [0,7]$ to fill the entire block $X_{i,j}$.
- Generate the final output by level shifting the DC coefficient and copying all other coefficients to $Q_{i,j}$. This step also implements the level shift specified in subclause F.1.1.3 of the legacy standard, which therefore need not to be performed explicitly.

  $Q_{0,0} = X_{0,0} - 2^{7+h+9+4+3+3}$ where h is the number of hidden DCT bits defined by the h parameter of the JPEG Extensions specification marker.

  $Q_{i,j} = X_{i,j}$ for $(i,j) \neq (0,0)$

The output block $Q_{i,j}$ contains now DCT coefficients preshifted by 9+4+3=16 bits. Quantization can now be performed by dividing $Q_{i,j}$ by $2^{16}q_{i,j}$ and rounding to the nearest integer, where $q_{i,j}$ is the quantization matrix for the component currently transformed.

5.45 One dimensional fixpoint DCT transformation (informative)

This subclause provides an implementation example for an unscaled one dimensional fixpoint DCT that is used as a building block of the two-dimensional fixpoint DCT specified in the above subclause. Alternative implementations are possible, even for the lossless compression path. The input to this algorithm is an eight-dimensional vector B_i of fixpoint coefficients preshifted by four bits, the output a vector A_i of eight fixpoint numbers preshifted by 4+9=13 bits.

- Set $T_0 = B_0 + B_7$
- Set $T_1 = B_1 + B_6$
- Set $T_2 = B_2 + B_5$
- Set $T_3 = B_3 + B_4$
- Set $T_{10} = T_0 + T_3$
- Set $T_{12} = T_0 - T_3$
- Set $T_{11} = T_1 + T_2$
- Set $T_{13} = T_1 - T_2$
- Set $A_0 = (T_{10} + T_{11}) \times 512$
- Set $A_4 = (T_{10} - T_{11}) \times 512$
- Set $Z_1 = (T_{12} + T_{13}) \times 277$
- Set $A_2 = Z_1 + T_{12} \times 392$
- Set $A_6 = Z_1 - T_{13} \times 946$
- Set $T_{20} = B_0 - B_7$

- Set T_{21} = $B_1 - B_6$
- Set T_{22} = $B_2 - B_5$
- Set T_{23} = $B_3 - B_4$
- Set T_{32} = $T_{20} + T_{22}$
- Set T_{33} = $T_{21} + T_{23}$
- Set Z_2 = $(T_{32} + T_{33}) \times 602$
- Set Z_3 = $(T_{20} + T_{23}) \times -461$
- Set Z_4 = $(T_{21} + T_{22}) \times -1312$
- Set Z_5 = $T_{32} \times -200 + Z_2$
- Set Z_6 = $T_{33} \times -1004 + Z_2$
- Set A_1 = $T_{20} \times 769 + Z_3 + Z_5$
- Set A_3 = $T_{21} \times 1573 + Z_4 + Z_6$
- Set A_5 = $T_{22} \times 1051 + Z_4 + Z_5$
- Set A_7 = $T_{23} \times 153 + Z_3 + Z_6$

NOTE – The above transformation is, by itself, not exactly invertible, neither that nor the choice of the forwards DCT matters for lossless reconstruction as long as the inverse fixpoint DCT is implemented exactly as shown.

Annex J
Quantization and Noise Shaping of the Residual Image

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

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

This Annex and its subclauses describe the procedure by which residual data coefficients decoded by the no-DCT process defined in Annex XXX are reconstructed before merging the reconstructed data with the LDR data following the procedures of Annex A. The reconstruction after entropy decoding consists of two steps: An optional adaptive low-pass filter removing the high-frequency noise generated by the optional noise-shaping filter at encoder side, and an inverse quantization step that reconstructs the residuals from the encoded coefficients. If noise shaping is turned off and the quantization bucket size is set to one, lossless reconstruction of the residuals is possible, enabling lossless compression in combination with the fixed point DCT and the FCT decorrelation transformation.

5.46 Adaptive Low-Pass Filtering and Inverse Quantization (normative)

This subclause specifies the adaptive low-pass filtering and inverse quantization algorithm decoders shall follow for dequantization of the residual data of component i if the Ns_i bit in the JPEG Extensions Specification Marker is set to one. If it is set to zero, the inverse quantization process defined in the next subsection applies to non-DCT data.

The input data to the adaptive low-pass filtering and inverse quantization is an 8×8 block of reconstructed integer residual coefficients $C_{n,m}$ coming from the non-DCT residual data entropy decoding process defined in Annex XXX. This data block is separated again into 2×2 subblocks $S^k_{i,j}$ where k is the subblock index (k=0..15) and i,j the position within the subblock (i,j=0..1). One has:

$$S^k_{i,j} \quad = \quad C_{(i+k \bmod 4, j+k/4)}$$

The quantization step size q_R that applies to inverse quantization of the residual coefficients is taken to be the entry q_{63} of one of two possible quantization matrices whose indices are given by the Rq_i parameter set in the JPEG Extensions Specification marker. In the first step, the decoder shall compute the 16 averages over the inversely quantized subblocks by:

$$Avg^k \quad = \quad \left\lfloor \left(\Sigma_{i,j=0..1} S^k_{i,j} \times q_R + 2\right)/4 \right\rfloor$$

In a second step, coefficients are inversely quantized by multiplying the entropy decoded coefficient value $C_{n,m}$ with the quantization bucket size q_R, but those coefficients that differ from the average in their subblock by not more than one quantization bucket are replaced by the average. That is:

$R_{(i+k \bmod 4, j+k/4)} = Avg^k$ if $|Avg^k - q_R \times S^k_{i,j}| < q_R$

$R_{(i+k \bmod 4, j+k/4)} = q_R \times S^k_{i,j}$ otherwise

NOTE – Noise shaping at the encoder side moves the quantization noise into higher frequencies. The adaptive averaging step described here removes the noise pattern introduced by the encoder whenever the noise level is below an expected noise threshold that is due to the noise shaping process. This averaging reduces the spatial resolution of the image but improves on average the amplitude resolution of the residual data and thus reduces potential visual artifacts due to quantization in the spatial domain.

5.47 Inverse Quantization (normative)

In case noise shaping is turned off by setting the Ns_i parameter of the JPEG Extensions Specification marker to zero, inverse dequantization of non-DCT transformed residual data shall be implemented by multiplying the decoded coefficient value $C_{i,j}$ with the quantization bucket size q_R. The parameter q_R shall be taken from entry q_{63} of the quantization matrix defined by the Rq_i parameter set in the JPEG Extensions parameter marker.

The residual data $R_{i,j}$ shall then be reconstructed by $R_{i,j} = q_R \times C_{i,j}$ where $C_{i,j}$ is the decoded symbol generated by the entropy decoder.

NOTE – Quantization matrices are encoded in DQT markers that can be found in the tables of the legacy codestream if the fq_i parameter in the Rq_i parameter set is zero, or the tables of the continuous codestream of the JPEG Extensions Residual data markers if fq_i is one.

5.48 Quantization and Noise Shaping of Residual Data (informative)

If the Ns_i bit in the JPEG Extensions Specification marker is set, the encoder will include a noise shaping process to reduce the quantization noise of component i at the cost of reduced spatial resolution of the non-DCT transformed residual data signal. This subclause describes a noise shaping and quantization process that is compatible to the normative adaptive low-pass filter defined in subclause XXX. For that, let $R_{i,j}$ be the residual signal, organized in an 8×8 block that is to be encoded by the non-DCT coding method defined in Annex XXX. Let SO(k) be the scan order over the block defined in figure J-1. At the start of the scan, set the cumulative quantization error, E, to zero.

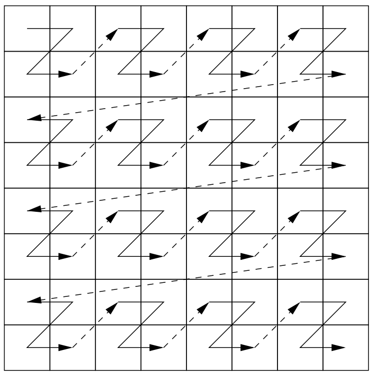

Figure J-1: Scan Pattern for the Noise Shaping of the Residual Data

Then, for each position k = 0..63 in the scan pattern, compute the following:

- Set $V=R_{SO(k)}+E$, i.e. V is the source signal plus the cumulative error.
- Set the quantized signal to $C_{SO(k)} = \mathrm{sign}(V)\times\lfloor |V|/q_R \rfloor$
- Update the cumulative error $E = E+R_{SO(k)}-q_R\times C_{SO(K)}$
- Repeat the above steps for k=0..63 in the order of the scan pattern of Figure J-1.

The output of this process is the quantized signal $C_{i,j}$ which is entropy encoded by the algorithm specified in Annex XXX.

NOTE – The rounding convention selected here creates a "fat zero" (deadzone) around the zero bucket that is twice the size of all other buckets. This is intentional and reduces the amount of noise encoded in the residual scan. Other rounding conventions may be used, and may, depending on the image, also provide better results.

5.49 Quantization of Residual Data (informative)

This subclause describes the quantization process of residual data that is used if the Ns_i parameter of the JPEG Extensions Specification marker is zero, and hence noise shaping is disabled. This process is suitable for lossless coding if the quantization bucket size q_R is equal to one. The parameter q_R is found as follows: For the component 0, find the quantization matrix whose index is given by the Lt parameter of the JPEG Extensions specification marker; for components 1 and 2, find the quantization matrix with index Ct. Then set q_R to the entry q_{63} of the found matrix; all other entries are irrelevant for quantization of residual data. The symbols C_{ij} to be coded by the lossless and near lossless entropy coding process for residual data defined in Annex XXX are then found by dividing the residual signal R_{ij} by q_R and rounding to zero:

$$C_{ij} = \mathrm{sign}(R_{ij}) \times \lfloor |R_{ij}| / q_R \rfloor$$

NOTE – Other rounding conventions than the one shown here may be possible and may, depending on the source image, even provide better results.

Annex K
Encoding and decoding of floating point data
(This Annex forms a normative and integral part of this Recommendation | International Standard.)

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

This Recommendation | International Standard allows the representation of images whose sample values are IEC 60559 half-float numbers. The final output of the decoding process are floating point numbers when the Oc parameter of the JPEG Extensions parameter field is one, and a re-interpretation process is necessary to generate floating point numbers from the integer outputs F_i produced by the decoder specification in Annex XXX.

Kommentar [tr32]: I'm not sure how Dolby handles this process. Probably they use floating point output of the decoding process directly. If so, we need to make Oc a two-bit field which also requires floating point. Or we ignore floating point completely.

In a first step, the integer sample values shall be represented by bit pattern using a binary representation. Note that this is the representation of integers on many, but not all computer architectures. In a second step, the bit-patterns are re-interpreted as floating point numbers using the IEC 60559 (IEEE 754) representation.

NOTE – It can be seen that this re-interpretation of bit-patterns on non-negative integers is equivalent to a piecewise linear approximation of the exponential function $f(x) \approx (2^x - 1)$. Negative numbers never appear as the output of this process due to the output clamping of the L and H transformation.

Annex L
Encoding and Decoding of Images with Alpha Channels
(This Annex forms a normative and integral part of this Recommendation | International Standard.)

To be done.

Annex M
Profiles and Levels
(This Annex forms a normative and integral part of this Recommendation | International Standard.)

To be done.

Annex N
Guidelines

To be done.

Annex O
References

To be done.