INTERNATIONAL STANDARD

ISO/IEC 15444-2

First edition
2004-05-15

Information technology — JPEG 2000 image coding system: Extensions

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

Reference number
ISO/IEC 15444-2:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

Foreword

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

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

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

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

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

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

— *Part 1: Core coding system*

— *Part 2: Extensions*

— *Part 3: Motion JPEG 2000*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Compound image file format*

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

— *Part 12: ISO base media file format*

The following parts are under preparation:

— *Part 8: Secure JPEG 2000*

— *Part 10: Extensions for theree-dimensional data and floating point data*

— *Part 11: Wireless JPEG 2000*

Annex M

JPX extended file format syntax

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

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate. This annex describes an extension to ITU-T Rec. T.800 | ISO/IEC 15444-1 that can be used alone or in conjunction with any of the other extensions in this Recommendation | International Standard.

M.1 File format scope

This annex defines an optional file format that applications may choose to use to contain JPEG 2000 compressed image data. This format is an extension to the JP2 file format defined in ITU-T Rec. T.800 | ISO/IEC 15444-1 Annex I. While not all applications will use this format, many applications will find that this format meets their needs. However, those applications that do implement this file format shall implement it as described in this entire annex.

This annex :

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

M.2 Introduction to JPX

As defined in ITU-T Rec. T.800 | ISO/IEC 15444-1 Annex I, the JP2 file format provides a method by which applications can interchange images files in such a way that all conforming readers can properly interpret and display the image. However, some applications require extensions to the JP2 file format that would prevent the file from being properly interpreted by a conforming reader. For example, an image encoded in a CMYK colourspace will not be properly interpreted by a conforming JP2 reader.

Placing these non-compatible extensions into a JP2 file will introduce confusion in the marketplace, as a situation will exist where some readers can interpret some JP2 files but not others. While this confusion is inevitable when you consider the global set of all applications profiles, it must be avoided in some applications, such as on the consumer desktop.

Thus this annex defines a second file format to be used in applications that require functionality or data structures beyond those defined in the JP2 file format. This file format is called JPX.

M.2.1 File identification

JPX files can be identified using several mechanisms. When stored in traditional computer file systems, JPX files should be given the file extension ".jpf" (readers should allow mixed case). On Macintosh file systems, JPX files should be given the type code 'jpx\040'.

However, if a particular JPX file is compatible with the JP2 reader specification (as indicated by placing the code 'jp2\040' in the compatibility list in the File Type box), then the writer of that file may choose to use the extensions for the JP2 file format, as specified in I.2.1 of the JP2 file format, for that particular file. This will maximize the interoperability of that file without sacrificing file indication (as the BR field in the File Type box shall be 'jpx\040' for files completely defined by this Recommendation | International Standard).

In the JP2 file format, the File Type box provides information that a file reader can use to determine if it is capable of reading the file. This box is also present in other JPEG 2000 family file formats.

M.2.2 File organization

As in JP2 files, a JPX file represents a collection of boxes. The binary structure of a file is a contiguous sequence of boxes. The start of the first box shall be the first byte of the file, and the last byte of the last box shall be the last byte of

the file. Many boxes are defined by this Recommendation | International Standard. In addition, other Recommendations | International Standards may define other boxes for use within JPX files. However, all information contained within a JPX file shall be in the box format; byte-streams not in the box format shall not be found in the file.

The binary structure of a box in a JPX file is identical to that defined in the JP2 file format (ITU-T Rec. T.800 | ISO/IEC 15444-1, I.4).

M.2.3 Greyscale/Colour/multi-component specification

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

- defining additional colourspaces for the Enumerated method (M.11.7.3.1);
- defining a method for vendors and other standards bodies to register additional Enumerated colourspaces (M.7.3.1);
- defining a new method to allow the use of any input ICC profile (the Any ICC method, M.11.7.3.2);
- defining a new method to allow vendors to define unique codes for colourspaces independently from a registration authority (the Vendor Colour method, M.11.7.3.3);
- allowing the use of extensions of the multiple component transformation and non-linearity transformation extensions within the codestream (see Annexes J and K).

M.2.4 Specification of opacity information

The JPX file format specifies two extensions for specifying opacity information. First, the file format allows for opacity channels to be stored in a separate codestream from the colour channels of the image. In many image editing applications, the colour data and the opacity data are edited separately, and thus it is useful to allow those channels to be stored in separate codestreams. This is described in the compositing layer architecture description in M.5.

Secondly, the JPX file format allows for the specification of fully transparent samples through a chroma-key. The file format specifies an array of sample values, one from each colour channel. Image locations that contain that combination of sample values shall be considered fully transparent. For example, the chroma-key may be specified as red = 134, green = 92 and blue = 47. Any location with this combination of red, green and blue sample values shall be considered fully transparent. This is defined in the specification of the Opacity box in M.11.7.6.

M.2.5 Metadata

In addition to specifying how the image data shall be stored, this Recommendation | International Standard defines, in Annex M, a number of metadata elements. These elements specify information such as how the image was created, captured or digitized, or how the image has been edited since it was originally created. This Recommendation | International Standard also includes the ability to specify intellectual property rights information, as well as the content of the image, such as the names of the people and places in the image.

In addition to the metadata defined within this Recommendation | International Standard, other forms of XML-based metadata and descriptions (for example those defined by Annex N), may be embedded in a JPX file within XML boxes.

Furthermore, the MPEG-7 binary box (as defined in M.11.19) may be used to store MPEG-7 binary (BiM) format metadata.

M.2.6 Storage of a codestream within JPX

In JP2, the entire codestream is required to be stored in a contiguous portion of the file. However, this restriction can be problematic for some applications. Image editing applications, for example, may desire to modify a single tile of the image and to write the modified tile to the end of the file without rewriting the file. Image servers or internet applications may desire to split the image up into multiple files on different disks or spread the codestream across the internet. The JPX file format allows these features by allowing the codestream to be divided into fragments. The fragmentation of codestreams is described in M.4.

M.2.7 Combining multiple codestreams

In addition to specifying the rendered result as a result of decompressing a single codestream and properly interpreting the colourspace of that codestream as specified in the JP2 file format, the JPX file format allows for multiple codestreams to be combined to produce the rendered result. These codestreams can be combined in a combination of two ways: compositing and animation. This is further described in M.5.

M.3 Greyscale/Colour/Palette/multi-component specification architecture

The JPX file format builds on the flexible colour architecture defined in the JP2 file format. Colourspace specifications are specified within Colour Specification boxes, as originally defined in JP2. However, JPX extends this box (within the binary structure limitations defined in JP2) to allow other methods to be used to specify the colourspace, and to allow a reader to select from the different colourspace specifications found in a single file when interpreting an image.

M.3.1 Extensions to the Colour Specification box header

In JP2, the APPROX and PREC fields were reserved for future use; JP2 writers are required to write default values into these fields. In JPX, these fields are defined and can be used by a JPX reader to make intelligent processing choices.

In both JP2 and JPX, a single file may contain multiple representations of the colourspace of the image. For example, a JPX image may contain an enumerated value, a complex ICC profile, and a Restricted ICC profile. These multiple methods are included to maximize interoperability as well as to provide optimized access. In this example, the enumerated value allows quick recognition, the complex ICC profile allows accurate interpretation using a full ICC engine, and the Restricted ICC profiles allows less complex readers to get a "good enough" result. Specifically, the Restricted ICC profile in this example is an approximation of the complex ICC profile. This approximation is specified within the Colour Specification box, and allows a reader to make trade-offs when interpreting the image.

The Colour Specification box also allows the writer to associate a precedence with each method. This information specifies a default priority in which the multiple colourspace specifications should be considered when selecting which specification will be used to interpret the decompressed code values.

However, the use of both the approximation and precedence information is beyond the scope of this Recommendation | International Standard. Applications are free to consider both pieces of information together and to define their own priorities for selecting which colourspace method to use when interpreting the image. For example, in a fast-preview mode, where speed is more important then quality, an application may desire to use the Restricted ICC profile even if it can use the more complex profile.

M.3.2 Extensions to the Enumerated method

The JPX format defines enumerated values for several additional colourspaces. In addition, this Recommendation | International Standard defines a mechanism by which vendors or other standards bodies can register additional values for the EnumCS field in the Enumerated Method. In general, there are no implementation requirements for these additional defined or registered colourspaces. Requirements for interpretation of specific spaces are defined within the file conformance definition.

In addition, the data structures for the enumerated method have been extended to allow for the specification of parameters that define exactly how that particular colourspace was encoded in the file. For example, the CIELab colourspace, as defined by ITU-T Rec. T.42 specifies six parameters that specify the exact encoding range and offsets of the stored data. To properly interpret a CIELab image, the decoder must know this information and apply those parameters to the decoded image data. Enumerated parameters are specified individually for each enumerated colourspace that requires parameters. However, many colourspaces do not require additional parameters, and thus additional parameters are not defined for those colourspaces. For colourspaces that do specify additional parameters, default values may be defined (as for example are done for the definitions of CIELab and CIEJab). If the entire block of additional parameters are not contained within the Colour Specification box, then the default values shall be used; however, if any additional parameter is specified for a particular Colour specification box, then all additional parameters must be specified for that Colour Specification box.

M.3.3 Any ICC method

In the JP2 file format, the Restricted ICC method was defined, allowing images to be encoded in a wide range of RGB and greyscale spaces. However, many colourspaces, such as CMYK and CIELab spaces, cannot be represented using the restricted set of ICC profiles allowed by the Restricted ICC method. JPX lifts this restriction by defining a separate method to allow any legal ICC input profile to be embedded in the file. This is a separate colour method than the Restricted ICC method, which is also legal in a JPX file. Applications shall not use the Restricted ICC METH value for embedding non-Restricted ICC profiles.

M.3.4 Vendor Colour method

While this Recommendation | International Standard defines a method by which new colourspaces can be registered, the registration method is not appropriate for use for defining codes for vendor-specific or private colourspaces. To allow the quick identification of these colourspaces, the JPX standard defines an additional colourspace specification method, called the Vendor Colour method. This method is very similar to the Enumerated method, except that instead of using 4-byte integer codes, the Vendor Colour method uses UUID's. These UUID values are generated by application developers when the definition of a particular colourspace is created.

It is legal to specify a Vendor Colour value in every JPX file. However, no reader is required to correctly interpret the image based solely on the Vendor Colour method. If an image writer desires to maximize interoperability outside the scope of the target application, it should use additional colour methods in the file (such as the Any ICC and Restricted ICC colour methods).

M.3.5 Palettized colour

Palettized colour is specified and works exactly as defined in the JP2 file format. A palettized image would contain a Palette box, which specifies the transformation from one to many components. The many components generated by the palette are then interpreted by the rest of the colour architecture as if they had been stored directly in the codestream.

M.3.6 Using multiple methods

The JPX file format allows for multiple methods to be embedded in a single file (as in the JP2 file format) and allows other standards to define extensions to the enumerated method and to define extended methods. This provides readers conforming to those extensions a choice as to what image processing path should be used to interpret the colourspace of the image.

If the file is to be JP2 compliant, the first method found in the file (in the first colourspace specification box in the JP2 Header box) shall be one of the methods as defined and restricted in the JP2 file format. However, a conforming JPX reader may use any method found in the file.

M.3.7 Interactions with the decorrelating multiple component transformation

The specification of colour within the JPX file format is independent of the use of a multiple component transformation or non-linearity correction within the codestream (the MCT, MCC, MCO and NLT markers specified in A.3.7, A.3.8, A.3.9 and A.3.10, respectively). The colourspace transformations specified through the sequence of Colour Specification boxes shall be applied to the image samples after the reverse multiple component transformation and reverse non-linearity correction has been applied to the decompressed samples. While the application of these decorrelating component transformations is separate, the application of an encoder-based multiple component transformation will often improve the compression of colour image data.

M.4 Fragmenting the codestream between one or more files

Another important feature of the JPX file format is the ability to fragment a single codestream within a single file or across multiple files. This allows applications to implement such features as:

- edit an image, resaving the changed tiles to the end of the file;
- distribute the image across several disks for faster access;
- distribute the image across the internet, allowing only certain customers access to the high quality or high resolution portions of the codestream;
- reuse of headers from within a codestream across multiple codestreams (to minimize file overhead when storing similar codestreams within the same JPX file).

Fragmentation in JPX works by specifying a table of pointers to the individual fragments. Each pointer specifies three things:

- The file in which the fragment is contained. Because multiple fragments across multiple codestreams may be stored in the same file, the format encapsulates all filename/URL data into a table (the Data Reference box). Each fragment specification then references an entry in the data reference table.
- The offset of the first byte of the fragment within the file specified. This offset is with respect to the first byte of the file (byte 0) and points directly to the first byte of codestream data for that fragment; it does not point to the start of a box containing that fragment.
- The length of the fragment, in bytes.

While the fragment offset does not point to the start of the box, any codestream data contained within a JPX file must be encapsulated in a box. If a codestream is contained within the JPX file in contiguous form, then it shall be encapsulated within a Contiguous codestream box as specified in the JP2 file format and M.11.8; the file shall not also contain a Fragment table representing that contiguous codestream. If the codestream is contained within the JPX file in multiple fragments, then the codestream shall be encapsulated within one or more Media Data boxes (defined in M.11.9).

Figure M.1 shows how a fragment table is used to specify a complete codestream in an example JPX file when all fragments are stored within the file itself. Because the data reference table was empty (no external references), it may not exist in the JPX file. Boxes other than the fragment related boxes are not explicitly shown.

In this example, the codestream is divided into four fragments. The dark lines on the bottom of the Media Data boxes show the portion of the contents of that Media Data box that represents the fragment. Two of those fragments are contained within the same Media Data box. For each fragment, the fragment list specifies the offset and the length of each fragment. The offset values point to the first byte of the codestream data, relative to the beginning of the file. For example, the first fragment is at the start of the contents of a Media Data box. The offset to that fragment is to the first byte of the contents of the box, not to the start of the box header. The length values specify the length only of the actual codestream data for that fragment.

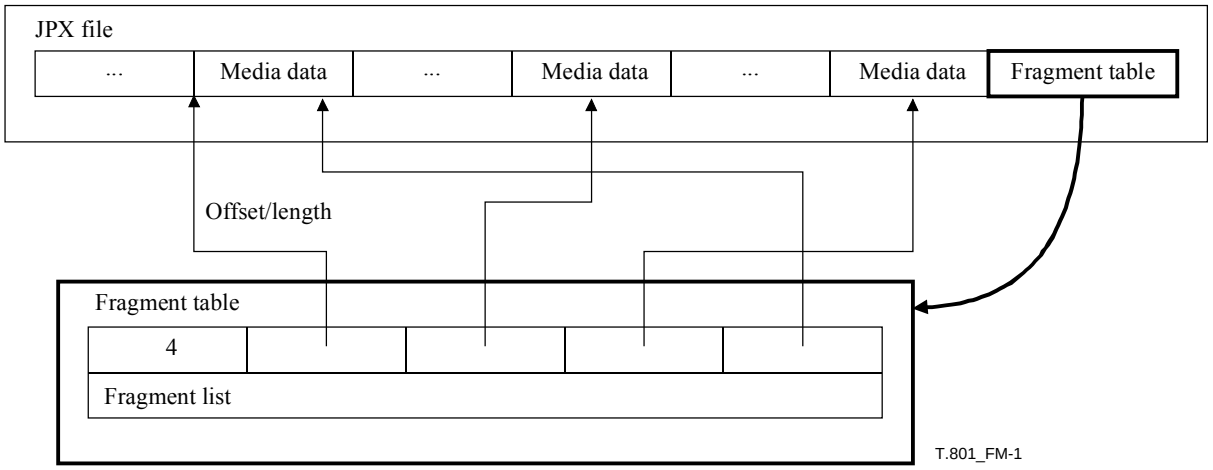

Figure M.1 – Example fragmented JPX file where all fragments are in the same file

To extract the complete codestream from the file, an application must locate the fragment table for that codestream in the file, and then parse the offsets and lengths from the fragment list. The application could then simply seek to the locations specified by the offsets and read the amount of data specified by the length.

Figure M.2 shows how a fragment table is used to specify a complete codestream in an example JPX file when some of the fragments are stored outside the file. In this case, the file shall contain a Data Reference box.

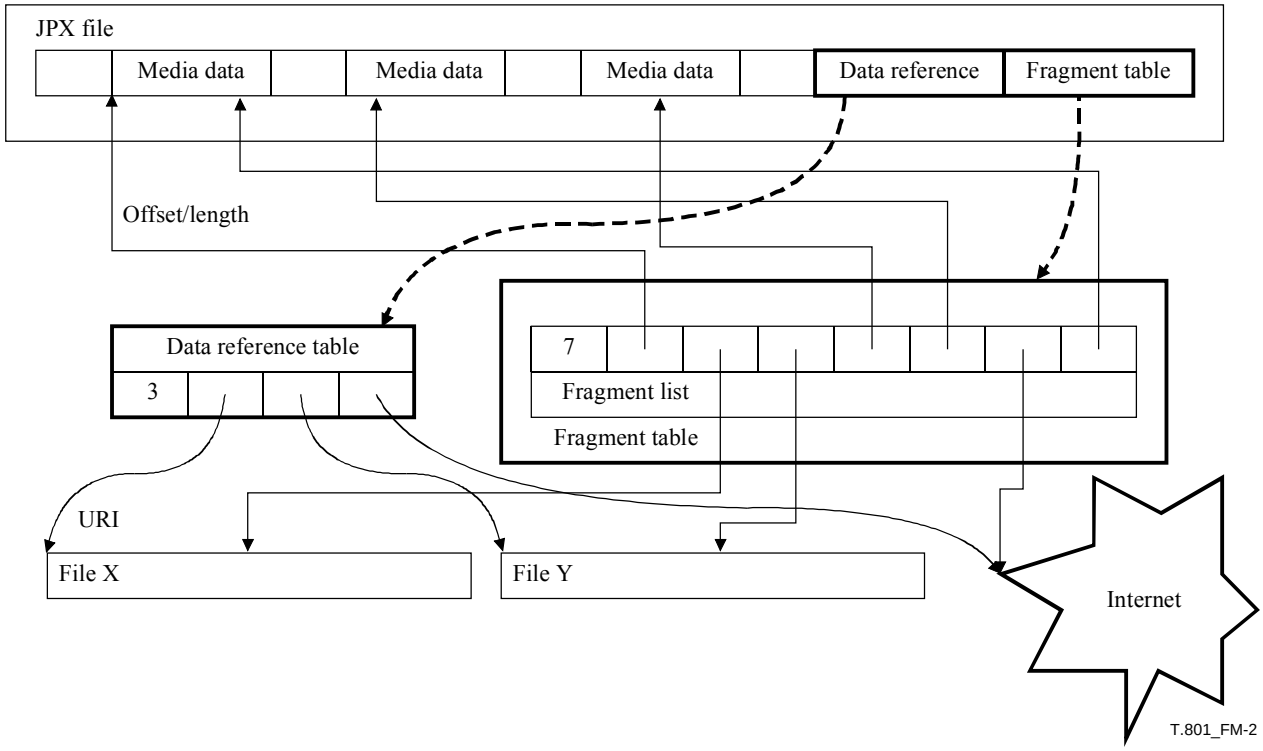

Figure M.2 – Example fragmented JPX file where some fragments are stored in other files or resources

In this example, two of the fragments are stored in separate but locally accessible files, and one of the fragments is stored across the internet.

M.5 Combining multiple codestreams

In the simplest JPX file, a rendered result is generated by decompressing a single codestream to one or more image channels and properly interpreting them in the context of the associated colourspace specification and optional opacity specification. This mode of operation is identical to that offered by JP2 except that JPX offers a wider range of colourspaces and specification methods. In addition to this, JPX offers a rich set of methods for combining multiple codestreams to form the rendered result.

In a JPX file it is possible to store multiple JP2 style "images." In the context of a single JPX file, these separate images are referred to as compositing layers. Each compositing layer comprises a set of channels that an application should treat as a unit for the purpose of rendering. The JPX file format includes syntax for specifying how the compositing layers in a file should be combined by the reader application to produce the rendered result. Both simple still image compositing and animation are supported.

In a JPX file, it is additionally possible to store a single image (or compositing layer) using multiple codestreams. This, for example, allows the separation of RGB components from an opacity channel component. This would permit a single opacity channel to be reused in other compositing layers in the JPX file. The file format also includes syntax for specifying how codestreams are combined to form compositing layers including how the codestreams should be spatially registered against each other.

In a JPX file, metadata can be associated with codestreams and compositing layers independently. Metadata may be shared between multiple codestreams.

M.5.1 Mapping codestreams to compositing layers

To facilitate the mapping of multiple codestreams to single compositing layers, the JPX format separates header fields defined for JP2 into two logical groups: those specific to a single codestream are grouped into a Codestream Header box (M.11.6) and those specific to a compositing layer are grouped into a Compositing Layer Header box (M.11.7). The process of mapping codestream components to channels is exemplified in Figure M.3. Multiple codestreams are combined via a Codestream Registration box (M.11.7.7) to provide the complete set of components for a compositing layer. A Component Mapping box (ITU-T Rec. T.800 | ISO/IEC 15444-1, I.5.3.5) in the Codestream Header box is used to specify how the components of any given codestream are mapped to channels. Interpretation of these channels is specified in the Compositing Layer Header box either using a Channel Definition box (M.11.7.5) or an Opacity box (M.11.7.6). The Opacity box is a new option in the JPX file format that provides an additional method for specifying compositing layers with simple compositing or that use chroma-key opacity.

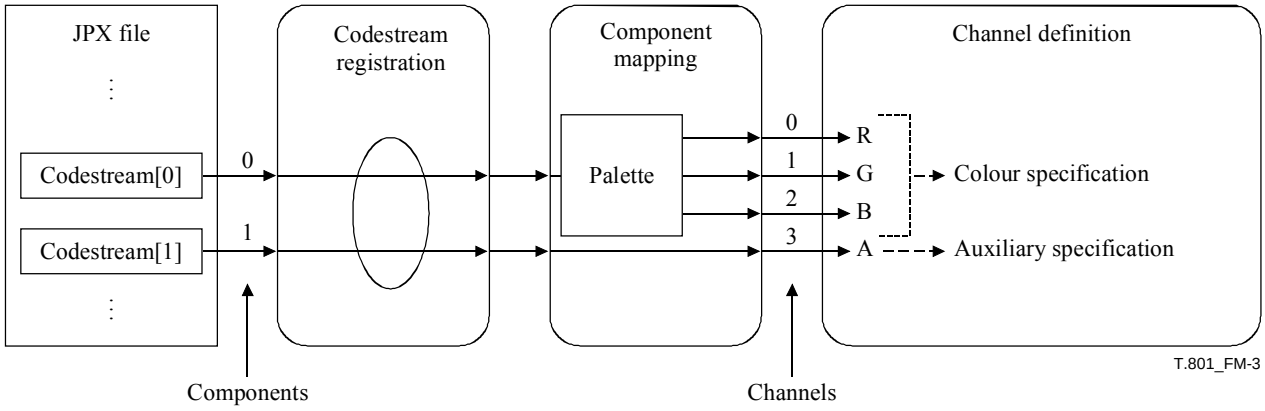

Figure M.3 – Example combination of two codestreams into a single compositing layer

M.5.1.1 Establishing a sequence order for compositing layers

A sequence order for compositing layers is required for any subsequent rendering or animation of the file. In the simplest case there are no Codestream Registration boxes in the file. In this case, which includes the case where all of the headers are present as global defaults, codestreams map directly to compositing layers and the compositing layer sequence order is given by the sequence order of the codestreams in the file.

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

M.5.1.2 Establishing an order for channels in a compositing layer

Where multiple codestreams are combined it is necessary to establish a sequence order over the combined set of channels generated from them. This sequence order is required so that specific channel numbers can be associated with channel definitions when using a Channel Definition box.

Channel ordering is zero based and performed independently for each compositing layer present in the file. The first n channels (numbered 0 through to $n - 1$) within the scope of a particular compositing layer are contributed by channels defined by the first Codestream Header box referenced in the layer's Component Registration box (where we assume this codestream generates n channels), the next m by the next codestream referenced in the layer's Component Registration box, and so on. Within each codestream, channel ordering is determined by the order of entries in the codestream's Component Mapping box, or by the order of components in the codestream if no Component Mapping box is present.

M.5.2 Sharing header and metadata information between codestreams and compositing layers

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

M.5.2.1 Default headers and metadata

When a JP2 Header box is found in a JPX file, the header information in that box shall be used as global default header information for all codestreams and compositing layers within the file. If a Codestream Header box includes boxes that also appear in the JP2 Header box, then these headers shall override the global headers for that specific codestream. If a Codestream Header box contains other boxes that do not appear in the global JP2 Header box, then these boxes shall augment the global header information for that specific codestream. Similarly, if a Compositing Layer Header box includes boxes that appear in the JP2 Header box, then these shall override the global headers for that specific compositing layer. If a Compositing Layer Header box contains other boxes that do not appear in the global JP2 Header box, then these boxes shall augment the global header information for that specific compositing layer.

Any metadata box, including IPR, XML and UUID boxes defined by the JP2 specification as well as additional metadata boxes defined by this specification, found at the file level (not contained within any other superbox) shall also be considered as containing global default information. As with header boxes, these global defaults may be overridden or augmented on a per codestream or per compositing layer basis by the inclusion of corresponding boxes in the codestream or layer header superboxes.

M.5.2.2 Cross-referencing headers and metadata

A Codestream Header box or Compositing Layer Header box may also contain a cross-reference to a box stored in another location. This cross-reference is very similar to the Fragment Table box used to specify the location of a fragmented codestream. In fact, a Cross-Reference box uses the same data structures as the Fragment Table box, with the addition of a field to specify the type of box being referenced. If a Codestream Header box or Compositing Layer Header box contains a Cross-Reference box, the reader shall consider the box pointed to by the reference as if it had been physically contained with the header. Cross-Reference boxes may be used equally for header and metadata.

M.5.2.3 Labelling and association

The Association box may be used to share a label (or other metadata) between codestreams and compositing layers by the inclusion of a Number List box within the Association box. A Number List box refers, by number, to a set of entities in the file. If the first box within an Association box is a Number List box, then any other boxes within the Association box will be associated with all of the entities referred to by the Number List box.

M.5.3 Composition

Composition data is divided into fixed options, contained in the Composition Options box (M.11.10.1), and a sequence of instructions contained in one or more Instruction Set boxes (M.11.10.2) boxes. Each instruction comprises a set of render parameters. Each instruction set has an associated repeat count which allows for the efficient representation of long sequences of repeating instructions such as occur in full motion sequences or in slide shows which use a repeated frame transition animation. A JPX file reader shall display a JPX file by reading and executing the instructions in sequence order, from each instruction set in sequence order and repeated according to its repeat value. The file is considered fully rendered either when there are no more instructions to execute, or no compositing layer is present for the current instruction.

M.5.3.1 Composition rendering

The composition data defines the width and height of a render area into which the compositing layers are to be rendered. The size of the render area is the size of the rendered result and can be thought of as the overall image size. Parameters in each render instruction may specify:

- a rectangular region to crop from the source compositing layer;
- the location to place the top left corner of the (possibly cropped) compositing layer with respect to the top left corner of the render area;
- the width and height of the region within the render area into which the (possibly cropped) compositing layer is to be rendered.

For example, in a composite image using an RGBA colour space for all compositing layers, the current compositing layer (R_t, G_t, B_t, A_t) is ideally rendered over the background (R_b, G_b, B_b, A_b) to form the composed image (R_c, G_c, B_c, A_c) according to the following equations:

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

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

$$\begin{aligned} R_c &= A_t R_t + (1-A_t)R_b \\ G_c &= A_t G_t + (1-A_t)G_b \\ B_c &= A_t B_t + (1-A_t)B_b \end{aligned} \qquad \text{(M-2)}$$

However, the above equations require access to the background pixel, and for a variety of reasons, individual applications may not be willing or able to support such a rendering process. It is possible to emulate continuous alpha blending even in these cases by thresholding or dithering the provided alpha channel in order to generate a set of completely transparent or completely opaque pixels which can be rendered using simple pixel replacement over an unknown background. Specification of such methods is however outside the scope of this Recommendation | International Standard.

M.5.3.2 Animation model

In addition to the basic cropping and positioning parameters, each render instruction may include LIFE, PERSIST and NEXT-USE parameters. The LIFE parameter assigns a temporal duration to the instruction. This is the period of time that the reader should aim to place between the appearance of any screen update resulting from execution of the current instruction and any screen update resulting from the execution of the next instruction. PERSIST is a binary field indicating whether or not the compositing layer rendered by the current instruction should be treated as part of the background for the next instruction. If an instruction specifies false for PERSIST, then the reader must save the background prior to execution and use this saved background when executing the next instruction.

M.5.3.2.1 Special cases of life and persistence

There are a number of special combinations of LIFE and PERSIST parameters that require specific treatment by the reader.

- When PERSIST is false and LIFE is zero, no action should be performed by the reader. This combination might be used for example to force a reader to step over a thumbnail or print frame that would be displayed by a reader not capable of displaying the file as an animation.
- When PERSIST is true and LIFE is zero then this instruction should be executed together with the next instruction. In practice this combination may occur for a sequence of more than two instructions and shall place the reader into a frame composition mode. This mode is exited when an instruction with non-

zero PERSIST is encountered or when the end of the animation is reached. The set of instructions executed whilst in frame composition mode is referred to as a frame composition sequence. In frame composition mode, a virtual compositing layer is created (off-screen) by executing the instructions in the frame definition sequence. The PERSIST and LIFE parameters for the closing instruction of a frame definition are applied to the virtual compositing layer. This mode permits multi-sprite animation.

– When LIFE is the maximum value that can be stored the reader shall interpret this as a request for indefinite life. If the driving application has the capacity, it shall progress to the next instruction upon completion of some predetermined user interaction such as a mouse click. In addition to its use in animations, this feature may be used in files that store multi-page documents in order to force the reader to pause after composition of each page.

In general, screen updates shall not be performed after an instruction which has zero LIFE unless it is the last instruction in a frame composition sequence.

M.5.3.2.2 Assigning compositing layers to instructions and layer reuse

Compression of animated sequences is considerably improved if compositing layers can be reused in multiple frames. At the same time it is desirable that instructions only reference compositing layers that have already been decoded or are sequentially next in the file. Further, decoders can better optimize their caching of compositing layers if they can tell which layers are to be reused ahead of time. These policies are enforced in JPX via the discipline used to associate compositing layers with instructions.

The first instruction is always associated with the first compositing layer in the file. This instruction may specify a value for NEXT-USE. This value is interpreted as the number of instructions, including the current instruction, until the current compositing layer is reused. A value of zero indicates to the reader that the current compositing layer shall not be used again and may be forgotten. A value of one implies that the current compositing layer is to be used with the next instruction and so on. The reader must maintain a record of which instructions have been assigned compositing layers in this fashion. Whenever an instruction is encountered that does not have a compositing layer assigned to it, the next unused compositing layer defined in the file in sequence order shall be used; the use of NEXT-USE does not specify a loop. A single compositing layer may be reused any number of times in any given animation.

M.5.3.2.3 Looping animations

It is possible to specify that an animation should be looped. That is, when the animation has been fully rendered, the reader resets the display to its initial state and displays the animation over again. A loop count is optionally specified as part of the composition options. As with life, the maximum value for the loop parameter is used to indicate indefinite looping. Looping impacts the caching strategy used by the reader as many readers will not wish to free any compositing layer once decoded.

M.6 Using reader requirements masks to determine how a file can be used

The JPX format defines a file architecture rather than a specific, fixed set of data structures that will be found in a file. This architecture is complex enough to permit quite distinct file structures. For example, a JPX file may include:

– animation;
– image collections;
– redundant image sets (e.g., print and display versions of the same scene); or
– single images in special colour spaces (e.g., Parameterized CIELab).

As a result, the JPX brand tells a reader little about what capabilities it will require in order to correctly read an arbitrary JPX file. Instead, JPX conveys this information using three expressions contained with in the header of the file. These expressions describe:

1) the full set of technologies/features present in the file;
2) the set of technologies/features required by a decoder in order to read the file in a form consistent with the intent of the files creator;
3) a fallback mode which can be used to display a minimally acceptable result (usually a thumbnail or preview).

The fallback mode is communicated using the File Type box and is primarily intended to indicate whether or not the file can be read by a reader with specifically standardized capabilities (such as a conforming JP2 or Baseline JPX reader). However, the combination of technologies required may be too complex for a simple listing of the functionalities in the file. For this reason, the technology/feature set information takes the form of encoded logic expressions and are contained in a Reader Requirements box.

In general, a reader need only identify that it satisfies the requirements outlined in point 2 in order to be assured of being able to read enough of the file to fulfil the creator's intent. On the other hand, an editor may want to inform a user if there is any aspect of the file that it does not know how to support. Because all JPEG 2000 family files are not necessarily interoperable with all JPX readers, these expressions describe each aspect of the file, and the combination of features that must be supported to interpret the file correctly.

M.6.1 Types of expressions

M.6.1.1 Fully understand aspects

An encoded expression is used to describe all of the items contained within the file, and the combinations of functionality required to read these items. This expression describes each major option a reader has for processing the features of the file, regardless of whether support for a particular feature is required to make use of that aspect of the file. For example, a file may contain metadata describing an original file from which it was created; however, a reader is not required to understand this metadata in order to correctly use the file.

M.6.1.2 Display contents

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

M.6.1.3 Fallback

In the event that a file cannot be displayed as desired by the writer, a fallback method for displaying the file is defined. The fallback methods are intended to be ultimately interoperable, as such, they may not generate the exact desired output.

A list of fallback methods is stored in the File Type box at the beginning of the JPEG 2000 family file: all files which start with a JPEG 2000 Signature box must contain a File Type box. The File Type box contains a list of known methods of reading the file. For example, the JP2 file format defines the JP2 fallback position. This specifies that a reader conforming to the JP2 file format specification, as defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, I.2.6, can read the file.

This Recommendation | International Standard defines one additional fallback position, JPX baseline, as defined in M.9. Also, other organizations may register other fallback positions by following the process defined in M.7. These define other minimal readers, and a set of file formats that are guaranteed to be readable by those readers.

Fallback methods are stored in the File Type box. The order within the box is of no significance. When a reader supports more than one of the fallback methods described in the file, it is up to the reader to determine which fallback methods to use.

M.6.2 Expression representation

The expressions of the requirements to fully understand all aspects, and to display the file as desired are stored in the Reader Requirements box, which is a mandatory feature of a JPX file format. If the Reader Requirements box is not present, the File Type box describes the full functionality of the file.

The Reader Requirements expressions are logical expressions involving functions which can be provided by the writer. These expressions may include vendor-specific options. The expression is factored into `AND`-separated sub-expressions, each containing only `OR` operations. These are then encoded into bitmasks which the reader can use to determine how to handle the file.

The Reader Requirements box includes two expressions, the Fully Understand Aspects expression, and the Display Contents expression. In general, these expressions will share several sub-expressions; shared sub-expressions are only stored once, and bitmasks are used to determine which sub-expressions belong to each expression.

M.6.2.1 Formulating requirements expressions

When formulating the requirements expressions describing a file, each expression is firstly factored into AND-separated sub-expressions, each sub-expression containing OR-separated options. Thus, an expression in the form:

$$(A \,\&\, B \,\&\, C \,\&\, E) \,|\, (D \,\&\, E) \qquad \text{(M-3)}$$

is factored into the expression:

$$(A \mid D) \,\&\, (B \mid D) \,\&\, (C \mid D) \,\&\, E \qquad \text{(M-4)}$$

Each sub-expression is expressed as a bit-array, with a flag set for each option appearing in that sub-expression. So, for example, the expression (A | D) becomes:

Table M.1 – Example expression

A	B	C	D	E
1	0	0	1	0

The full expression is written in table form:

Table M.2 – Expanded expression

A | D	B | D	C | D	E	
1	0	0	0	A
0	1	0	0	B
0	0	1	0	C
1	1	1	0	D
0	0	0	1	E

with each sub-expression in a column of the table. Thus, to satisfy the requirements of this expression, a reader has to support one of the functionalities in each column of the table.

However, there are two expressions to be encoded, and they will, in general, share common factors (because the functionality required to display a file is part of the functionality required to fully understand its contents). Thus, the two expressions are combined into one table, and a bitmask is provided to determine which columns in the table belong to each expression. So, if the expression in Equation M-3 is Display Contents and if the expression for Fully Understand Aspects is:

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

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

Table M.3 – Example factored expression

1	0	0	0	0	A
0	1	0	0	0	B
0	0	1	0	0	C
1	1	1	0	0	D
0	0	0	1	0	E
0	0	0	0	1	F
0	0	0	0	1	G
1	1	1	1	0	Display contents
1	1	1	1	1	Fully understand

where the first four columns are the Display Contents requirement, and all five sub-expressions are required to Fully Understand Aspects. Thus the bitmask for Display Contents is 11110, and the bitmask for Fully Understand Aspects is 11111.

This table can be read in columns, as a set of sub-expressions defining the functionality required of a reader, or in rows, as a set of compatibility bitmasks which a reader can use to determine whether it can read the file. By obtaining the

bitwise OR of the rows which correspond to the functionalities present, and comparing the result with the bitmasks for the two expressions, a reader can determine whether it can satisfy the requirements of each.

Thus, a writer can construct the table in columns, setting flags corresponding to the options in each sub-expression, and generating the bitmasks describing which sub-expressions are ANDed together to form the full expressions for Fully Understand Aspects, and Display Contents. It can then obtain the compatibility bitmasks for each function which a reader may use in reading the file, by extracting the row corresponding to each functionality present.

M.6.2.2 Encoding requirements expressions

The requirements expressions are encoded in the Reader Requirements box, starting with a mask length, indicating the width of the compatibility bitmasks, to byte precision. This is followed by the bitmasks for the Fully Understand Aspects and Display Contents expressions, and in turn by a list of the features used and their compatibility masks, obtained from the rows of the expression table.

The list includes a set of standard features used (as specified in Table M.14) and their compatibility bitmasks, followed by a list of vendor-specific features (represented as UUIDs), together with the compatibility bitmasks associated with these. Apart from the separation into standard and vendor-specific features, the order of presentation is unimportant. This structure is fully specified in M.11.1.

M.6.2.3 Examples

For example, consider an image processing program that produces a JPX file containing a single image in the sRGB colour space, and a multiple codestream version containing the compositing layers, to allow an editor to work with the image, and includes metadata containing the history of the file, then the requirement to display the file is:

$$\text{sRGB} \,\&\, (\text{single codestream} \,|\, (\text{multiple codestream} \,\&\, \text{compositing})) \tag{M-6}$$

and to fully understand the file requires:

$$\text{sRGB} \,\&\, (\text{single codestream} \,|\, (\text{multiple codestream} \,\&\, \text{compositing})) \,\&\, \text{metadata} \tag{M-7}$$

Equation M-6 factors as:

$$\text{sRGB} \,\&\, (\text{single codestream} \,|\, \text{multiple codestream}) \,\&\, (\text{single codestream} \,|\, \text{compositing}) \tag{M-8}$$

Equation M-7 factors similarly, so the sub-expressions are:

a) sRGB;

b) single codestream | multiple codestream;

c) single codestream | compositing;

d) metadata.

The resulting expression table and bitmasks is shown in Table M.4.

The bitmasks indicate which sub-expressions are required for each degree of functionality. Thus the expression for Display Contents is:

$$(\text{Sub-expr a}) \,\&\, (\text{Sub-expr b}) \,\&\, (\text{Sub-expr d}) \tag{M-9}$$

Table M.4 – Example of a Reader Requirements expressions for Equations M-6 and M-7

	Sub-Exp a	**Sub-Exp b**	**Sub-Exp c**	**Sub-Exp d**
sRGB	1	0	0	0
single codestream	0	1	1	0
multiple codestream	0	1	0	0
compositing	0	0	1	0
metadata	0	0	0	1
Fully Understand Aspects bitmask	1	1	1	1
Display Contents bitmask	1	1	1	0

Thus, the above table is stored in the file as shown in Table M.5:

Table M.5 – Example of a Reader Requirements box for Equations M-6 and M-7

Mask Length (in bytes)	1[a)]				
Fully Understand Aspects bitmask	1	1	1	1	
Display Contents bitmask	1	1	1	0	
Number of Standard Features	5				
Standard Feature Compatibility list	sRGB				
	1	0	0	0	
	single codestream				
	0	1	1	0	
	multiple codestream				
	0	1	0	0	
	compositing				
	0	0	1	0	
	metadata				
	0	0	0	1	
Number of Vendor-Specific features	0				

a) 1 byte, because masks are 4 bits wide, which fits into 1 byte.

As a second example, suppose the ACME printer driver produces a JPX file which contains a single codestream sRGB image, for display, and a CMYK image which can be read by a printer driver using ACME's vendor-specific functions. For this file, the expression to Fully Display Contents is:

$$(\text{sRGB} \,\&\, \text{single codestream}) \,|\, (\text{CMYK} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions}) \qquad \text{(M-10)}$$

while the expression to Understand All Aspects is:

$$((\text{sRGB} \,\&\, \text{single codestream}) \,|\, (\text{CMYK} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions})) \,\&\, \text{metadata} \,\&\, \text{ACME print metadata} \qquad \text{(M-11)}$$

Factorizing these into sub-expressions gives:

$$\text{single codestream} \,\&\, (\text{sRGB} \,|\, \text{CMYK}) \,\&\, (\text{sRGB} \,|\, \text{ACME extensions}) \tag{M-12}$$

and:

$$\text{single codestream} \,\&\, (\text{sRGB} \,|\, \text{CMYK}) \,\&\, (\text{sRGB} \,|\, \text{ACME extensions}) \,\&\, \text{metadata} \,\&\, \text{ACME print metadata} \tag{M-13}$$

respectively.

The resulting file Reader Requirements table is shown in Table M.6:

Table M.6 – Reader Requirements table for Equations M-10 and M-11

sRGB	0	1	0	1	0
CMYK	0	1	0	0	0
single codestream	1	0	0	0	0
metadata	0	0	1	0	0
ACME extensions	0	0	0	1	0
ACME print metadata	0	0	0	0	1
Fully Understand Aspects bitmask	1	1	1	1	1
Display Contents bitmask	1	1	0	1	0

As always, each column represents a factor sub-expression, and each row provides a compatibility bitmask which a reader can use to determine whether it can read the file. This example includes vendor-specific features, and that sub-expressions can involve both standard and vendor-specific functionality.

These are stored in the file as shown in Table M.7:

Table M.7 – Reader Requirements box data for Equations M-10 and M-11

Mask Length	1					
Fully Understand Aspects bitmask	1	1	1	1	1	
Display Contents bitmask	1	1	0	1	0	
Number of Standard Features	4					
Standard Feature Compatibility list	sRGB					
	0	1	0	1	0	
	CMYK					
	0	1	0	0	0	
	single codestream					
	1	0	0	0	0	
	metadata					
	0	0	1	0	0	
Number of Vendor Features	2					
Vendor Feature Compatibility list	ACME extensions UUID					
	0	0	0	1	0	
	ACME print metadata UUID					
	0	0	0	0	1	

Also consider a JPX file that contains two compositing layers that are not combined by either animation or compositing; they are conceptually two separate rendered results. The first compositing layer contains a single codestream in the sRGB colourspace (specified using the Enumerated method). The second compositing layer contains a single codestream, for which the colourspace is specified using the Any ICC method. In addition, the second compositing layer contains vendor-specific metadata.

For this file, the expression to Fully Display Contents is:

$$(\text{sRGB} \,\&\, \text{single codestream}) \,|\, (\text{full ICC} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions}) \qquad \text{(M-14)}$$

while the expression to Understand All Aspects is:

$$((\text{sRGB} \,\&\, \text{single codestream}) \,|\, (\text{full ICC} \,\&\, \text{sRGB} \,\&\, \text{single codestream} \,\&\, \text{ACME extensions})) \qquad \text{(M-15)}$$

Factorizing these into sub-expressions gives:

$$\text{single codestream AND (sRGB} \,|\, \text{full ICC) AND (sRGB} \,|\, \text{ACME extensions)} \qquad \text{(M-16)}$$

and:

$$\text{single codestream AND sRGB AND full ICC AND ACME extensions} \qquad \text{(M-17)}$$

respectively. The resulting file Reader Requirements table is shown in Table M.8:

Table M.8 – Reader Requirements box data for Equations M-16 and M-17

Mask Length	1						
Fully Understand Aspects bitmask	1	0	0	1	1	1	
Display Contents bitmask	1	1	1	0	0	0	
Number of Standard Features	3						
Standard Feature Compatibility list	sRGB						
	0	1	1	1	0	0	
	Any ICC						
	0	1	0	0	1	0	
	single codestream						
	1	0	0	0	0	0	
Number of Vendor Features	1						
Vendor Feature Compatibility list	ACME extensions UUID						
	0	0	0	1	0		

M.6.3 Testing an Implementation against requirements expressions

In order to determine whether it can read the file, the reader extracts the compatibility bitmask from the feature list entry corresponding to each functionality which it provides. If a flag is set in the bitmask, then this function is an option in the sub-expression corresponding to the flag.

Thus, if the reader performs a bitwise OR of the bitmasks for all of the functions which it provides, it can determine whether it can read the file by comparing the result with the Fully Understand Aspects and Display Contents bitmasks from the file. Also, by reconstructing the expression table and looking up the column (or columns) of the table where the file bitmask flag is set, and the reader's compatibility bitmask flag is not, the reader can determine which extra functionality is required to read the file.

If there is functionality provided by the reader, which is not in the feature list for the file, then the feature is not required to read the file (and the bitmask may be assumed to be all zeroes).

Consider the first example Reader Requirements box:

Table M.9 – Example Reader Requirements box to test

Mask Length (in bytes)	1				
Fully Understand Aspects bitmask	1	1	1	1	
Display Contents bitmask	1	1	1	0	
Number of Standard Features	5				
Standard Feature Compatibility list	sRGB				
	1	0	0	0	
	single codestream				
	0	1	1	0	
	multiple codestream				
	0	1	0	0	
	compositing				
	0	0	1	0	
	metadata				
	0	0	0	1	
Number of Vendor-Specific features	0				

In this example, if the reader supports the sRGB and single codestream functions, it looks up the bitmasks for these features (`1000` and `0110`, respectively). The bitwise OR gives the compatibility mask of this file for the reader, `1110`. Thus this reader can fully display the contents of the file; however, it will not understand all aspects of the file.

Noting that the compatibility mask for the reader (DCM) is 1110, and the Fully Understand Aspects Mask (FUAM) mask is 1111, the reader can perform (FUAM & !DCM) bitwise, to get 0001. This tells it that the missing functionality's bitmask has bit 4 set, so it can search the list for this, and determine that the missing functionality is metadata support.

M.7 Extensions to the JPX file format and the registration of extensions

Registration is the process of adding extensions to the capabilities to this Specification after the Specification has been published. In this Recommendation | International Standard, many capabilities may be extended through registration. Other items may be extended, but do not require the intervention of a third party to prevent extension conflict. This clause identifies those items which may be extended by registration and the process by which capabilities may be registered, as well as identifying those items which may be extended independent of registration and the process by which the Registration Authority will publish those extensions.

M.7.1 Registration elements

The registration process is composed of the following elements.

Table M.10 – Registration elements

Element	Identification
Registration Authority	WG1
Submitter	Entity creating the extension to this Recommendation | International Standard
Review Board	WG1 file format committee
Submission/Item	The proposed extension
Review Board chair	File Format editor
Test	Varies by item

Registration Authority: The organizational entity responsible for reviewing, maintaining, distributing, and acting as a point of contact for all activities related to the registration.

Submitter: The submitter is the organization or person who requests that the item be registered.

Review board: The review board is the organizational entity that approves the registration of a proposed item. It is composed of an ad hoc committee appointed by the Review Board Chair.

Review board chair: The review board chair is responsible for seeing that each candidate item is considered. He communicates with the submitter through the Registration Authority.

Test: Rationale that the Review Board should use to determine if submission/item should be registered.

Submission/Item: This is the proposal for registration. Each proposal shall include the name of the item to be extended, the proposed tag/identity for the extension, and a rationale/purpose for the extension.

M.7.2 Differentiation between publication and registration

In the JPX file format, several features of the file format may be extended independent of a registration process. For example, the format provides the Vendor Colour method to allow individual vendors to indicate custom colourspaces through a form of enumeration (using UUID's) without involving a third-party.

However, to promote interoperability, it is useful to gather the definitions indicated by these UUID's in one place. In this case, "registration" of the UUID is not needed to eliminate possible conflict with other vendors, and does not help a developer when considering which features should be implemented in a particular product.

As such, this proposal clearly differentiates between solutions that require the intervention of a registration committee from those solutions that can be created solely by the individual vendor. This proposal also allows the Registration Authority to label particular proposed element definitions as preferred solutions.

M.7.2.1 Published

Published items are those elements of a JPX file that can be safely extended, generally through the use of URL's or UUID's, without risk of conflict with other vendors. Values can be assigned for published items without the help of a third-party. However, it is useful to involve a third-party as a single "publisher" of the definitions of the extended elements from all vendors.

For example, a Vendor Colourspace value is a published item; the value is indicated through the use of a UUID. To promote interoperability, the Registration Authority shall publish a database of all known vendor colourspaces and the colourimetric definitions associated with each UUID.

M.7.2.2 Registered

Registered items are those elements of a JPX file that are restricted to a limited (albeit large in some cases) number of values. For these items, there exists the possibility that two vendors would use the same value for different meanings if there is not a third-party mediating the use of the element values. Also, in most cases, there are additional criteria for the allocation of values to registered items. Because the number of available values for most registered items is limited, and given that most problems can be solved using publishable items rather than registered items, the allocation of a registered value shall be considered as the specification of a preferred solution.

For example, an Enumerated Colourspace is a registered item; the value is indicated through the use of a 4-byte integer. The Review Board shall evaluate all proposed enumerated colourspaces in terms of preferred technologies. Proposed solutions that are considered preferred solutions shall be allocated a value by the Registration Authority. Proposers of solutions that are not preferred shall be referred to the Vendor Colourspace method as an alternate solution to the proposed problem.

M.7.2.3 Preferred published solutions

In some cases, such as the use of the UUID or XML boxes to embed metadata within a JPX file, there is not a corresponding registered item which can be used for preferred solutions. As such, the Registration Authority, upon recommendation from the Review Board, may choose to label a particular value of a published item as a preferred solution.

M.7.3 Items which can be extended by registration

The following items may be extended by registration. Only items that are listed here may be extended by registration.

Table M.11 – Items which can be extended by registration

Item	Purpose
Enumerated colourspaces	Define additional standard colourspaces
Desired reproduction boxes	Define additional reproduction scenarios and the data required to transform images for output in those scenarios
Compatibility modes	Define additional compatibility modes to promote interoperability in markets not explicitly addressed by the JPX baseline feature set
Standard feature list	Define additional standard feature codes for the Reader Requirements box

M.7.3.1 Enumerated colourspace

New values of the EnumCS field in the Colour Specification Box shall be registrable. A proposal to register a new enumerated colourspace must contain a complete colourimetric definition of that colourspace, instructions on how to use images in that colourspace, any required enumerated parameters (for the EP field in the Colour Specification box) and any default values of those parameters.

However, when evaluating proposed enumerated colourspaces, the Review Board shall limit the allocation of enumerated values to international and defacto standards, in addition to determining appropriateness of the proposed solution. Non-standard colourspaces shall be specified through the use of the Vendor Colourspace method.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

M.7.3.2 Desired reproduction boxes

New box types for Desired Reproduction information (like the Graphics Technology Standard Output box in the JPX format) shall be registrable. A proposal to register a new desired reproduction must contain a complete definition of the reproduction scenario, including the binary structure of the reproduction data as well as when an application should use the reproduction data.

The Review Board shall evaluate proposed reproductions based on the following criteria:

- Does it meet a need not already met by other defined reproductions?
- Is the binary format of the reproduction data sufficiently defined?
- Is it a general case or a vendor-specific case (i.e., output on a typical CRT vs output on a particular CRT model from a particular vendor)?

The Review Board shall restrict the allocation of Desired Reproduction boxes to general cases that meet needs not already met by other defined reproductions. Other proposed reproductions shall be specified by embedding the data in a UUID box and placing that UUID box within the Desired Reproduction superbox.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

M.7.3.3 Compatibility modes

New compatibility modes for the File Type box (values for the CLi fields) shall be registrable. A proposal to register a new compatibility mode must contain a complete definition of the JPX reader requirements for that compatibility mode, as well as the definition of the 4-byte CLi field for this mode.

The Review Board shall evaluate proposed compatibility modes based on the following criteria:

- Does it meet a need not already met by other compatibility modes?
- Is it expected that a wide range of applications will desire to implement support for the particular set of features required by this compatibility mode, or is this mode specific to a particular vendor or application?
- Are readers that support this compatibility mode required to support the entire JPX baseline feature set?
- Will the creation of this compatibility mode negatively affect interoperability in the target application area?

The Review Board shall restrict the allocation of compatibility modes to cases that meet the needs of a wide range of applications, that are not already met by other modes, and that do not negatively affect interoperability in the target application area. The allocation of modes to feature sets that do not require support for the baseline feature set will be denied in cases where the baseline features are appropriate for the target application.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

M.7.3.4 Standard feature codes

New values of the SFi field in the Reader Requirements box shall be registrable. A proposal to register a new standard feature code colourspace must be made along with the proposal to register that feature. As such, new standard feature codes shall only be allocated for new registered features.

M.7.4 Published items

The following items may be extended without the intervention of the Registration Authority, and the Registration Authority shall publish the specifications of those extensions. Only items that are listed here will be published. The text of extension to be published shall be evaluated by the Review Board before publication by the Registration Authority. In addition, the Review Board may choose to label particular published solutions as preferred, as described in Table M.12.

Table M.12 – Items which can be extended by registration

Item	Purpose
Vendor feature codes	Define additional vendor-specific features
Vendor colourspaces	Define additional vendor-specific colourspaces
Binary filter algorithms	Define additional algorithms for use in the Binary Filter box
UUID metadata	Define additional metadata for use within UUID boxes
XML metadata	Define additional metadata for use within XML boxes

M.7.4.1 Vendor feature codes

The Review Board shall publish the definition of submitted vendor feature codes (values of the VF^i field in the Application Profile box). All submissions must include a complete definition of the feature, including defined data structure, interactions with other data structures, and instructions on how to implement a decoder that supports that feature.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

M.7.4.2 Vendor colourspaces

The Review Board shall publish the definition of submitted vendor colourspace codes (values of the VCLR field in the METHDAT field for Colour Specification boxes that use the Vendor Colour method). All submissions must include a complete colourimetric definition of that colourspace, instructions on how to use images in that colourspace, any required vendor parameters (for the VP field in the Colour Specification box) and any default values of those parameters.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Review Board may choose to label a particular vendor colourspace as a preferred solution. The committee shall make this decision using the same criteria as would be used when evaluating a proposal for allocation of an enumerated colourspace value (as specified in M.7.3.1).

M.7.4.3 Binary filter algorithms

The Review Board shall publish the definition of submitted binary filter type values (values of the F field in the Binary Filter box). All submissions must include a complete definition of the algorithm and the format of the DATA field in the binary filter box.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Board may choose to label a particular binary filter as a preferred solution. The Board shall reserve this label for international or defacto standards, based on the desired use of the binary filter. For example, encryption technology can be used for both encrypting data and for creating digital signatures. While a particular binary filter may be a preferred solution for encrypting metadata, it may not be preferred for digital signatures.

M.7.4.4 UUID metadata

The Review Board shall publish the definition of submitted UUID's used in UUID boxes. All submissions must include a complete definition of the DATA field in the UUID box and instructions on using that data.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the Board may choose to label a particular metadata specification as a preferred solution. The committee shall reserve this label for international or defacto standards, based on the target application for the metadata.

M.7.4.5 XML metadata

The Review Board shall publish the definition of submitted Document Type Definitions (DTD's) and XML Schema's used in XML boxes. All submissions must include a complete definition of the information contained in XML instance documents (found in XML boxes) that use that DTD or schema, as well as instructions on using that data.

The Review Board shall evaluate all submissions. If the text of the submission does not meet the requirements, then it shall be returned to the submitter for clarification.

In addition, the committee may choose to label a particular metadata specification as a preferred solution. The committee shall reserve this label for international or defacto standards, based on the target application for the metadata.

M.7.5 Registration process

The following is the registration process:

1) A submitter creates a candidate item for registration.
2) The candidate item is submitted to the Registration Authority.
3) The Registration Authority passes the candidate item to the Review Board Chair.
4) The Review Board Chair distributes the candidate item to the Review Board and schedules meetings, phone calls, etc. as appropriate for consideration of the item.

5a) If approved, the Chair passes the approval to the Registration Authority who notifies ISO and the submitter, and makes the registered or published item available.

5b) If declined, the Chair prepares a response document indicating why the item was declined and passes this to the Registration Authority who notifies the submitter.

M.7.6 Timeframes for the registration process

M.7.6.1 Requests for registration

The Review Board shall respond to all requests for registration within five months from the date of submission. Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 to evaluate the proposal, make a decision, and draft the response.

M.7.6.2 Requests for publication

The Review Board shall respond to all request for publication within two months from the date of submission. Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 or use e-mail discussions or conference calls to evaluate the proposal, make a decision, and draft the response.

M.7.6.3 Requests for preferred status for published solutions

The Review Board shall respond to all requests for preferred status for published solutions within five months from the date of submission. A request for preferred status may be made at the same time that the request for publication is made. Within that time period, the Review Board will meet at an official meeting of ISO/IEC JTC1/SC29/WG1 to evaluate the proposal, make a decision, and draft the response.

M.8 Differences from the JP2 binary definition

The box structure of a JPX file is identical to that of a JP2 file. A JPX file is a sequence of boxes, defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, I.6. However, many new boxes are defined, and the structures of several boxes are extended as follows:

– The BR field in the File Type box shall be "jpx\040" for files that are completely defined by this Recommendation | International Standard. In addition, a file that conforms to this Recommendation | International Standard shall have at least one CLi field in the File Type box, and shall contain the value 'jpx\040' in one of the CLi fields in the File Type box.

- Additional forms of the Colour Specification box are defined (M.11.7.2).
- The JPEG 2000 compressed codestream may contain extensions as defined in Annex A.
- Under some circumstances, the JP2 header box may be found in anywhere in the file, provided that it is not encapsulated within another box (it shall always be at the top level of the file). See M.11.5 for a description of the storage of the JP2 header box within a JPX file.
- Additional box types are defined within the scope of this Recommendation | International Standard.

M.9 Conformance

M.9.1 Interpretation of JPX data structures

All conforming files shall contain all boxes required by this Recommendation | International Standard as shown in Table M.13, and those boxes shall be as defined in this Recommendation | International Standard.

A JPX reader that supports a particular subset of JPX features is a conforming JPX reader if that reader properly supports all files that contain a Display Contents mask (in the Reader Requirements box) or a Fallback position (in the File Type box) indicating that the file can be read using only that particular subset of features; a conforming reader may fall back from any extended feature, as allowed by the Reader Requirements or File Type box, provided that the reader does not claim a higher level of conformance than it actually supports.

M.9.2 Support for JPX feature set

In general, a JPX reader is not required to support the entire set of features defined within this Recommendation | International Standard. However, to promote interoperability, the following baseline set of features is defined. Files that are written in such a way as to allow a reader that supports only this JPX baseline set of features to properly open the file shall contain a CLi field in the File Type box with the value 'jpxb' (0x6a70 7862); all JPX baseline readers are required to properly support all files with this code in the compatibility list in the File Type box. The definition of a JPX baseline file is as follows:

M.9.2.1 Compression types

Support for compression types other than JPEG 2000 (the C field in the Image Header box = 7) shall not be required to properly display the file.

M.9.2.2 Compositing layers

Support for multiple compositing layers is not required to properly display the file. However, the file may contain multiple compositing layers. If the file does contain compositing layers, the first compositing layer in the file (signalled by the first Compositing Layer Header box) shall be rendered. That compositing layer shall consist of one and only one codestream, which shall represent the rendered result as rendered into a single codestream. In addition, the codestream that shall be processed by a reader that only supports the JPX feature set shall be the first codestream in the file.

M.9.2.3 Codestreams

The codestream specified by the first compositing layer shall be compressed using the JPEG 2000 compression algorithm, as defined by ITU-T Rec. T.800 | ISO/IEC 15444-1 and shall not require support for extensions other than the irreversible decorrelation transformation (specified in J.3.1.1.1) and non-linearity transformation (specified in Annex K) extensions.

A conforming JPX baseline reader is not required to support other portions of the multiple component transformation extension. If support for the irreversible decorrelation transformation is required, then the first codestream shall be restricted as follows:

- The value of the Qmcc field in any MCC marker segment shall be 1.
- The $Xmcc^i$ field in any MCC marker segment shall indicate an array based decorrelation transformation.
- The $Tmcc^i$ field in any MCC marker segment shall indicate an irreversible transformation
- The Nmco field in any MCO marker segment shall be 1.

That codestream may contain other extensions provided that support for those extensions is not required to decode the codestream.

Other codestreams in the file may require support for other extensions in order to be decoded.

M.9.2.4 Colour specification

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

- Enumerated method EnumCS values indicating either sRGB, sRGB-grey, ROMM-RGB, sYCC, e-sRGB, or e-sYCC.
- Enumerated method EnumCS value of CIELab using default values (EP fields are not specified).
- Enumerated method EnumCS value of CIELab using enumerated parameters (as specified in the EP fields in the Colour Specification box).
- Enumerated method EnumCS value of CIEJab using default values (EP fields are not specified).
- Enumerated method EnumCS value of CIEJab using enumerated parameters (as specified in the EP fields in the Colour Specification box).
- Restricted ICC method.
- Any ICC method.

A baseline JPX file may contain additional colourspace specifications, such as other enumerated values or vendor defined colourspace specifications. However, the file shall contain at least one colour specification method from the list above.

In addition, at least one Colour Specification box specified for the first compositing layer shall have an APPROX value of 3 or less (indicating a "reasonable" or better approximation of the true colourspace of the image).

M.9.2.5 Codestream fragmentation

The codestream used by the first compositing layer in a baseline JPX file may be fragmented. However, all fragments shall be in the JPX file itself and shall be found in the file in the order they are listed in the Fragment Table box, starting the search at byte 0 of the file and proceeding sequentially to the end of the file.

M.9.2.6 Cross-reference boxes

All Cross-Reference boxes that must be parsed in order to properly interpret or decode the first compositing layer in the file shall only point to fragments that are contained within the JPX file itself. Those fragments shall be in the same order in the file as they are listed in the Fragment List box, starting the search at byte 0 of the file and proceeding sequentially to the end of the file. In addition, all fragments shall be found in the file before the data representing the codestream used by the compositing layer. If that codestream is specified by a Contiguous Codestream box, then all fragments for the cross-reference shall be found before that Contiguous Codestream box. If the codestream is specified by a Fragment Table box, then all fragments for the cross-reference shall be found before the Media Data box containing the first fragment from the codestream.

M.9.2.7 JP2 Header box location

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

M.9.2.8 Opacity

A baseline JPX reader shall properly interpret opacity channels, through either direct mapping to a codestream component using either the Channel Definition box or the Opacity box, or by expansion from a palette. The use of opacity outside of the use of compositing layers within the JPX file indicates that the decoded image data shall be composited onto an application defined background.

M.9.2.9 Other data in the file

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

M.10 Key to graphical descriptions (informative)

Each box is described in terms of its function usage and length. The function describes the information contained in the box. The usage describes the logical location and frequency of this box in the file. The length describes which parameters determine the length of the box.

These descriptions are followed by a figure that shows the order and relationship of the parameters in the box. Figure M.4 shows an example of this type of figure. A rectangle is used to indicate the parameters in the box. The width of the rectangle is proportional to the number of bytes in the parameter. A shaded rectangle (diagonal stripes) indicates that the parameter is of varying size. Two parameters with superscripts and a gray area between indicate a run of several of these parameters. A sequence of two groups of multiple parameters with superscripts separated by a gray area indicates a run of that group of parameters (one set of each parameter in the group, followed by the next set of each parameter in the group). Optional parameters or boxes will be shown with a dashed rectangle.

The figure is followed by a list that describes the meaning of each parameter in the box. If parameters are repeated, the length and nature of the run of parameters is defined. As an example, in Figure M.4, parameters C, D, E and F are 8, 16, 32 bit and variable length respectively. The notation G^0 and G^{N-1} implies that there are *n* different parameters, G^i, in a row. The group of parameters H^0 and H^{M-1}, and J^0 and J^{M-1} specify that the box will contain H^0, followed by J^0, followed by H^1 and J^1, continuing to H^{M-1} and J^{M-1} (M instances of each parameter in total). Also, the field E is optional and may not be found in this box.

After the list is a table that either describes the allowed parameter values or provides references to other tables that describe these values.

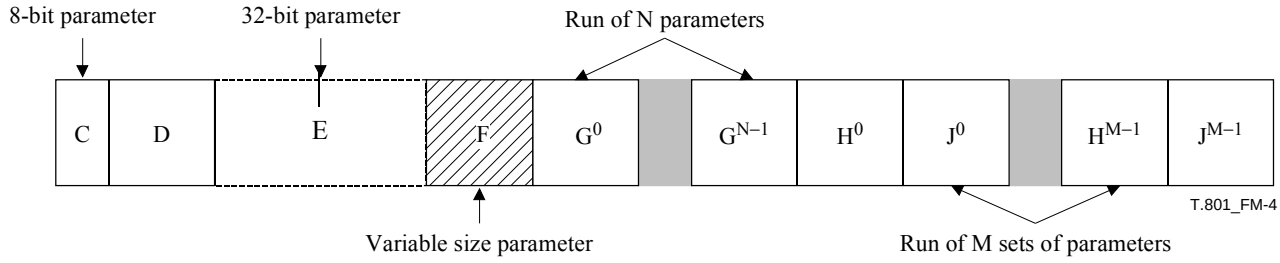

Figure M.4 – Example of the box description figures

In addition, in a figure describing the contents of a superbox, an ellipsis (…) will be used to indicate that contents of the file between two boxes is not specifically defined. Any box (or sequence of boxes), unless otherwise specified by the definition of that box, may be found in place of the ellipsis.

Figure M.5 – Example of the superbox description figures

For example, the superbox shown in Figure M.5 must contain an AA box and a BB box, and the BB box must follow the AA box. However, there may be other boxes found between boxes AA and BB. Dealing with unknown boxes is discussed in M.12.

M.11 Defined boxes

The following boxes are defined as part of JPX file format. In addition, any box defined as part of the JP2 file format that is not also listed here is also defined for use in a JPX file. However, this Recommendation | International Standard may redefine the binary structure of some boxes defined as part of the JP2 file format. For those boxes, the definition found in this Recommendation | International Standard shall be used for all JPX files.

Figure M.6 shows the hierarchical organization of the boxes in a JPX file. Several of these boxes are defined within the JP2 file format specification. This illustration does not specify nor imply a specific order to these boxes. In many cases, the file will contain several boxes of a particular box type. The meaning of each of those boxes is dependent on the placement and order of that particular box within the file.

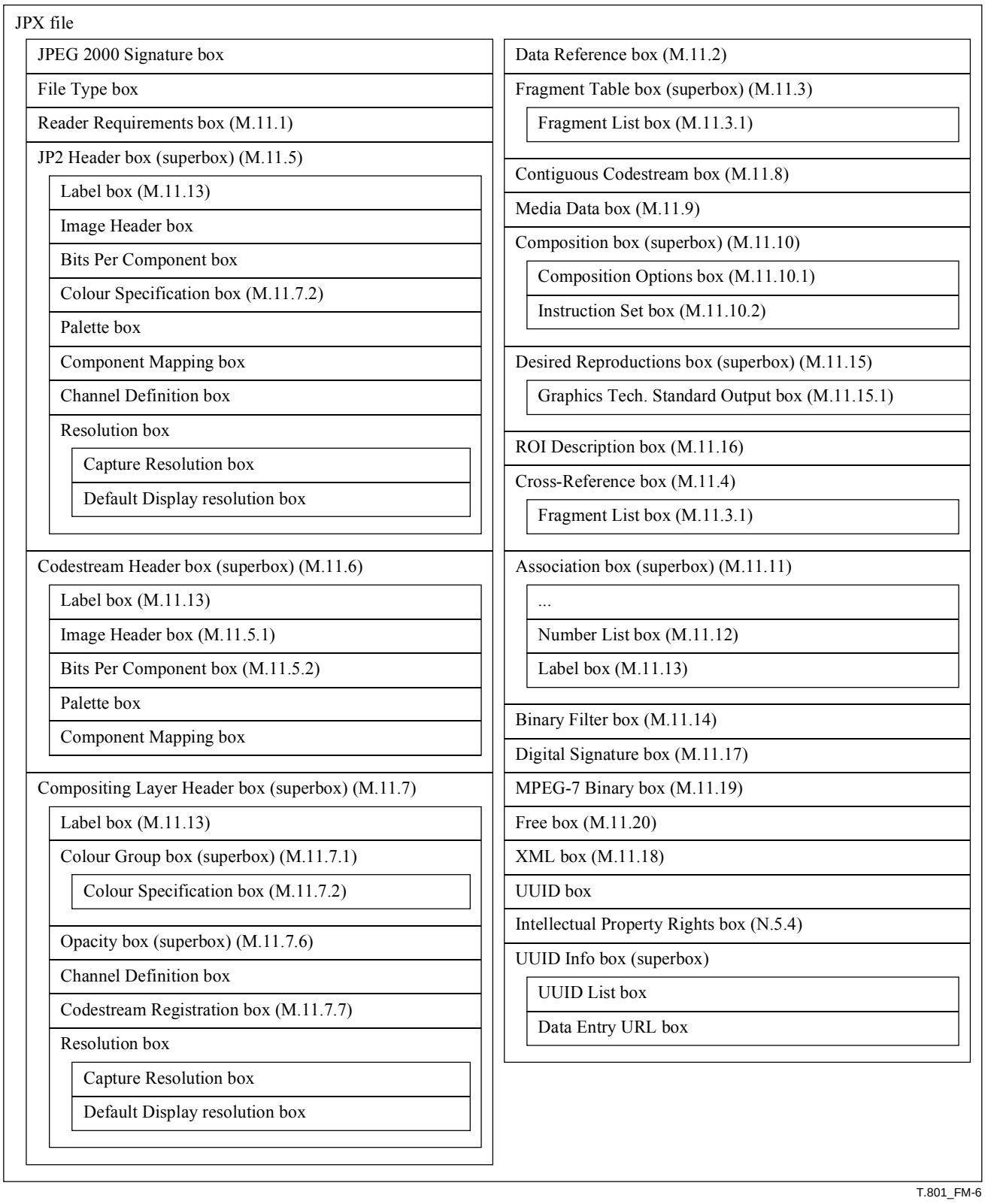

Figure M.6 – Boxes defined within a JPX file

Table M.13 lists all boxes defined as part of this Recommendation | International Standard. Boxes defined as part of the JP2 file format are not listed. A box that is listed in Table M.13 as "Required" shall exist within all conforming JPX files. For the placement of and restrictions on each box, see the relevant subclause defining that box.

Table M.13 – Boxes defined within this Recommendation | International Standard

Box name	Type	Required ?	Comments
Reader Requirements box (M.11.1)	'rreq' (0x7272 6571)	Yes	This box specifies the different modes in which this file may be processed.
JP2 Header box (superbox) (M.11.5)	'jp2h' (0x6A70 3268)	No	This box specifies JP2 compatibility and default header information for the codestreams and compositing layers.
Image Header box (M.11.5.1)	'ihdr' (0x6968 6472)	Yes	This box specifies the size of the image and other related fields.
Bits Per Component box (M.11.5.2)	'bpcc' (0x6270 6363)	No	This box specifies the bit depth of the components in the file in cases where the bit depth is not constant across all components.
Codestream Header box (superbox) (M.11.6)	'jpch' (0x6A70 6368)	No	This box specifies general information, such as bit depth, height and width about one specific codestream in the file.
Compositing Layer Header box (superbox) (M.11.7)	'jplh' (0x6A70 6C68)	No	This box specifies general information, such as colourspace and resolution, about one specific compositing layer in the file.
Colour Group box (superbox) (M.11.7.1)	'cgrp' (0x6367 7270)	No	This box groups a sequence of Colour Specification boxes that specify the different ways that the colourspace of a layer can be processed.
Colour Specification box (M.11.7.2)	'colr' (0x636F 6C72)	Yes	This box specifies one way in which the colourspace of an image can be processed. The definition of this box is extended from the definition in the JP2 file format.
Opacity box (M.11.7.6)	'opct' (0x6F70 6374)	No	This box specifies how opacity information is contained within a set of channels.
Codestream Registration box (M.11.7.7)	'creg' (0x6372 6567)	No	This box specifies the alignment between the set of codestreams that make up one compositing layer.
Data Reference box (M.11.2)	'dtbl' (0x6474 626C)	No	This box contains a set of pointers to other files or data streams not contained within the JPX file itself.
Fragment Table box (superbox) (M.11.3)	'ftbl' (0x6674 626C)	No	This box specifies how one particular codestream has been fragmented and stored within this JPX file or in other streams.
Fragment List box (M.11.3.1)	'flst' (0x666C 7374)	No	This box specifies a list of fragments that make up one particular codestream within this JPX file.
Cross-Reference box (M.11.4)	'cref' (0x6372 6566)	No	This box specifies that a box found in another location (either within the JPX file or within another file) should be considered as if it was directly contained at this location in the JPX file.
Contiguous Codestream box (M.11.8)	'jp2c' (0x6A70 3263)	No	This box contains one codestream from the JPX file, stored contiguously in a single box.
Media Data box (M.11.9)	'mdat' (0x6D64 6174)	No	This box contains generic media data, which is referenced through the Fragment Table box.
Composition box (superbox) (M.11.10)	'comp' (0x636F 6D70)	No	This box specifies how a set of compositing layers shall be combined to create the rendered result.
Composition Options box (M.11.10.1)	'copt' (0x636F 7074)	No	This box specifies generic options for the composition of multiple compositing layers.
Instruction Set box (M.11.10.2)	'inst' (0x696E 7374)	No	This box specifies the specific instructions for combining multiple compositing layers to create the rendered result.
Association box (superbox) (M.11.11)	'asoc' (0x6173 6F63)	No	This box allows several other boxes (i.e., boxes containing metadata) to be grouped together and referenced as a single entity.
Number List box (M.11.12)	'nlst' (0x6E6C 7374)	No	This box specifies what entities are associated with the data contained within an Association box.
Label box (M.11.13)	'lbl\040' (0x6C62 6C20)	No	This box specifies a textual label for either a Codestream Header, Compositing Layer Header, or Association box.
Binary Filter box (M.11.14)	'bfil' (0x6266 696C)	No	This box contains data that has been transformed as part of the storage process (such as compressed or encrypted).

Table M.13 – Boxes defined within this Recommendation | International Standard

Box name	Type	Required ?	Comments
Desired Reproductions box (superbox) (M.11.15)	'drep' (0x6472 6570)	No	This box specifies a set of transformations that must be applied to the image to guarantee a specific desired reproduction on a set of specific output devices.
Graphics Technology Standard Output box (M.11.15.1)	'gtso' (0x6774 736F)	No	This box specifies the desired reproduction of the rendered result for commercial printing and proofing systems.
ROI Description box (M.11.16)	'roid' (0x726F 6964)	No	This box specifies information about specific regions of interest in the image.
Digital Signature box (M.11.17)	'chck' (0x6368 636B)	No	This box contains a checksum or digital signature for a portion of the JPX file.
MPEG-7 Binary box (M.11.19)	'mp7b' (0x6D70 3762)	No	This box contains metadata in MPEG-7 binary format (BiM) as defined by ISO/IEC 15938.
Free box (M.11.20)	'free' (0x6672 6565)	No	This box contains data that is no longer used and may be overwritten when the file is updated.
Intellectual Property Rights (N.5.4)	'ipr' (0x6A70 3269)	No	This box contains intellectual rights information.

M.11.1 Reader Requirements box

The Reader Requirements box specifies what features or feature groups have been used in this JPX file, as well as what combination of features must be supported by a reader in order to fully use the file. The Reader Requirements box must immediately follow the File Type box, and there shall be one and only one Reader Requirements box in the file.

The type of a Reader Requirements box shall be 'rreq' (0x7272 6571'). The contents of the Reader Requirements box is as follows:

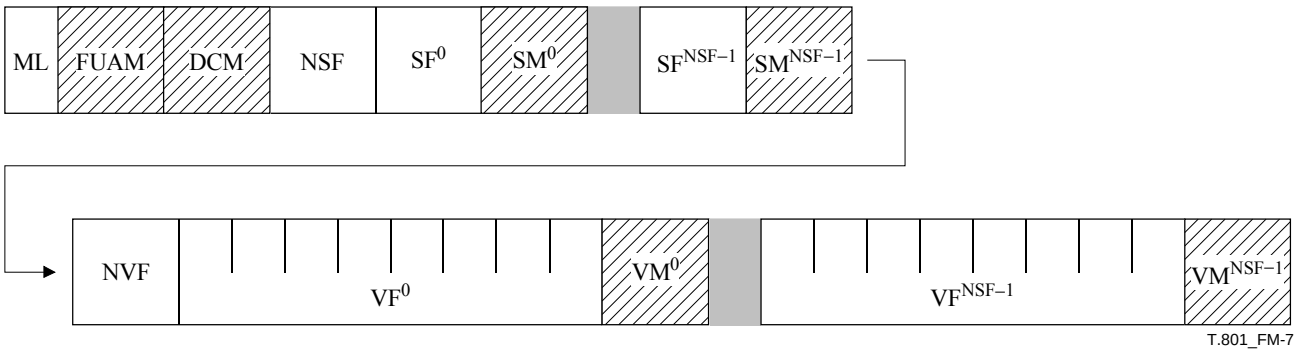

Figure M.7 – Organization of the contents of the Reader Requirements box

ML: Mask length. This field is a byte that specifies the number of bytes used for the compatibility masks. This field is encoded as a 1-byte unsigned integer.

FUAM: Fully Understand Aspects mask. This field is the mask describing the Fully Understand Aspects expression. This field is specified as a big endian integer of the size as specified by the ML field.

DCM: Decode Completely mask. This field is the mask describing the expression to display the image correctly. This field is specified as a big endian integer of the size as specified by the ML field.

NSF: Number of standard flags. This field specifies the number of standard feature flags contained within the Reader Requirements box. The value of this field shall be equal to the number of SF^i fields found within the Reader Requirements box. This field is encoded as a 2-byte big endian unsigned integer.

SF^i: Standard flag. This field specifies a standard feature flag. The number of SF^i fields shall be equal to the value of the NSF field. This field is encoded as a 2-byte big endian unsigned integer. Legal values of this field are shown in Table M.14.

SM^i: Standard mask. This field specifies the compatibility mask for the feature specified by SF^i. This field is specified as a big endian integer of the size as specified by the ML field.

NVF: Number of vendor features. This field specifies the number of vendor features specified in the Reader Requirements box. The value of this field shall be equal to the number of VF^i fields in the Reader Requirements box. This field is encoded as a 2-byte big endian unsigned integer.

VF^i: Vendor feature. This field specifies one vendor defined feature that is used in this JPX file. This field is encoded as a 128-bit UUID. Information about the feature specified by this UUID can be specified using the UUID Info box as defined in the JP2 file format.

VM^i: Vendor mask. This field specifies the compatibility mask for the feature specified by VF^i. This field is specified as a big endian integer of the size as specified by the ML field.

Table M.14 – Legal values of the SF^i field

Value	Meaning
1	Codestream contains no extensions
2	Contains multiple composition layers
3	Codestream is compressed using JPEG 2000 and requires at least a Profile 0 decoder as defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, A.10 Table A.45
4	Codestream is compressed using JPEG 2000 and requires at least a Profile 1 decoder as defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, A.10 Table A.45
5	Codestream is compressed using JPEG 2000 as defined by ITU-T Rec. T.800 | ISO/IEC 15444-1
6	Codestream is compressed using JPEG 2000 as defined in this Recommendation | International Standard
7	Codestream is compressed using DCT
8	Does not contain opacity
9	Compositing layer includes opacity channel (non-premultiplied)
10	Compositing layer includes premultiplied channel opacity
11	Compositing layer specifies opacity using a chroma-key value
12	Codestream is contiguous
13	Codestream is fragmented such that fragments are all in file and in order
14	Codestream is fragmented such that fragments are all in file but out of order
15	Codestream is fragmented such that fragments are in multiple local files
16	Codestream is fragmented such that fragments are across the internet
17	Rendered result created using compositing
18	Support for compositing layers is not required (reader can load a single, discrete compositing layer)
19	Contains multiple, discrete layers that should not be combined through either animation or compositing
20	Compositing layers each contain only a single codestream
21	Compositing layers contain multiple codestreams
22	All compositing layers are in the same colourspace
23	Compositing layers are in multiple colourspaces
24	Rendered result created without using animation
25	Animated, but first layer covers entire area and is opaque
26	Animated, but first layer does not cover the entire rendered result area
27	Animated, and no layer is reused
28	Animated, but layers are reused
29	Animated with persistent frames only
30	Animated with non-persistent frames
31	Rendered result created without using scaling
32	Rendered result involves scaling within a layer
33	Rendered result involves scaling between layers
34	Contains ROI metadata
35	Contains IPR metadata
36	Contains Content metadata

Table M.14 – Legal values of the SF^i field

Value	Meaning
37	Contains History metadata
38	Contains Creation metadata
39	Portion of file is digitally signed in a secure method
40	Portion of file is checksummed
41	Desired Graphic Arts reproduction specified
42	Compositing layer uses palettized colour
43	Compositing layer uses Restricted ICC profile
44	Compositing layer uses Any ICC profile
45	Compositing layer uses sRGB enumerated colourspace
46	Compositing layer uses sRGB-grey enumerated colourspace
47	Compositing layer uses BiLevel 1 enumerated colourspace
48	Compositing layer uses BiLevel 2 enumerated colourspace
49	Compositing layer uses YCbCr 1 enumerated colourspace
50	Compositing layer uses YCbCr 2 enumerated colourspace
51	Compositing layer uses YCbCr 3 enumerated colourspace
52	Compositing layer uses PhotoYCC enumerated colourspace
53	Compositing layer uses YCCK enumerated colourspace
54	Compositing layer uses CMY enumerated colourspace
55	Compositing layer uses CMYK enumerated colourspace
56	Compositing layer uses CIELab enumerated colourspace with default parameters
57	Compositing layer uses CIELab enumerated colourspace with parameters
58	Compositing layer uses CIEJab enumerated colourspace with default parameters
59	Compositing layer uses CIEJab enumerated colourspace with parameters
60	Compositing layer uses e-sRGB enumerated colourspace
61	Compositing layer uses ROMM–RGB enumerated colourspace
62	Compositing layers have non-square samples
63	Compositing layers have labels
64	Codestreams have labels
65	Compositing layers have different colour spaces
66	Compositing layers have different metadata

Table M.15 – Format of the contents of the Reader Requirements box

Field name	Size (bits)	Value
ML	8	1, 2, 4 or 8
EM	$8 \times ML$	Variable
DCM	$8 \times ML$	Variable
NSF	16	0-65 535
SF^i	16	0-65 535
SM^i	$8 \times ML$	Variable
NVF	16	0-65 535
VF^i	128	Variable
VM^i	$8 \times ML$	Variable

M.11.2 Data Reference box

The Data Reference box contains an array of URL's which are referenced by this file. Many of these references will be from Fragment Table boxes, specifying the location of the codestream fragments. Other references will be from Cross-Reference boxes. A JPX file shall contain zero or one Data Reference boxes, and that Data Reference box shall be at the top level of the file; it shall not be in any superboxes.

The Data Reference box is not a superbox because it does not contain only boxes.

The type of the Data Reference box shall be 'dtbl' (0x6474 626C), and its contents shall be as follows:

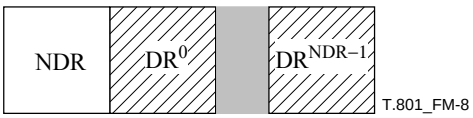

Figure M.8 – Organization of the contents of a Data Reference box

NDR: Number of data references. This field specifies the number of data references, and thus the number of URL boxes contained within this Data Reference Box.

DR^i: Data Reference URL. This field contains a Data Entry URL box, as defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, I.7.3.2. However, in this context, the Location field in the box is not specific to UUID Info Boxes. The meaning of the URL is specified in the context of the box that refers to the particular entry in the Data Reference box.

The indices of the elements in the array of DR^i fields is 1 based; a data reference of 1 in a DR^i field within a Fragment List box specifies the first Data Reference URL contained within the Data Reference Box. A data reference value of 0 is a special case that indicates that the reference is to data contained within this JPX file itself.

Table M.16 – Format of the contents of the Data Reference box

Field name	Size (bits)	Value
NDR	16	0-65 535
DR^i	Variable	Variable

M.11.3 Fragment Table box (superbox)

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

The type of the Fragment Table box shall be 'ftbl' (0x6674 626C), and its contents shall be as follows:

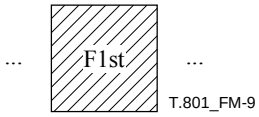

Figure M.9 – Organization of the contents of a Fragment Table box

Flst: Fragment List. This field contains a Fragment List box as specified in M.11.3.1.

M.11.3.1 Fragment List box

The Fragment List box specifies the location, length and order of each of the fragments that, once combined, form a valid and complete data stream. Depending on what box contains this particular Fragment List box, the data stream forms either a codestream (if the Fragment List box is contained in a Fragment Table box) or shared header or metadata (if the Fragment List box is contained in a Cross-Reference box).

If this Fragment List box is contained within a Fragment Table box (and thus specifies the location of a codestream), then the first offset in the fragment list shall point directly to the first byte codestream data; it shall not point to the header of the box containing the first codestream fragment.

If this Fragment List box is contained within a Cross-Reference box (and thus specifies the location of shared header or metadata), then the first offset in the fragment list shall point to the first byte of the contents of the referenced box; it shall not point to the header of the referenced box. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox.

For all other offsets in the Fragment List box, the offsets shall point directly to the first byte of the fragment data and not to the header of the box that contains that fragment.

In addition, an offset within any Fragment List shall not point into a Binary Filter box. If the JPX file does contain one or more Binary Filter boxes, then all offsets in all Fragment list boxes shall be interpreted with respect to the length of the Binary Filter boxes, as stored in the file, not the length of the data after the application of the filter.

The type of the Fragment List box shall be 'flst' (0x666C 7374) and it shall have the following contents:

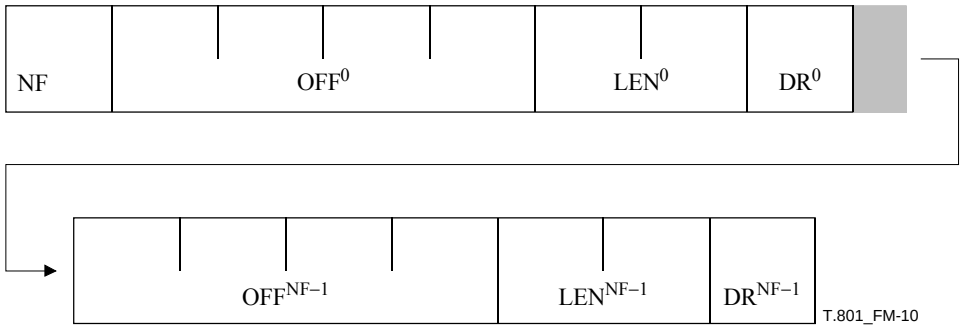

Figure M.10 – Organization of the contents of a Fragment List box

NF: Number of fragments. This field specifies the number of fragments used to contain the data stream. The number of {OFF, LEN, DR} tuples in the Fragment list box shall be the same number as the value of the NF field.

OFF^i: Offset. This field specifies the offset to the start of the fragment in the specified file. The offset is relative to the first byte of the file (for example, the first byte of the length field of the JPEG 2000 signature box header for a JPX file). This field is encoded as a 64-bit unsigned integer.

LEN^i: Length of fragment. This field specifies the length of the fragment. This value includes only the actual data and not any headers of an encapsulating box. This field is encoded as a 32-bit unsigned integer.

DR^i: Data reference. This field specifies the data file or resource that contains this fragment. If the value of this field is zero, then the fragment is contained within this file. If the value is not zero, then the fragment is contained within the file specified by this index into the array of DR^i fields in the Data Reference box, where an index value of 1 indicates the first element in the array. This field is encoded as a 16-bit unsigned integer.

Table M.17 – Format of the contents of the Fragment List box

Parameter	Size (bits)	Value
NF	16	0-65 535
OFF^i	64	12-(2^{64}–1)
LEN^i	32	0-(2^{32}–1)
DR^i	16	0-65 535

M.11.4 Cross-Reference box

If a JPX file contains multiple codestreams or compositing layers, it may be useful to share header and metadata information between those codestreams or compositing layers to minimize file size. One mechanism to share such data is to place a cross-reference to the actual metadata or header box into the Codestream Header or Compositing Layer Header box in place of the actual data. This is done using a Cross-Reference box. A JPX file may contain zero or more Cross-Reference boxes, and the Cross-Reference boxes shall be found only within Codestream Header boxes,

Compositing Layer Header boxes, or Association boxes. Also, a Cross-Reference box shall not point to another Cross-Reference box. Also, because the Cross-Reference box contains a field followed by a box, the Cross-Reference box is not a superbox.

The type of the Cross-Reference box shall be 'cref' (0x6372 6566) and it shall have the following contents:

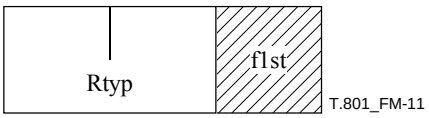

Figure M.11 – Organization of the contents of a Fragment table box

Rtyp: Referenced box type. This field specifies the actual type (as would be found in the TBox field in an actual box header) of the box referenced by this Cross-Reference box. However, a reader shall not attempt to locate a physically stored box header for the box represented by this cross-reference box, as it is legal to use a Cross-Reference box to create a new box that is not contiguously contained in other locations within this or other files, and thus the box header will not exist.

flst: Fragment List box. This box specifies the actual locations of the fragments of the referenced box. When those fragments are concatenated, in order, as specified by the Fragment List box definition, the resulting byte-stream shall be the contents of the referenced box and shall not include the box header fields. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox. The format of the Fragment List box is specified in M.11.3.1.

Table M.18 – Format of the contents of the Cross-Reference box

Parameter	Size (bits)	Value
Rtyp	32	$0-(2^{32}-1)$
flst	Variable	Variable

M.11.5 JP2 Header box (superbox)

The JP2 Header box is syntactically unchanged from the structures defined in the JP2 file format. However, if the JPX file contains multiple codestreams or multiple compositing layers, then any boxes contained within the JP2 Header box shall be considered as defaults for all codestreams and compositing layers. For example, if a Compositing Layer Header box does not specify a Colourspace specification, then a reader shall apply the Colourspace Specification contained within the JP2 Header box to that particular compositing layer.

Also, if the codestream is specified by the JP2 Header box, then the semantic relationship of the Image Header box and Bits Per Component box contained within the JP2 Header box shall follow the rules defined in M.11.5.1 and M.11.5.2 respectively.

Also, the JPX file format allows the JP2 Header box to be located anywhere at the top-level of the file (but not within any superbox). However, certain fallback positions, such as the JPX baseline definition, may restrict the placement of this box. In addition, if this file does not require the JP2 Header box to meet the requirements of a fallback position, nor does it use the JP2 Header box to specify default information for multiple compositing layers or codestreams, then this box may be omitted from the file.

M.11.5.1 Image Header box

The format and structure of the Image Header box is identical to that defined in I.5.3.1 in ITU-T Rec. T.800 | ISO/IEC 15444-1 in the JP2 file format. However, the additional values of the fields within that box are defined for the JPX file format. In a JPX file, this box may be found either within the JP2 Header box or within a Codestream Header box.

The type of the Image Header box shall be 'ihdr' (0x6968 6472) and contents of the box shall have the following format:

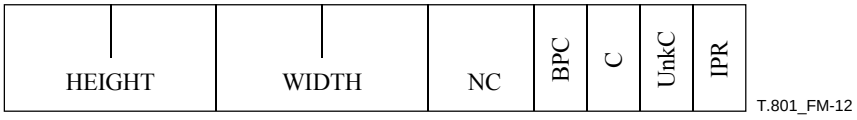

Figure M.12 – Organization of the contents of an Image Header box

HEIGHT: Image area height. The value of this field is identical to that defined for the JP2 file format.

WIDTH: Image area width. The value of this field is identical to that defined for the JP2 file format.

NC: Number of components. The value of this field is identical to that defined for the JP2 file format.

BPC: Bits per component. This parameter specifies the bit depth of the fully decompressed component, minus 1, and is stored as a 1-byte field. This shall represent the bit depth of the component after any inverse multiple component transformation or inverse non-linearity transformation extension has been applied. However, if the compression type of the codestream corresponding to this Image Header box is not JPEG 2000 or if neither the multiple component or non-linearity extension are used within the codestream, then the value of the field in this box shall match the respective bits per component data in the respective codestream format specification.

If the bit depth is the same for all components, then this parameter specifies that bit depth and shall be equivalent to the bit depth specified within the codestream using the data structures defined for that particular codestream format. If the components vary in bit depth, then the value of this field shall be 255, and the superbox that contains this Image Header box (either the JP2 Header box or a Codestream Header box) must contain a Bits Per Component box defining the bit depth of each component (as defined in I.5.3.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1 in the JP2 file format). Components should be considered to have different bit depths if either the magnitude or sign of the bit depth of the components differ.

The low 7-bits of the value indicate the bit depth of the components. The high-bit indicates whether the components are signed or unsigned. If the high-bit is 1, then the components contain signed values. If the high-bit is 0, then the components contain unsigned values.

C: Compression type. This parameter specifies the compression algorithm used to compress the image data. Legal values of this field are as follows:

Table M.19 – Legal C values

Value	Meaning
0	**Uncompressed.** Picture data is stored in component interleaved format, encoded at the bit depth as specified by the BPC field. This value is only permitted for codestreams where all components are encoded at the same bit depth. When the bit depth of each component is not 8, sample values shall be packed into bytes so that no bits are unused between samples. However, each sample shall begin on a byte boundary and padding bits having value zero shall be inserted after the last sample of a scan line as necessary to fill out the last byte of the scan line. Simple values appear in component-interleaved order. When multiple sample values are packed into a byte, the first sample shall appear in the most significant bits of the byte. When a sample is larger than a byte, its most significant bit shall appear in earlier bytes.
1	**ITU-T Rec. T.4, the basic algorithm known as MH (Modified Huffman).** This value is only permitted for bi-level images.
2	**ITU-T Rec. T.4, commonly known as MR (Modified READ).** This value is only permitted for bi-level images.
3	**CCITT Rec. T.6, commonly known as MMR (Modified Modified READ).** This value is only permitted for bi-level images.
4	**ITU-T Rec. T.82 | ISO/IEC 11544.** Commonly known as JBIG. This value is only permitted for bi-level images.
5	**CCITT Rec. T.81 | ISO/IEC 10918-1 or ITU-T Rec.T.84 | ISO/IEC 10918-3**. Commonly known as JPEG. This compressed image stream shall conform to the syntax of interchange format for compressed image data as specified in the aforementioned standards. This value is only permitted for continuous tone, greyscale or colour images.
6	**JPEG-LS.**
7	JPEG 2000 compression (as defined by ISO/IEC 15444).
8	**JBIG2.**
9	**ITU-T Rec. T.82 | ISO/IEC 11544.** Commonly known as JBIG. This value is permitted for any image permitted by the JBIG standard.
	All other values reserved.

UnkC: Colourspace Unknown. The value of this field is identical to that defined for the JP2 file format.

IPR: Intellectual Property. The value of this field is identical to that defined for the JP2 file format.

Table M.20 – BPC values

Values (bits) MSB LSB	Component sample precision
x000 0000 to x010 0101	Component bit depth = value + 1. From 1 bit deep through 38 bits deep respectively (counting the sign bit, if appropriate)
0xxx xxxx	Components are unsigned values
1xxx xxxx	Components are signed values
1111 1111	Components vary in bit depth
	All other values reserved

M.11.5.2 Bits Per Component box

The Bits Per Component box specifies the bit depth of each fully decompressed component. This shall represent the bit depth of the component after any inverse multiple component transformation or reverse non-linearity transformation extension has been applied. components in the codestream. The structure of this box is identical to that defined in I.5.3.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1 in the JP2 file format. However, if the compression type of the codestream corresponding to this Bits Per Component box is not JPEG 2000 or if neither the multiple component or non-linearity extension are used within the codestream, then the value of the field in this box shall match the respective bits per component data in the respective codestream format specification.

Table M.21 – Format of the contents of the Image Header box

Field name	Size (bits)	Value
HEIGHT	32	1-(2^{32}–1)
WIDTH	32	1-(2^{32}–1)
NC	16	1-16 384
BPC	8	See Table M.20
C	8	7
UnkC	8	0-1
IPR	8	0-1

M.11.6 Codestream Header box (superbox)

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

Both codestreams and Codestream Header boxes are numbered separately, starting with 0, by their order in the file. Codestream Header box *i* shall be applied to codestream *i*. There shall either be one Codestream Header box in the file for each codestream, or there shall be zero Codestream Header boxes in the file. In the event that there are zero Codestream Header boxes, then the header information for all of the codestreams shall be taken to be the default header information contained within the JP2 Header box.

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

The type of a Codestream Header box shall be 'jpch' (0x6A70 6368). The contents of a Codestream Header box is as follows:

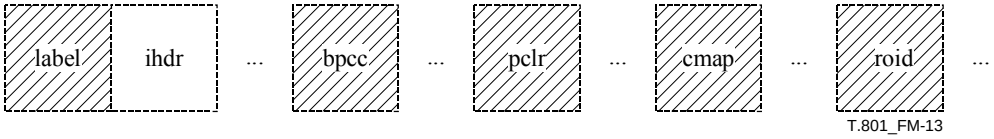

Figure M.13 – Organization of the contents of a Codestream Header box

label: Label box. This box specifies a label for this codestream. Its structure is specified in M.11.13.

ihdr: Image Header box. This box specifies information about this codestream, such as its height and width. Its structure is specified in M.11.5.1. If the JP2 Header box contains an Image Header box that accurately specifies this codetream, then it is not required that this Codestream Header box contain an Image Header box. Otherwise, this Codestream Header box shall contain an Image Header box. In addition, if the IPR flag in the Image Header box is set to 0, indicating no intellectual property rights information is specified for this codestream, then this Codestream Header box shall not contain an IPR box, and the reader shall not apply the contents of an IPR box at the top level of the file to this codestream.

bpcc: Bits Per Component box. This box specifies the bit depth of each component in the codestream after decompression. Its structure is specified in M.11.5.2.

pclr: Palette box. This box defines the palette to use to create multiple components from a single component. Its structure is specified in I.5.3.4 in ITU-T Rec. T.800 | ISO/IEC 15444-1 of the JP2 file format.

cmap: Component Mapping box. This box defines how image channels are identified from the actual components in the codestream. Its structure is specified in I.5.3.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1 of the JP2 file format.

roid: ROI Description box. This box describes regions of interest within this codestream. These ROIs may or may not be directly associated with coded ROIs in the codestream. Its structure is defined in M.11.16.

The Codestream Header box may also contain other metadata boxes, including an IPR box, or cross-references to other boxes. If the Codestream Header contains a cross-reference, then the box pointed to by the cross-reference shall be considered as if it was physically stored in this Codestream Header box.

Also, if any of these boxes are contained within the JP2 header box and are not contained within this Codestream Header box, then those boxes should also be applied to this codestream.

M.11.7 Compositing Layer Header box (superbox)

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

The type of a Compositing Layer Header box shall be 'jplh' (0x6A70 6C68). The contents of a Compositing Layer Header box is as follows:

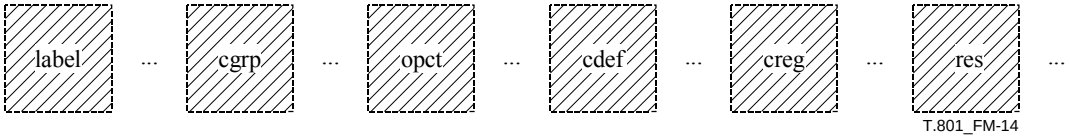

Figure M.14 – Organization of the contents of a Compositing Layer Header box

label: Label box. This box specifies a label for this compositing layer. Its structure is specified in M.11.13.

cgrp: Colour Group box. This box contains the complete colourspace specification (represented by a sequence of colour specification boxes) for this compositing layer. Its structure is specified in

M.11.7.1. If neither this box nor a cross-reference to another Colour Group box are found within the Compositing Layer Header box, then the default value of the colourspace specification for this compositing layer shall be the set of individual Colour Specification boxes found within the JP2 Header box. These Colour Specification boxes shall not be encapsulated within a Colour Group box.

opct: Opacity box. This box specifies that this compositing layer uses a simple opacity mode. Its structure is specified in M.11.7.6. If the Compositing Layer Header box contains an Opacity box, then it shall not contain a Channel Definition box, and any default Channel Definition box in the JP2 Header box shall be ignored for this compositing layer.

cdef: Channel Definition box. This box defines the channels in the image. Its structure is specified in I.5.3.6 in ITU-T Rec. T.800 | ISO/IEC 15444-1 of the JP2 file format. This box shall not be found if this Compositing Layer Header box contains an Opacity box.

creg: Codestream Registration box. This box specifies the spatial registration between the codestreams in this compositing layer. Its structure is specified in M.11.7.7. If any Compositing Layer Header box contains a Codestream Registration box, then every Compositing Layer Header box shall contain a Codestream Registration box.

res: Resolution box. This box specifies the capture and default display resolutions of the image. Its structure is specified in I.5.3.7 in ITU-T Rec. T.800 | ISO/IEC 15444-1 of the JP2 file format.

The Compositing Layer Header box may also contain other metadata boxes, including an IPR box, or cross-references to other boxes. If the Compositing Layer Header contains a cross-reference, then the box pointed to by the cross-reference shall be considered as if it was physically stored in this Compositing Layer Header box.

Also, if any of these boxes are contained within the JP2 header box and are not contained within this Compositing Layer Header box, then those boxes should also be applied to this compositing layer.

M.11.7.1 Colour Group box (superbox)

A Colour group box contains a set of related, equivalent, colour specification methods. When interpreting the colourspace of a codestream, any colour specification method contained within the specified Colour Group box may be used. This box shall be found only within a Compositing Layer Header box. This encapsulation reduces the storage overhead of sharing an entire set of colour specifications between layers.

A Colour Group box (or the JP2 Header box) shall not contain multiple Colour Specifications boxes with a METH value of 1 (Enumerated method), or multiple boxes with a METH value of 2 (Restricted ICC method). A single colour group may contain multiple Colour Specification boxes with a METH value of 3 (Any ICC method) or 4 (Vendor Colour method). Multiple ICC profiles (of the unrestricted variety) may be used to specify a particular colourspace with varying degrees of complexity (1D LUT's vs 3D LUT's), and multiple Vendor Colour methods may be used to specify multiple non-ICC based representations of the colourspace.

The JPX file may contain zero Colour Group boxes, which indicates that all compositing layers are in the colourspace specified within the JP2 Header Box (through a set of Colour Specification boxes stored directly within the JP2 Header boxes and not encapsulated within a Colour Group box).

However, if the file does not contain a colourspace specification within the JP2 Header Box (or does not contain the JP2 Header Box), then the JPX file shall contain at least one Colour Group box.

The type of a Colour Group box shall be 'cgrp' (0x6367 7270). The contents of a Colour Group box is as follows:

Figure M.15 – Organization of the contents of a Colour Group box

CLR^{i}: Colour Specification Box. This Colour Specification box specifies one method by which the colourspace of a particular codestream can be interpreted. The format of the Colour Specification box is specified in M.11.7.2

M.11.7.2 Colour Specification box

Each Colour Specification box defines one method by which an application can interpret the colourspace of the decompressed image data. This colour specification is to be applied to the image data after it has been decompressed

and after any reverse multiple component transformation and reverse non-linearity transformation has been applied to the decompressed image data.

Colour Specification boxes may be found in either the JP2 Header box or in Colour Group boxes. In total, a JPX file may contain multiple Colour Specification boxes, and either the JP2 Header box or a particular Colour Group box may contain multiple Colour Specification boxes. However, all JPX files shall contain at least one Colour Specification box.

The box type and binary structure of a Colour Specification box is identical to that defined in the JP2 file format. However, to clarify the extensibility of the box with respect to defining new colour specification methods, the way in which it is described is changed within JPX. The contents of a Colour Specification box is as follows:

Figure M.16 – Organization of the contents of a Colour Specification box

METH: Specification method. This field specifies the method used by this Colour Specification box to define the colourspace of the decompressed image. This field is encoded as a 1-byte unsigned integer. The legal values of the METH field are as follows:

Table M.22 – Legal METH values

Value	Meaning
1	**Enumerated method.** This Colour Specification box indicates that the colourspace of the codestream is specified by an enumerated integer code. The definition of the format of this method is identical to the Enumerated Method in JP2. However, the JPX file format defines additional enumerated values as specified in M.11.7.3.1, as well as additional parameters for some enumerated colourspaces as specified in M.11.7.4.
2	**Restricted ICC method.** This Colour Specification box indicates that the colourspace of the codestream is specified by an embedded ICC profile of restricted type. The definition of and format of this method is identical to the Restricted ICC method defined in the JP2 file format, I.5.3.3 in ITU-T Rec. T.800 | ISO/IEC 15444-1.
3	**Any ICC method.** This Colour Specification box indicates that the colourspace of the codestream is specified by an embedded input ICC profile. Contrary to the Restricted ICC method defined in the JP2 file format, this method allows for any input ICC profile, defined by ICC-1. The binary format of the METHDAT field is specified in M.11.7.3.2.
4	**Vendor Colour method.** This Colour Specification box indicates that the colourspace of the codestream is specified by a unique vendor defined code. The binary format of the METHDAT field is specified in M.11.7.3.3.
	All other values reserved. For any value of the METH field, the length of the METHDAT field may not be 0, and applications shall not expect that the APPROX field be the last field in the box if the value of the METH field is not understood. In this case, a conforming reader shall ignore the entire Colour Specification box.

PREC: Precedence. This field specifies the precedence of this Colour Specification box, with respect to the other Colour Specification boxes within the same Colour Group box, or the JP2 Header box if this Colour Specification box is in the JP2 Header box. It is suggested, but not required, that conforming readers use the colour specification method that is supported with the highest precedence. This field is specified as a signed 1-byte integer.

APPROX: Colourspace approximation. This field specifies the extent to which this colour specification method approximates the "correct" definition of the colourspace. An example of approximation of a colourspace specification may be increased quantization in look-up tables or rounding in matrix coefficients. This field is specified as 1-byte unsigned integer. Legal values of this field are as follows:

Contrary to the APPROX field in a JP2 file (a file with "jp2\040" in the BR field in the File Type box), a value of 0 in the APPROX field is illegal in a JPX file (a file with "jpx\040" in the BR field in the File Type box). JPX writers are required to properly indicate the degree of approximation of the colour specification to the correct definition of the colourspace. This does not specify if the writer of the file knew the actual colourspace of the image data. If the actual colourspace is unknown, then the value of the UnkC field in the Image Header box shall be set to 1 and the APPROX field shall specify the degree to which this Colour Specification box matches the correct definition of the assumed or target colourspace.

In addition, high values of the APPROX field (indicating poor approximation) shall not be used to hide that the multiple Colour Specification boxes in either a Colour Group box or the JP2 Header box actually represent different colourspaces; the specification of multiple different colourspaces within a single Colour Group box is illegal.

Table M.23 – Legal APPROX values

Value	Meaning
1	This colour specification method accurately represents the correct definition of the colourspace
2	This colour specification method approximates the correct definition of the colourspace with exceptional quality
3	This colour specification method approximates the correct definition of the colourspace with reasonable quality
4	This colour specification method approximates the correct definition of the colourspace with poor quality
	All other values reserved

Table M.24 – Format of the contents of the Colour Specification box

Field name	Size (bits)	Value
METH	8	1-4
PREC	8	–128-127
APPROX	8	1-4
METHDAT	Variable	Variable

M.11.7.3 METHDAT field specifications in the Colour Specification box

The following subclauses define the fields and values that make up the METHDAT field for each defined colour specification method.

M.11.7.3.1 METHDAT values for the Enumerated method

The contents of the METHDAT field for Colour Specification boxes using the Enumerated method is defined as follows:

Figure M.17 – Organization of the contents of the METHDAT field for the Enumerated method

EnumCS: Enumerated colourspace. This field specifies the colourspace of the image using an integer code. To correctly interpret the colour of an image using an enumerated colourspace, the application must know the definition of that colourspace internally. This field contains a 4-byte big endian unsigned integer value indicating the colourspace of the image. Valid EnumCS values are those values defined for the Enumerated method in the JP2 file format and the values defined as follows (Table M.25).

Table M.25 – Additional legal EnumCS values

Value	Meaning
0	**Bi-level**: This value shall be used to indicate bi-level images. Each image sample is one bit: 0 = white, 1 = black.
1	**YC_bC_r(1)**: This is a format often used for data that originated from a video signal. The colourspace is based on ITU-R Rec. BT.709-4. The valid ranges of the YC_bC_r components in this space is limited to less than the full range that could be represented given an 8-bit representation. ITU-R Rec. BT.601-5 specifies these ranges as well as defines a 3×3 matrix transformation that can be used to convert these samples into RGB.

Table M.25 – Additional legal EnumCS values

Value	Meaning
3	**YC_bC_r(2)**: This is the most commonly used format for image data that was originally captured in RGB (uncalibrated format). The colourspace is based on ITU-R Rec. BT.601-5. The valid ranges of the YC_bC_r components in this space is [0, 255] for Y, and [–128, 127] for C_b and C_r (stored with an offset of 128 to convert the range to [0, 255]). These ranges are different from the ones defined in ITU-R Rec. BT.601-5. ITU-R Rec. BT.601-5 specifies a 3×3 matrix transformation that can be used to convert these samples into RGB.
4	**YC_bC_r(3)**: This is a format often used for data that originated from a video signal. The colourspace is based on ITU-R Rec. BT.601-5. The valid ranges of the YC_bC_r components in this space is limited to less than the full range that could be represented given an 8-bit representation. ITU-R Rec. BT.601-5 specifies these ranges as well as defines a 3×3 matrix transformation that can be used to convert these samples into RGB.
9	**PhotoYCC**: This is the colour encoding method used in the Photo CD™ system. The colourspace is based on ITU-R Rec. BT.709 reference primaries. ITU-R Rec. BT.709 linear RGB image signals are transformed to non-linear R'G'B' values to YCC corresponding to ITU-R Rec. BT.601-5. Details of this encoding method can be found in Kodak Photo CD products, *A Planning Guide for Developers*, Eastman Kodak Company, Part No. DC1200R and also in Kodak Photo CD Information Bulletin PCD045.
11	**CMY**: The encoded data consists of samples of Cyan, Magenta and Yellow samples, directly suitable for printing on typical CMY devices. A value of 0 shall indicate 0% ink coverages, whereas a value of $2^{BPS}–1$ shall indicate 100% ink coverage for a given component sample.
12	**CMYK**: As CMY above, except that there is also a black (K) ink component. Ink coverage is defined as above.
13	**YCCK**: This is the result of transforming original CMYK type data by computing $R = (2^{BPS}–1)–C$, $G = (2^{BPS}–1)–M$, and $B = (2^{BPS}–1)–Y$, applying the RGB to YCC transformation specified for YC_bC_r (2) above, and then recombining the result with the unmodified K-sample. This transformation is intended to be the same as that specified in Adobe Postscript.
14	**CIELab**: The CIE 1976 (L*a*b*) colourspace. A colourspace defined by the CIE (Commission Internationale de l'Eclairage), having approximately equal visually perceptible differences between equally spaced points throughout the space. The three components are L*, or Lightness, and a* and b* in chrominance. For this colourspace, additional Enumerated parameters are specified in the EP field as specified in M.11.7.4.1
15	**Bi-level(2)**: This value shall be used to indicate bi-level images. Each image sample is one bit: 1 = white, 0 = black.
18	**sYCC** as defined by IEC 61966-2-1, Amd.1. NOTE – It is not recommended to use ICT or RCT specified in ITU-T Rec. T.800 | ISO/IEC 15444-1 Annex G with sYCC image data. See ITU-T Rec. T.800 | ISO/IEC 15444-1, J.15, for guidelines on handling YCC codestreams.
19	**CIEJab**: As defined by CIE Colour Appearance Model 97s, CIE Publication 131. For this colourspace, additional Enumerated parameters are specified in the EP field as specified in M.11.7.4.2.
20	**e-sRGB**: As defined by PIMA 7667.
21	**ROMM-RGB**: As defined by PIMA 7666.
22	**YPbPr(1125/60)**: This is the well-known colour space and value definition for the HDTV (1125/60/2:1) system for production and international program exchange specified by ITU-R Rec. BT.709-3. The Recommendation specifies the colour space conversion matrix from RGB to YPbPr(1125/60) and the range of values of each component. The matrix is different from the 1250/50 system. In the 8-bit/component case, the range of values of each component is [1, 254], the black level of Y is 16, the achromatic level of Pb/Pr is 128, the nominal peak of Y is 235, and the nominal extremes of Pb/Pr are 16 and 240. In the 10-bit case, these values are defined in a similar manner.
23	**YPbPr(1250/50)**: This is the well-known colour space and value definition for the HDTV (1250/50/2:1) system for production and international program exchange specified by ITU-R Rec. BT.709-3. The Recommendation specifies the colour space conversion matrix from RGB to YPbPr(1250/50) and the range of values of each component. The matrix is different from the 1125/60 system. In the 8-bit/component case, the range of values of each component is [1, 254], the black level of Y is 16, the achromatic level of Pb/Pr is 128, the nominal peak of Y is 235, and the nominal extremes of Pb/Pr are 16 and 240. In the 10-bit case, these values are defined in a similar manner.
24	**e-sYCC**: e-sRGB based YCC colourspace as defined by PIMA 7667 Annex B.
	All other values reserved.

The generic RGB and grayscale spaces from the SPIFF file format are explicitly not included. Applications wishing to transcode SPIFF images using colourspaces 8 and 10 should specify, within the JPX file, the colourspace definition that a reader shall use to unambiguously interpret the image data. In many cases, this will be the sRGB or sRGB-greyscale spaces from JP2. In addition, the file writer should set the UnkC field in the Image Header box indicating that the actual colourspace is not known.

EP: Enumerated parameters. This field contains a series of parameters that augment the generic colourspace definition specified by EnumCS. Together, the EnumCS and EP fields describe the colourspace and how that colour data has been encoded in the JPX file. For example, the CIELab colourspace as described by ITU-T Rec. T.42 requires several parameters to describe the ITU encoding of the colour data. The format and value of the EP field is defined individually for each EnumCS as required. If a value of EP is not defined for a particular value of EnumCS, then the length of the EP field for that EnumCS value shall be zero, indicating that the EnumCS value alone describes the colourspace or default values are used as defined by the referenced colourspace definition. The format and values of the EP field are defined in M.11.7.4. However, the EP field shall be the last field in the Colour Specification box and shall be all bytes in the box following the EnumCS field to the end of the box.

Table M.26 – Format of the contents of the METHDAT field for the Enumerated method

Field name	Size (bits)	Value
EnumCS	32	0-(2^{32}–1)
EP	Variable	Variable

M.11.7.3.2 METHDAT values for the Any ICC method

The contents of the METHDAT field for Colour Specification boxes using the Any ICC method is defined as follows:

Figure M.18 – Organization of the contents of the METHDAT field for the Any ICC method

Profile: ICC Profile. This field contains an ICC input profile as defined by ICC-1, specifying the transformation between the decompressed code values and the PCS. Any input ICC profile, regardless of profile class, may be contained within this field.

Table M.27 – Format of the contents of the METHDAT field for the Any ICC method

Field name	Size (bits)	Value
PROFILE	Variable	Variable

M.11.7.3.3 METHDAT values for the Vendor Colour method

The contents of the METHDAT field for Colour Specification boxes using the Vendor Colour method is defined as follows:

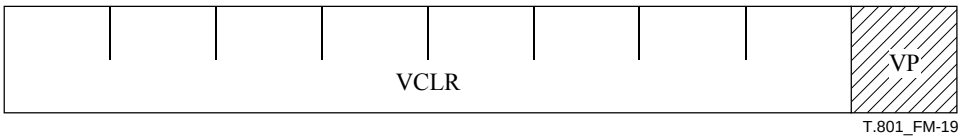

Figure M.19 – Organization of the contents of the METHDAT field for the Vendor Colour method

VCLR: Vendor Defined Code. This field specifies the colourspace of the image using a UUID. To correctly interpret the colour of an image using a Vendor defined colourspace, the application must know the definition of that colourspace internally. This field contains a 16-byte UUID indicating the colourspace of the image. These values are defined and shared by individual vendors and are outside the scope of this Recommendation | International Standard.

VP: Vendor parameters. This field specifies a series of parameters that augment the generic colourspace definition specified by VCLR. Together, the VCLR and VP fields unambiguously describe the colourspace. The format and value of the VP field is defined individually for each VCLR value as required. If a value of VP is not defined for a particular value of VCLR, then the length of the VP field for that VCLR value shall be zero, indicating that the VCLR value alone unambiguously describes the colourspace, or default values are used as defined by the referenced colourspace definition. The format and values of the VP field are defined by each individual vendor colourspace definition, and are outside of the scope of this Recommendation | International Standard. However, the VP field shall be the last field in the Colour Specification box and shall be all bytes in the box following the VCLR field to the end of the box.

Table M.28 – Format of the contents of the METHDAT field for the Vendor Colour method

Field name	Size (bits)	Value
VCLR	128	Variable
VP	Variable	Variable

M.11.7.4 EP field format and values

This field defines the format and values of the EP fields for Colour Specification boxes using the Enumerated method. If an EP field is not defined for a particular value of the EnumCS field, then the length of the EP field shall be zero.

M.11.7.4.1 EP field format for the CIELab colourspace

If the value of EnumCS is 14, specifying that the layer is encoded in the CIELab colourspace, then the format of the EP field shall be as follows:

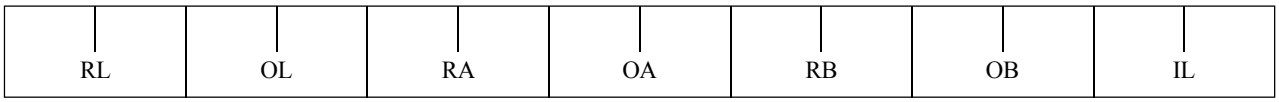

Figure M.20 – Organization of the contents of the EP field for the CIELab (EnumCS = 14)

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

$$N_L = \frac{2^{n_L} - 1}{RL} \times L^* + OL$$
$$N_a = \frac{2^{n_a} - 1}{RA} \times a^* + OA \qquad \text{(M-18)}$$
$$N_b = \frac{2^{n_b} - 1}{RB} \times b^* + OB$$

The IL field specifies the illuminant data used in calculating the CIELab values.

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

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

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

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

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

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

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

Table M.29 – Standard illuminant values for CIELab

Illuminant	Standard IL field value
CIE Illuminant D50	0x0044 3530
CIE Illuminant D65	0x0044 3635
CIE Illuminant D75	0x0044 3735
CIE Illuminant SA	0x0000 5341
CIE Illuminant SC	0x0000 5343
CIE Illuminant F2	0x0000 4632
CIE Illuminant F7	0x0000 4637
CIE Illuminant F11	0x0046 3131

When the illuminant is specified by a colour temperature, then the 4 bytes consist of the string 'CT', followed by two unsigned bytes representing the temperature of the illuminant in degrees Kelvin as a 2-byte big endian unsigned integer. For example, a 7500K illuminant is represented by the 4 bytes 0x4354 1D4C.

When the EP fields are omitted for the CIELab colourspace, then the following default values shall be used. The default L*, a* and b* range parameters are 100, 170 and 200. The default L*, a* and b* offset values are 0, 2^(N_a–1) and 2^(N_b–2) + 2^(N_b–3). These defaults correspond to the CIELab encoding in ITU-T Rec. T.42. The default value of the IL field is 0x0044 3530, specifying CIE Illuminant D50.

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

Table M.30 – Format of the contents of the EP field for CIELab (EnumCS = 14)

Field name	Size (bits)	Value
RL	32	0-(2^{32}–1)
OL	32	0-(2^{32}–1)
RA	32	0-(2^{32}–1)
OA	32	0-(2^{32}–1)
RB	32	0-(2^{32}–1)
OB	32	0-(2^{32}–1)
IL	32	Variable

M.11.7.4.2 EP field format for the CIEJab colourspace

If the value of EnumCS is 19, specifying that the layer is encoded in the CIEJab colourspace, then the format of the EP field shall be as follows:

Figure M.21 – Organization of the contents of the EP field for the CIEJab (EnumCS = 19)

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

$$N_J = \frac{2^{n_J} - 1}{RJ} \times J + OJ$$
$$N_a = \frac{2^{n_a} - 1}{RA} \times a + OA \qquad \text{(M-19)}$$
$$N_b = \frac{2^{n_b} - 1}{RB} \times b + OB$$

RJ: Range for J. This field specifies the *RJ* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 100 for RJ shall be used.

OJ: Offset for J. This field specifies the *OJ* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value 0 shall be used for OJ.

RA: Range for a. This field specifies the *RA* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 255 for RA shall be used.

OA: Offset for a. This field specifies the *OA* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value 2^{b-1} shall be used for OA, where *b* is the number of bits per sample for the 'a' channel.

RB: Range for b. This field specifies the *RB* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value of 255 for RB shall be used.

OB: Offset for b. This field specifies the *OB* value from Equation M-19. It is encoded as a 4-byte big endian unsigned integer. If the EP field is not specified for this Colour Specification box, then the value 2^{b-1} shall be used for OB, where *b* is the number of bits per sample for the 'b' channel.

Table M.31 – Format of the contents of the EP field for CIEJab (EnumCS = 19)

Field name	Size (bits)	Value
RJ	32	$0-(2^{32}-1)$
OJ	32	$0-(2^{32}-1)$
RA	32	$0-(2^{32}-1)$
OA	32	$0-(2^{32}-1)$
RB	32	$0-(2^{32}-1)$
OB	32	$0-(2^{32}-1)$

M.11.7.5 Channel Definition box

The binary format of the Channel Definition box is identical to that defined in ITU-T Rec. T.800 | ISO/IEC 15444-1, I.5.3.6. However, in a JPX file that is not readable by a JP2 reader, or in a codestream in a JPX file that will not be read by a JP2 reader, any channel may be associated with any colour or type. The following additional value of the $Asoc^i$ field are normatively defined:

Table M.32 – Colours indicated by the $Asoc^i$ field

Class of colourspace	Colour indicated by the following value of the $Asoc^i$ field			
	1	**2**	**3**	**4**
RGB	R	G	B	
Greyscale	Y			
XYZ	X	Y	Z	
Lab	L	a	b	
Luv	L	u	v	
YC_bC_r	Y	C_b	C_r	
Yxy	Y	x	y	
HSV	H	S	V	
HLS	H	L	S	
CMYK	C	M	Y	K
CMY	C	M	Y	
Jab	J	a	b	
n colour colourspaces	1	2	3	4

M.11.7.6 Opacity box

The Opacity box provides a minimal-overhead mechanism for specifying opacity through a chroma-key or specifying that a particular compositing layer contains only colour channels followed by a single opacity channel. If a Compositing Layer Header box contains an Opacity box, then it shall not contain a Channel Definition box. Compositing layers that require a channel definition more complex than can be defined using an Opacity box shall use a Channel Definition box. Each Compositing Layer Header box shall contain zero or one Opacity boxes, and Opacity boxes shall be found in no other locations in the file.

Chroma-keyed opacity is a form of paletization and as such images using chroma-keyed opacity must obey similar rules to full palettized images with respect to lossy compression. In either case, differences between the original image and the decompressed images reflect errors in a space that does not directly map to visual perception, and thus should not be coded or decompressed in a lossy mode. However, for chroma-key values, in contrast to a fully palettized component, only the samples of the image that are of the chroma-key value must be encoded and decoded losslessly. Joint lossless encoding of the chroma-keyed region and lossy coding of the remaining image region can be achieved using a ROI within the codestream.

The type of the Opacity box shall be 'opct' (0x6F70 6374). The contents of this box shall be as follows:

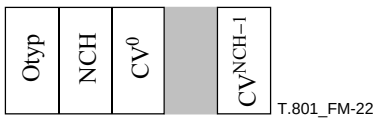

Figure M.22 – Organization of the contents of an Opacity box

Otyp: Opacity type. This field specifies the type of opacity used by this compositing layer. This field is encoded as a 1-byte unsigned integer. Legal values of the Otyp field are as follows:

Table M.33 – Otyp field values

Value	Meaning
0	The last channel in this compositing layer is an opacity channel and all other channels are colour channels where the channel association is equal to the channel number +1. For example, a four-channel compositing layer would contain 3 colour channels (with associations 1, 2 and 3 respectively) followed by an opacity channel. If the value of Otyp is 0, then the NCH, PR and CV^i fields shall not be found.
1	The last channel in this compositing layer is a premultiplied opacity channel and all other channels are colour channels where the channel association is equal to the channel number +1. For example, a four-channel compositing layer would contain 3 colour channels (with associations 1, 2 and 3 respectively) followed by a premultiplied opacity channel. If the value of Otyp is 0, then the NCH, PR and CV^i fields shall not be found.
2	This compositing layer specifies that samples of a particular colour shall be considered fully transparent (chroma-key). The chroma-key colour is specified by the NCH, PR and CV^i fields.
	All other values reserved.

NCH: Number of channels. This field specifies the number of channels used to specify the chroma-key colour. This value shall be equal to the number of channels in the compositing layer. This field is specified as a 1-byte unsigned integer.

CV^i: Chroma-key value. This field specifies the value of channel *i* for the chroma-key colour. Samples that match the chroma-key value for all channels shall be considered fully transparent. The size of this field is specified by the bit depth of the corresponding channel. If the value is not a multiple of 8, then each CV^i value shall be padded to a multiple of 8 bits with bits equal to the sign bit and the actual value shall be stored in the low-order bits of the padded value. For example, if the depth of a channel is a signed 10-bit value, then the CV^i value shall be stored in the low 10 bits of a 16-bit field and the high-order 6 bits shall be all equal to the sign bit of the value in this CV^i field.

Table M.34 – Format of the contents of the Opacity box

Field name	Size (bits)	Value
Otyp	8	0-2
NCH	8 0	0-255; if Otyp ≠ 2 Not applicable; if Otyp = 2
CV^i	Variable 0	Variable; if Otyp ≠ 2 Not applicable; if Otyp = 2

M.11.7.7 Codestream Registration box

When combining multiple codestreams to create a single compositing layer, it is important that the reference grids of those codestreams be properly registered to ensure the registration of the individual samples from the multiple components. This box specifies how those codestreams shall be registered when rendering the layer. A Compositing Layer Header box shall contain zero or one Codestream Registration boxes, and Codestream Registration boxes shall be found in no other locations in the file; a Codestream Registration box shall not be placed into the JP2 Header box to specify a default registration. If any Compositing Layer Header box contains a Codestream Registration box, then every Compositing Layer Header box shall contain a Codestream Registration box. If this Compositing Layer Header box does not contain a Codestream Registration box, then the compositing layer shall be represented by one and only one codestream.

If codestream registration is not specified for a particular compositing layer, then the codestreams in that compositing layer shall be aligned by directly aligning their reference grids at both (0,0) and (1,1).

If a Codestream Registration box exists, then the default display resolution (specified within a Resolution box with the same Compositing Layer Header box) applies to the compositing layer registration grid.

This registration is specified with respect to an independent compositing layer registration grid.

The type of the Codestream Registration box shall be 'creg' (0x6372 6567). The contents of this box shall be as follows:

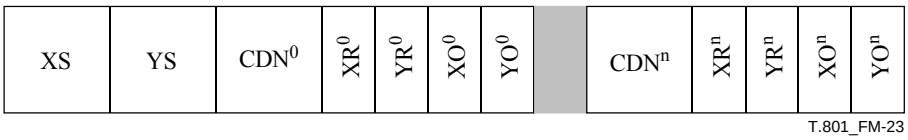

Figure M.23 – Organization of the contents of a Codestream Registration box

XS: Horizontal grid size. This field specifies the number of horizontal grid points on the compositing layer registration grid used to measure the distance between the reference grids of the individual codestreams. This field is encoded as a 2-byte unsigned integer.

YS: Vertical grid size. This field specifies the number of vertical grid points on the compositing layer registration grid used to measure the distance between the reference grids of the individual codestreams. This field is encoded as a 2-byte unsigned integer.

CDN^i: Codestream number. This field specifies the number of the codestream for this registration value.

XR^i: Horizontal resolution. This field specifies the horizontal distance between points on the reference grid of the codestream specified by the CDN^i parameter, measured in the number of points on the compositing layer registration grid. This field effectively specifies the horizontal scaling needed to match the codestream's reference grid with the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

YR^i: Vertical resolution. This field specifies the vertical distance between points on the reference grid of the codestream specified by the CDN^i parameter, measured in the number of points on the compositing layer registration grid. This field effectively specifies the vertical scaling needed to match the codestream's reference grid with the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

XO^i: Horizontal offset. This field specifies the horizontal distance from centre of the top left point on the reference grid of the codestream specified by the CDN^i parameter to the centre of the top left point on the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

YO^i: Vertical offset. This field specifies the vertical distance from centre of the top left point on the reference grid of the codestream specified by the CDN^i parameter to the centre of the top left point on the compositing layer registration grid. This field is encoded as a 1-byte unsigned integer.

Table M.35 – Format of the contents of the Codestream Registration box

Field name	Size (bits)	Value
XS	16	0-65 535
YS	16	0-65 535
CDN^i	16	0-65 535
XR^i	8	0-255
YR^i	8	0-255
XO^i	8	0-255
YO^i	8	0-255

M.11.8 Contiguous Codestream box

In a JPX file, the Contiguous Codestream box contains an entire codestream as defined by the codestream syntax. However, unlike the JP2 file format, the codestreams contained within a JPX file are not restricted to codestreams defined by Annex A of ITU-T Rec. T.800 | ISO/IEC 15444-1. Codestreams contained within a JPX file may also use extensions to the codestream syntax defined in Annex A of this Recommendation | International Standard.

Contiguous Codestream boxes shall be found only at the top level of the file; they shall not be found within a superbox.

M.11.9 Media Data box

The Media Data box contains fragments of the JPEG 2000 codestream or other media data, such as MPEG-4 audio data. In any case, there shall be other boxes in the file that specify the meaning of the data within the Media Data box. Applications should not access Media Data boxes directly, but instead use the fragment table to determine what parts of which Media Data boxes represent a valid JPEG 2000 codestream or other media stream.

The type of a Media Data box shall be 'mdat' (0x6D64 6174). The contents of a Media Data box in general are not defined by this Recommendation | International Standard.

M.11.10 Composition box (superbox)

The Composition box specifies how the individual composition layers are combined to create the rendered result. It contains a set of global options, followed by a sequence of one or more sets of rendering instructions (each contained within an Instruction Set box). Each individual instruction is associated with a composition layer in the file and defines how that composition layer shall be rendered: its location, scaling, composite operation, etc. A reader that supports composition and animation shall display the file containing the Composition box by executing the sequence of instructions defined within the Composition box. Details on the composition and animation model are specified in M.5.3. A JPX file shall contain zero or one Composition boxes. If present, that box shall be found at the top level of the JPX file; it shall not be found within a superbox.

The type of the Composition box shall be 'comp' (0x636F 6D70) and it shall have the following contents:

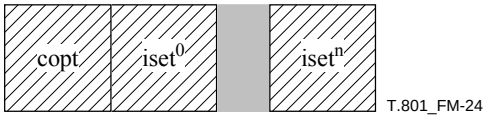

Figure M.24 – Organization of the contents of a Composition box

copt: Composition Options box. This box specifies parameters that apply to the composition or animation as a whole. It is defined in M.11.10.1.

iset^i: Instruction Set box. This box contains a set of instructions for how to combine the multiple composition layers in the file. The entire set of Instruction Set boxes specify the entire composition or animation, and are processed in the order they are found within the Composition box. The Composition Instruction box is defined in M.11.10.2

Table M.36 – Format of the contents of the Composition box

Parameter	Size (bits)	Value
copt	Variable	Variable
iset^i	Variable	Variable

M.11.10.1 Composition Options box

The Composition Options box specifies parameters that apply to the composition or animation as a whole. The Composition Options box shall be the first box in the Composition box and a Composition Options box shall not be found in any other location in the file.

The type of the Composition Options box shall be 'copt' (0x636F 7074) and contents of the box shall have the following format:

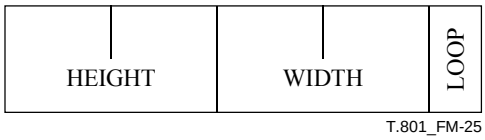

Figure M.25 – Organization of the contents of a Composition Options box

HEIGHT: Rendered result height. This field specifies the height, in samples, of the final rendered result. The resolution of this value is optionally defined in the Default Display Resolution box in the JP2 Header box. This field is encoded as a 4-byte unsigned integer.

WIDTH: Rendered result width. This field specifies the width, in samples, of the final rendered result. The resolution of this value is optionally defined in the Default Display Resolution box in the JP2 Header box. This field is encoded as a 4-byte unsigned integer.

LOOP: Looping count. This field specifies the number of times to fully execute the display instructions. A value of 255 indicates that the reader should repeat the entire set of instructions indefinitely. Prior to each execution of the instruction set, the display area shall be restored to its original state and all instructions' composition layer association reset. Each loop execution should be visually equivalent to redisplaying the composition from scratch. This field is encoded as a 1-byte unsigned integer.

Table M.37 – Format of the contents of the Composition Options box

Parameter	Size (bits)	Value
HEIGHT	32	$1\text{-}2^{32}–1$
WIDTH	32	$1\text{-}2^{32}–1$
LOOP	8	0-255

M.11.10.2 Instruction Set box

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

The type of the Instruction Set box shall be 'inst' (0x696E 7374) and contents of the box shall have the following format:

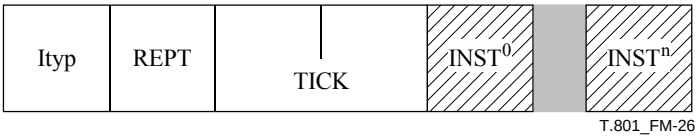

Figure M.26 – Organization of the contents of an Instruction Set box

Ityp: Instruction type. This field specifies the type of this instruction, and thus which instruction parameters shall be found within this Composition Instruction box. This field is encoded as a 16-bit flag. The meaning of each bit in the flag is as follows:

Table M.38 – Ityp field values

Value	Meaning
`0000 0000 0000 0000`	No instructions are present, and thus no instructions are defined for the compositing layers in the file.
`xxxx xxxx xxxx xxx1`	Each instruction contains XO and YO parameters.
`xxxx xxxx xxxx xx1x`	Each instruction contains the WIDTH and HEIGHT parameters.
`xxxx xxxx xxxx x1xx`	Each instruction contains the LIFE, N and PERSIST animation parameters.
`xxxx xxxx xxx1x xxxx`	Each instruction defines the crop parameters XC, YC, WC and HC.
	All other values reserved.

REPT: Repetition. This field specifies the number of times to repeat this particular set of instruction. This field is encoded as a 2-byte big endian unsigned integer. A value of 65 535 indicates to repeat the instruction indefinitely.

TICK: Duration of timer tick. This field specifies the duration of a timer tick (used by the LIFE instruction parameter) in milliseconds. This field is encoded as a 4-byte big endian unsigned integer. If the Ityp field specifies that the LIFE instruction parameter is not used, then this field shall be set to 0, and shall be ignored by readers.

INST^i: Instruction. This field specifies a series of instruction parameters for a single instruction. The format of this field is specified in M.11.10.2.1.

Table M.39 – Format of the contents of the Instruction Set box

Parameter	Size (bits)	Value
Ityp	16	0-65 535
REPT	16	0-65 535
TICK	32	0-(2^{32}–1)
INST^i	Variable	Variable

M.11.10.2.1 Instruction parameter

Figure M.27 shows the contents of each individual INST field (a single compositing instruction) within an Instruction Set box:

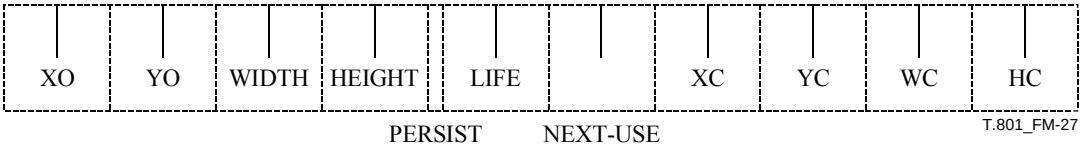

Figure M.27 – Organization of the contents of an INST field within an Instruction Set box

XO: Horizontal offset. This field specifies the horizontal location at which the top left corner of the compositing layer being acted on by this instruction shall be placed in the render area, in samples. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, a default value of zero shall be used.

YO: Vertical offset. This field specifies the vertical location at which the top left corner of the compositing layer being acted on by this instruction shall be placed in the render area, in samples. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, a default value of zero shall be used.

WIDTH: Width of the current compositing layer. This field specifies the width on the render area, in display samples, into which to scale and render the compositing layer being acted on by this instruction. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, the width of the compositing layer shall be used.

HEIGHT: Height of the current compositing layer. This field specifies the height on the render area, in display samples, into which to scale and render the compositing layer being acted on by this instruction. This field is encoded as a 4-byte big endian unsigned integer. If this field is not present, the height of the compositing layer shall be used.

PERSIST: Persistence. This field specifies whether the samples rendered to the display as a result of the execution of the current instruction shall persist on the display background or if the display background shall be reset to the its state before the execution of this instruction, before the execution of the next instruction. This field is encoded as a 1-bit boolean field. A value of 1 indicates true, that the current compositing layer shall persist. If this field is not present, the persistence shall be set to true.

LIFE: Duration of this instruction. This field specifies the number of timer ticks that should ideally occur between completing the execution of the current instruction and completing execution of the next instruction. A value of zero indicates that the current instruction and the next instruction shall be executed within the same display update; this allows a single frame from the animation to be composed of updates to multiple compositing layers. A value of 2^{31}–1 indicates an indefinite delay or pause for user interaction. This field is encoded as a 31-bit big endian unsigned integer. If this field is not present, the life of the instruction shall be set to 0.

NEXT-USE: Number of instructions before reuse. This field specifies the number of instructions that shall be executed before reusing the current compositing layer. This field allows readers to simply optimize their caching strategy. A value of zero implies that the current image shall not be reused for any ensuing instructions, notwithstanding the execution of a global loop as a result of a non-zero value of the LOOP parameter in the Composition Options box. The composition

layer passed on for reuse in this manner must be the original compositing layer, prior to any cropping or scaling indicated by the current instruction. If this field is not present, the number of instructions shall be set to zero, indicating that the current compositing layer shall not be reused. This field is encoded as a 4-byte big endian unsigned integer.

XC: Horizontal crop offset. This field specifies the horizontal distance in samples to the left edge of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the horizontal crop offset shall be set to 0. This field is encoded as a 4-byte big endian unsigned integer.

YC: Vertical crop offset. This field specifies the vertical distance in samples to the top edge of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the vertical crop offset shall be set to 0. This field is encoded as a 4-byte big endian unsigned integer.

WC: Cropped width. This field specifies the horizontal size in samples of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the cropped width shall be set to the width of the current compositing layer. This field is encoded as a 4-byte big endian unsigned integer.

HC: Cropped height. This field specifies the vertical size in samples of the desired portion of the current compositing layer. The desired portion is cropped from the compositing layer and subsequently rendered by the current instruction. If this field is not present, the cropped height shall be set to the height of the current compositing layer. This field is encoded as a 4-byte big endian unsigned integer.

References to Ityp within the individual instruction parameters in Table M.40 refers to the Ityp field within the Instruction Set box that contains this Instruction.

Table M.40 – Format of the contents of the INST^i parameter in the Instruction Set box

Parameter	Size (bits)	Value
XO	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xxx1 Not applicable otherwise
YO	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xxx1 Not applicable otherwise
WIDTH	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xx1x Not applicable otherwise
HEIGHT	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xxxx xx1x Not applicable otherwise
PERSIST	1 0	0, 1; if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
LIFE	31 0	0-(2^{31}–1); if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
NEXT-USE	32	0-(2^{31}–1); if Ityp contains xxxx xxxx xxxx 1xxx Not applicable otherwise
XC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
YC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
WC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise
HC	32 0	0-(2^{32}–1); if Ityp contains xxxx xxxx xx1x xxxx Not applicable otherwise

M.11.11 Association box (superbox)

The Association box allows data in the file to be associated with other data in the file. The Association box is a superbox, containing a sequence of two or more boxes. It creates independent semantic associations between the boxes it contains or the entities represented by those boxes. In particular, associations are created between the first box (or

entities represented by it) (referred to as BF) and each of the other boxes (or represented entities) (referred to as B^i) in the sequence. In the case where there are more than one B^i boxes, it can be thought of as creating semantic clusters around the BF box. There is no explicit association between the B^i boxes.

For example, the association box may be used to associate a label with an entity (image, image set, metadata document, etc.) by placing a Label box in the Association box as BF and the other appropriate boxes as the B^i boxes. It may also be used to associate several items of metadata with the same image or image set by placing a Number List box as BF, followed by the metadata boxes as the B^i boxes. In addition, it may be used recursively to create different levels of association, for example to associate some metadata with a Region of Interest (ROI) and then to associate that ROI and its metadata with an image or image set. These examples are illustrated in Figures M.28 to M.31.

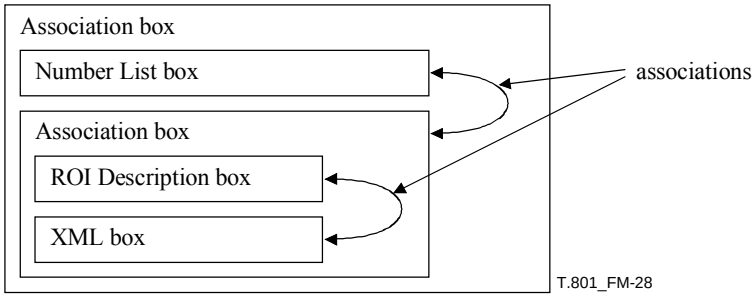

Figure M.28 – Example of ROI specific metadata associated with one or more images

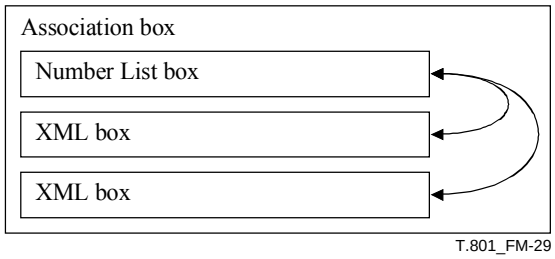

Figure M.29 – Example of Multiple XML documents associated with one or more images

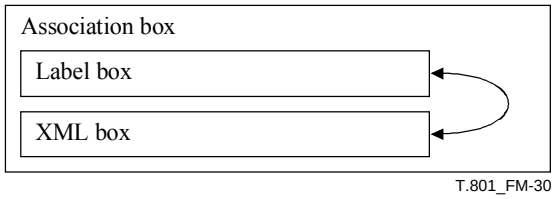

Figure M.30 – Example of a Labelled XML document

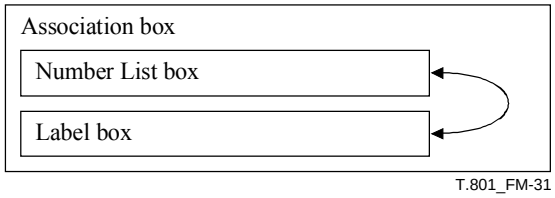

Figure M.31 – Example of a labelled image

The Association box is optional, and there may be multiple Association boxes in the file. An Association box may be found anywhere in the file except before the Reader Requirements box.

The type of an Association box shall be 'asoc' (0x6173 6F63). The contents of the Association box are defined as follows:

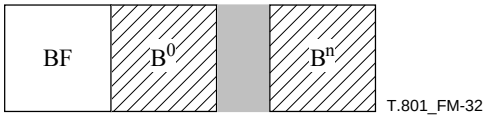

Figure M.32 – Organization of the contents of an Association box

BF: First box. This is the box to which all other boxes within this Association box are associated.

B^i: Box to be associated. This may be any box other than those that are restricted to occurring at particular locations within the file. This box shall be associated with the box BF.

Table M.41 – Format of the contents of the Association box

Parameter	Size (bits)	Value
BF	Variable	Variable
B^i	Variable	Variable

M.11.12 Number List box

The Number List box contains a list of numbers designating entities in the file. Within an Association box, a Number List box stands for the listed entities.

The type of a Number List box shall be 'nlst' (0x6E6C 7374). The contents of the Number List Box shall be as follows:

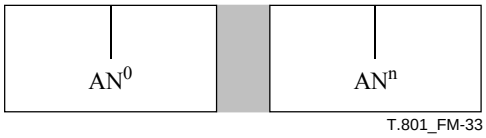

Figure M.33 – Organization of the contents of a Number List box

AN^i: Associate Number. This field specifies the number of an entity with which the data contained within the same Association box is associated. This value is stored as a 4-byte big endian unsigned integer, where the high order byte specifies the type of entity with which the data is associated, and the three low order bytes specify the number of that entity. Legal values of this field are as follows:

Table M.42 – AN^i field values

Value	Meaning
`0x0000 0000`	The rendered result.
`0x01XX XXXX`	The low three order bytes (of value *i*) specify Codestream *i* in the JPX file.
`0x02XX XXXX`	The lower three order bytes (of value *i*) specify Compositing Layer *i* in the JPX file.
	All other values reserved.

Table M.43 – Format of the contents of the Number List box

Parameter	Size (bits)	Value
AN^i	32	$0-(2^{32}-1)$

M.11.13 Label box

The Label box contains a textual label that may be associated with an entity or entities in the file by inclusion of the Label box within an Association box, a Codestream Header box, or a Compositing Layer Header box.

The type of a Label box shall be 'lbl\040' (0x6C62 6C20). The contents of the Label box are as follows:

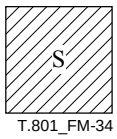

Figure M.34 – Organization of the contents of a Label box

S: Label string. A textual label associated with an entity. This value is stored as ISO/IEC 10646 characters in the UTF-8 encoding. Characters in the ranges U+0000 to U+001F inclusive and U+007F to U+009F inclusive, as well as the specific characters '/', ';', '?', ':' and '#', are not permitted in the label string. Label strings are not null-terminated or padded in any other way; every character that is present is significant.

M.11.14 Binary Filter box

The Binary Filter box allows portions of the file to be further compressed or encoded (i.e., encrypted). For example, if the file contains a significant amount of metadata in XML, it can be losslessly compressed to drastically reduce the file size. This box contains an indicator specifying how the data was transformed, as well as the transformed data. Once the data is transformed through the reverse operation (i.e., decrypted or decompressed), the resulting data shall be a sequence of boxes, where the first byte is the first byte of the first box header, and the last byte is the last byte of the last box. The Binary Filter box is optional, and there may be multiple Binary Filter boxes in the file. A Binary Filter box may be found anywhere in the file except before the Reader Requirements box.

A conforming decoder is not required to process the data within a Binary Filter box. Thus, a Binary Filter box shall not contain boxes for which interpretation is required for reader conformance.

The type of a Binary Filter box shall be 'bfil' (0x6266 696C). The contents of the Binary Filter box are defined as follows:

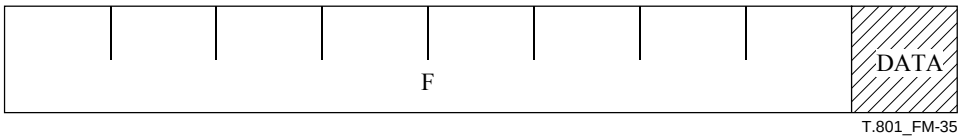

Figure M.35 – Organization of the contents of a Binary Filter box

F: Filter type. This field specifies how the data was transformed before storage. This value is encoded as a UUID. Standard defined values are:

Table M.44 – Legal Filter types

Value	Meaning
EC340B04-74C5-11D4-A729-879EA3548F0E	**Compressed with GZIP.** The contents of the DATA field have been compressed using the DEFLATE algorithm (as specified in RFC 1951). The compressed data is stored in the binary structure defined by the GZIP file format, as specified in RFC 1952.
EC340B04-74C5-11D4-A729-879EA3548F0F	**Encrypted using DES.** The contents of the DATA field has been encrypted using DES as defined in ISO 10126-2.
	All other values reserved.

If a conforming reader does not recognize the particular UUID, then the reader shall ignore this Binary Filter box.

DATA: Transformed data. This field contains previously transformed data. Once the reverse transformation has been applied (as specified by F), the result shall be a sequence of boxes. The contents of the data field may include information needed to perform the reverse filter, in addition to the filtered data. It is fully up to the definition of the F field to define the binary structure and format of the DATA field.

Table M.45 – Format of the contents of the Binary Filter box

Parameter	Size (bits)	Value
F	128	Variable
DATA	Variable	Variable

M.11.15 Desired Reproductions box (superbox)

The Desired Reproductions box specifies a set of transformations that must be applied to the image to guarantee a specific desired reproduction on a set of different output devices, respectively. For example, consider an image that contains real-world blue colours. This image is intended to be printed in a catalogue, and thus the printed image must match the actual colour of the original physical object when seen by a human viewer. However, the CMYK printing process does not reproduce the same range of blue colours as are viewable by the human visual system. In this instance, the catalog artist must determine how to best convert the blue colour in the image to a printed blue colour to minimize differences between the physical object from the printed reproduction.

A JPX reader is not required to process the image through the specified transformations.

This box contains a set of separate desired reproductions. There shall be only one Desired Reproductions box within the file, which may be found anywhere within the file.

The type of the Desired Reproductions box is 'drep' (0x6472 6570). This box is a superbox, and the contents of the box shall be as follows:

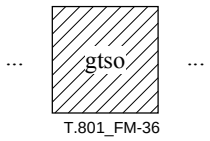

Figure M.36 – Organization of the contents of the Desired Reproductions box

gtso: Graphics Technology Standard Output box. This box specifies the desired output colour and tone reproduction for the rendered result when printed under commercial printing conditions. The format and definition of this box is specified in M.11.15.1

Other boxes may be found within the Desired Reproductions box. Readers shall ignore any boxes that they do not understand.

The Desired Reproduction box is optional for conforming files.

M.11.15.1 Graphics Technology Standard Output box

A Graphics Technology Standard Output box specifies the desired reproduction of the rendered result for commercial printing and proofing systems. The box contains an Output ICC profile specifying the desired conversion of the image from the Profile Connection Space (PCS) to the desired device specific output colourspace. There shall be only zero or one Graphics Technology Standard Output box within the file. If present, this box shall be found within the Desired Reproductions box.

The type of a Graphics Technology Standard Output box is 'gtso' (0x6774 736F). The contents of the box shall be as follows:

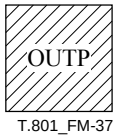

Figure M.37 – Organization of the contents of the Graphics Technology Standard Output box

OUTP: This field shall be a valid Output ICC profile as defined by the ICC Profile format specification ICC-1. Version information is embedded within the profile itself. Applications that only support specific versions of the ICC Profile Format Specifications can extract the version number from bytes 8-11 of the profile (bytes 8–11 of the contents of the Output ICC Profile box).

Table M.46 – Format of the contents of the Graphics Technology Standard Output box

Parameter	Size (bits)	Value
OUTP	Variable	Variable

M.11.16 ROI Description box

An ROI Description box contains information about parts of an image that might be useful in certain applications such as random access. The ROI description box can also be used together with the association box to associate metadata to parts of the image. The ROIs described in this box are not necessarily coded as ROIs within the codestream; it also allows an application or user to signal the importance of certain parts of an image even if theses parts are not emphasized by the RGN or ARN marker segment in the codestream. There can be multiple ROI Description boxes within the file. However, a ROI Description box shall be found only at the top level of the file, or within the JP2 Header box, a Codestream Header box or an Association box. If the ROI Description box is found within a Codestream Header box, then the ROIs described in that ROI Description box pertain to the particular codestream described by that Codestream Header box. If the ROI Description box is found within the JP2 Header box, then the ROI description box specifies default ROI information for all codestreams. If the ROI Description box is found at the top level of the file, then it specifies ROI information for the rendered result; the ROIs described at a top-level box are not directly associated with coded ROIs within any codestream.

The type of the ROI Description box shall be 'roid' (0x726F 6964). The contents of this box shall be as follows:

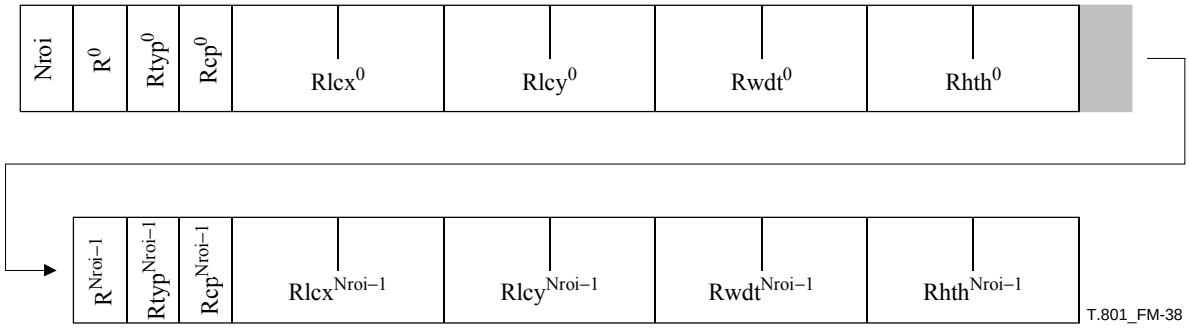

Figure M.38 – Organization of the contents of the ROI Description box

Nroi: Number of Regions of Interest. Encoded as an 8-bit integer.

R^i: Region of Interest present in codestream. Encoded as an 8-bit integer. Legal values of the R^i field are as follows:

Table M.47 – Legal R^i values

Value	Meaning
0	Codestream does not contain a static region of interest at this location.
1	Codestream contains a static region of interest at this location.
	All other values reserved.

$Rtyp^i$: Region of Interest type, can be either rectangular or ellipse. Encoded as an 8-bit integer. Legal values of the $Rtyp^i$ field are as follows:

Table M.48 – Legal $Rtyp^i$ values

Value	Meaning
0	Rectangular region of interest.
1	Elliptical region of interest.
	All other values reserved.

Rcp^i: Region of Interest coding priority. This value describes the coding priority of the Region of Interest. The value 0 means low coding priority and 255 means maximum coding priority. This value is encoded as a 1-byte unsigned integer. In transcoding applications, bits should be allocated with respect to the coding priority of each ROI.

$Rlcx^i$: Region of Interest horizontal location. In the case of rectangular area this is the location of the top left corner of the rectangle. In the case of an elliptic Region of Interest, this is the horizontal position of the centre point. This value is stored as a 4-byte big endian unsigned integer.

$Rlcy^i$: Region of Interest vertical location. In the case of rectangular area, this is the location of the top left corner of the rectangle. In the case of an elliptic Region of Interest, this is the vertical position of the centre point. This value is stored as a 4-byte big endian unsigned integer.

$Rwdt^i$: Region of Interest width. In the case of rectangular Region of Interest, this is the width of the rectangle. In the case of an elliptic Region of Interest, this is the horizontal axis. This value is stored as a 4-byte big endian unsigned integer.

$Rhth^i$: Region of Interest height. In the case of rectangular Region of Interest, this is the height of the rectangle. In the case of an elliptic Region of Interest, this is the vertical axis. This value is stored as a 4-byte big endian unsigned integer.

Table M.49 – Format of the contents of the ROI Description box

Parameter	Size (bits)	Value
Nroi	8	0-255
R^i	8	0-255
$Rtyp^i$	8	0-255
$Rlcx^i$	32	$1-(2^{32}-1)$
$Rlcy^i$	32	$1-(2^{32}-1)$
$Rwdt^i$	32	$1-(2^{32}-1)$
$Rhth^i$	32	$1-(2^{32}-1)$

M.11.17 Digital Signature box

This box contains a checksum or digital signature that may be used to verify data contained within the file. This digital signature is used to protect a very specific byte stream within the file. Any change to that byte stream will invalidate the digital signature. For example, if a compressed codestream is signed, and then later modified by adding error resilience markers, the digital signature will indicate that the byte stream has been modified.

The type of a Digital Signature box shall be 'chck' (0x6368 636B). The Digital Signature box is optional and may occur anywhere in the file except before the Reader Requirements box. There may be more than one Digital Signature box in the file. The contents of a Digital Signature box shall be as follows:

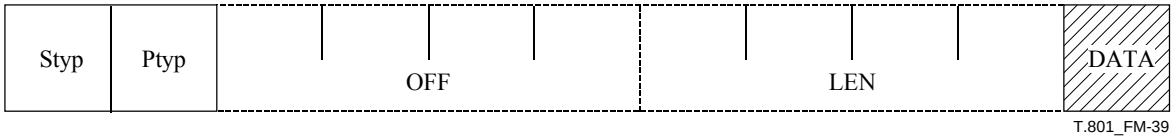

Figure M.39 – Organization of the contents of a Digital Signature box

Styp: Signature type. This field specifies the type of digital signature contained within this Digital Signature box. This field is encoded as a 1-byte unsigned integer. Legal values of the Styp field are as follows:.

Table M.50 – Legal Styp values

Value	Meaning
0	A checksum, generated by applying the MD5 algorithm to the source data, is stored in the DATA field of this Digital Signature box as a sequence of bytes in the order specified by RFC 1321.
1	The checksum is generated by applying the SHA-1 algorithm, as defined in ANSI X9.30.2, to the source data. The resulting signature is stored in the DATA field of this Digital Signature box as a 20-byte big endian unsigned integer.
2	The digital signature is generated by applying the DSA algorithm, as defined in FIPS 186-2, to the source data. The signature consists of two unsigned integers, r and s. The DATA field shall be 40 bytes long. The first 20 bytes shall be the value r, encoded as a 20-byte big endian unsigned integer. The second 20 bytes shall be the value s, encoded as a 20-byte big endian unsigned integer.
3	A digital signature, generated by applying the RSA signature algorithm with the MD5 message digest algorithm (according to PKCS #1 Version 1.5) to the source data, is stored in the DATA field of this Digital Signature box.
4	A digital signature, generated by applying the RSA signature algorithm with the SHA-1 message digest algorithm (according to PKCS #1 Version 1.5) to the source data, is stored in the DATA field of this Digital Signature box.
5	A ContentInfo value of the Cryptographic Message Syntax is stored in the DATA field of this Digital Signature box. Its 'content' field shall contain either a DigestedData value or a SignedData value, and in either case shall use the 'external signatures' mechanism described in section 5.2 of RFC 2630 to apply the chosen digest or signature algorithm to the source data.
	All other values reserved.

If key management is not an issue for a particular application (for example, if a checksum is being sent, or if the recipient already knows the public key with which to verify a signature), and if the CMS method (Styp = 5) is being used, it may help simple readers to include in the file an additional Digital Signature box using one of the other methods (Styp < 5) on the same source data. Readers without CMS support would still be able to process the additional box.

Determination of any required public key is outside the scope of this Recommendation | International Standard.

Ptyp: Source pointer type. This field indicates how the source data range that is signed by this Digital Signature box is specified. This field is encoded as a 1-byte unsigned integer. Legal values of the Ptyp field are as follows:

Table M.51 – Legal Ptyp values

Value	Meaning
0	The source data that is signed by this Digital Signature box shall be all bytes of the file, starting with the first byte, up to the byte immediately preceding the box header for this Digital Signature box. If the source data is specified using a Ptyp of 0, then this Digital Signature box shall not be in any superbox in the file; it must be at the top level of the file.
1	The source data that is signed by this Digital Signature box shall be a range bytes, starting with the byte at the location specified by the OFF field. The length of this range is specified by the LEN field. If the source data is specified using a Ptyp of 1, then OFF shall point to the start of a box header and the source range shall include only complete boxes; the Digital Signature box shall be at the same level in the box hierarchy as the box pointed to by the OFF field.
	All other values reserved.

OFF: Source data offset. This field specifies the offset in bytes to the start of the source data range that is signed by this Digital Signature box. This offset is relative to the first byte of the file. This field is encoded as an 8-byte big endian unsigned integer. If the value of Ptyp is 1, then this field shall not exist.

LEN: Source data length. If non-zero, this field specifies the length in bytes of the source data range that is signed by this Digital Signature box. A value of zero specifies that the end of the source data range is the last byte of the file. This field is encoded as an 8-byte big endian unsigned integer.

DATA: Signature data. This field contains the digital signature produced from the source data range. The format of this data is as specified by the Styp field.

Table M.52 – Format of the contents of the Digital Signature box

Parameter	Size (bits)	Value
Styp	1	0
Ptyp	1	0-1
OFF	64 0	0-(2^{64}–1); if Ptyp = 1 not applicable; if Ptyp = 0
LEN	64 0	0-(2^{64}–1); if Ptyp = 1 not applicable; if Ptyp = 0
DATA	variable	variable

M.11.18 XML box

The JP2 file format defines the XML box to contain a well-formed XML document as defined by XML 1.0. In a JPX file, this box is extended to allow JPX readers to better make use of the XML data. The format of the XML box is unchanged. However, a JPX reader should take the following actions to parse the XML document:

- If the reader finds the "xsi:schemaLocation" attribute in the root element, then the structure of this XML document or instance data is defined by a schema. This attribute specifies the physical location of the schema document.
- If the reader finds the "!DOCTYPE" line in the header of the XML document, then the structure of the XML document or instance data is defined by a Document Type Definition (DTD) document. This line specifies which DTD is used by this XML document or instance data, as well as the root element name and the location of the DTD.
- If the XML schema or DTD document referred to by the XML document contained within the XML box contains a comment field of the form "<!--HUMAN_SCHEMA_DTD_LOCATION: *LOC* -->", then a reader may retrieve a human readable document from the URL specified by *LOC*. This document shall contain a human readable description of the schema or DTD. This document will support application developers and users of the metadata.

M.11.19 MPEG-7 Binary box

This box contains metadata in MPEG-7 binary format (BiM) as defined by ISO/IEC 15938.

The type of an MPEG-7 Binary box shall be 'mp7b' (0x6D70 3762). It may be found anywhere in the file after the Reader Requirements Box. The contents of the MPEG-7 Binary box are as follows:

Figure M.40 – Organization of the contents of a MPEG-7 Binary box

DATA: MPEG-7 BiM Stream.

M.11.20 Free box

The Free box specifies a section of the file that is not currently used and may be overwritten when editing the file. Readers shall ignore all Free boxes. A Free box may be found anywhere in the file except before the Reader Requirements box.

The type of a Free box shall be 'free' (0x6672 6565). As a free box contains meaningless data, the contents of a Free box are undefined.

M.12 Dealing with unknown boxes

A conforming JPX file may contain boxes not known to applications based solely on this Recommendation | International Standard. If a conforming reader finds a box that it does not understand, it shall skip and ignore that box.

M.13 Using the JPX file format in conjunction with other multi-media standards (informative)

While the JPX file format provides a powerful architecture for storing still-images, there are many applications in which still-images are stored in conjunction with other multi-media types. For example, many digital still cameras allow the user to capture an audio annotation to describe a particular photograph.

This integration with other multimedia types is facilitated by the use of the Box structure for encapsulating data within the JP2 and JPX file format. The box structure itself has the same binary definition as a QuickTime atom or an MPEG-4 atom. As such, a file can be created using both JPX boxes and Quicktime or MPEG-4 atoms. Provided all offsets within the file are correct with respect to the data location from the beginning of the file (and take into account the presence of all boxes and atoms), a dual-mode file can be created.

For example, it is very easy to create a file that contains both a still photograph and an audio annotation. The boxes required to store the still photograph file can be combined with the atoms required to store a MPEG-4 audio file into a single file, as the MPEG-4, JPX and JP2 formats are flexible with respect to the location of many important boxes or atoms. A file writer would only need to be concerned that the offsets within the boxes and atoms are all determined such that they point to the location of the data in the combined file.

A reader that supports only the JPX file format would treat the file as a photograph. A reader that supports only the MPEG-4 audio standard would treat the file as an audio file. A new reader that supports both standards could then provide advanced features by combining photographic capabilities with audio capabilities.