ISO/IEC JTC 1/SC 29/WG1 N2401

Date: 16 November 2001

TITLE:	**JPEG 2000 Part 6 FCD 15444-6**
SOURCE:	Robert Buckley, Xerox Corporation, Part 6 Editor Louis Sharpe, Picture Elements, Part 6 Co-editor Simon McPartlin, LuraTech GmbH, Part 6 Co-editor Shigetaka Ogawa, NEC Corporation
PROJECT:	**JPEG 2000 Part 6**
STATUS:	
REQUESTED ACTION:	
DISTRIBUTION:	WG1 website and distribute to WG1

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

INFORMATION TECHNOLOGY

JPEG 2000 IMAGE CODING SYSTEM: COMPOUND IMAGE FILE FORMAT

JPEG 2000 PART VI FCD, 16 NOVEMBER 2001

Foreword

Forward to be supplied by ISO and ITU.

Introduction

Introduction to be supplied by ISO and ITU.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – JPEG 2000 IMAGE CODING SYSTEM PART VI: COMPOUND IMAGE FILE FORMAT

1 Scope

This Recommendation | International Standard defines a normative but optional file format for storing compound images using the JPEG 2000 file format family architecture. This format is an extension of the JP2 file format defined in Part 1 Annex I of this Recommendation | International Standard and uses boxes defined for both the JP2 file format and the JPX file format in Part 2 Annex M of this Recommendation | International Standard. While not all applications will use this format, many applications will find that it meets their needs. Applications that implement this file format shall implement it as described in this Recommendation | International Standard.

This Recommendation | International Standard

— specifies a binary container for multiple bi-level and continuous-tone images used to represent a compound image.

— specifies a mechanism by which multiple images can be combined into a single compound image, based on the Mixed Raster Content model.

— specifies a mechanism for grouping multiple images in a hierarchy of layout objects, pages and page collections.

— allows image compression methods other than JPEG 2000.

— specifies a mechanism by which metadata can be included in files specified by this Recommendation | International Standard.

2 References

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

2.1 Identical Recommendations | International Standards

— ITU-T Recommendation T.4, Standardization of Group 3 Facsimile Terminals for Document Transmission

— ITU-T Recommendation T.6, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus

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

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

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

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

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

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

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

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

— ITU-T Recommendation T.89, Application Profiles for Recommendation T.88 (JBIG2).

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

— ITU-T Recommendation T.800 | ISO/IEC 15444-2, JPEG 2000 Image Coding System - Extensions.

— ITU-T Recommendation T.42, Continuous tone colour representation method for facsimile.

— ITU-T Recommendation T.44 | ISO/IEC 16485, Mixed Raster Content.

— ITU-T Recommendation T.44, Mixed Raster Content - Amendment 1: Accommodation of New Annex B.

— ITU-T Recommendation T.45, Run-length Colour Encoding.

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

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

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

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

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

— W3C, Extensible Markup Language (XML 1.0), 2nd Edition, Rec-xml-20000106, <http://www.w3.org/TR/1999/REC-xml>.

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

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

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

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

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

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

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

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

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

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

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. The definitions defined in ITU-T T.800 | IS 15444-1 Section 3 also apply to this Recommendation | International Standard.

3.1 **base color**: The color of an object for which no image data is available.

3.2 **box**: A portion of the file format defined by a length and a unique box type. Boxes of some types may contain other boxes.

3.3 **component**: A two-dimensional array of samples.

3.4 **compound image**: An image that may contain scanned images, synthetic images or both and that preferably requires a mix of continuous tone and bi-level compression methods.

3.5 **file format**: A codestream or codestreams and additional support and information not explicitly required for decoding of the codestream or codestreams. Examples of such support data include text fields providing security and historical information, data to support the placement of multiple codestreams within a given data file, and data to support exchange between platforms or conversions to other file formats.

3.6 **fragment**: A portion of the codestream for an image. Sections 5.2.6 describes fragment usage.

3.7 **JP2 file**: The name of a file in the file format described in Part 1 of this Recommendation | International Standard. Structurally, a JP2 file is a contiguous sequence of boxes.

3.8 **JPM file**: The name of a file in the file format described in this Part of this Recommendation | International Standard. A JPM file can contain one or more pages, composed from one or more layout objects, each of which is composed from at most two objects. Structurally, a JPM file is a contiguous sequence of boxes.

3.9 **JPX file:** The name of a file in the file format described in Part 2 of this Recommendation | International Standard. Structurally, a JPX file is a contiguous sequence of boxes.

3.10 **layout object**: An entity that comprises at most two paired objects or MRC layers.

3.11 **mask object:** An object that is used to select the samples of a corresponding image object that are to be imaged on a page.

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

3.13 **MRC**: Mixed Raster Content; a multi-layer imaging model described in ITU-T Recommendation T.44 | ISO/IEC 16485.

3.14 **object:** An image that is part of a layout object; an MRC layer.

3.15 **page**: The largest collection of layout objects that can be imaged independently of any other layout objects; a canvas or frame for imaging.

3.16 **profile**: A subset of all possible field values in a file.

3.17 **superbox**: A box that itself contains a contiguous sequence of boxes (and only a contiguous sequence of boxes).

4 Abbreviations

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

IPR:	Intellectual Property Rights
UUID:	Universal Unique Identifier
CCITT:	International Telegraph and Telephone Consultative Committee, now ITU-T
DPI:	Dots per inch.
JP2:	**JP**EG **2**000 File Format defined in Part 1 of this Recommendation | International Standard
JPX:	JPEG 2000 File Format defined in Part 2 of this Recommendation | International Standard; **JP**EG 2000 File Format E**X**tended
JPM:	JPEG 2000 File Format defined in Part 6 (this part) of this Recommendation | International Standard; **JP**EG 2000 File Format - **M**ulti-layer
MRC:	Mixed Raster Content

5 General description

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

This Recommendation | International Standard defines a file format for storing compound images using the JPEG 2000 file format family architecture. A compound image is an image that may contain scanned images, synthetic images or both. This Recommendation | International Standard is based on the multi-layer Mixed Raster Content (MRC) imaging model, defined in ITU-T T.44 | ISO 16485, to represent a compound image. Therefore, the file described in this Recommendation | International Standard can contain multiple continuous tone and bi-level images, using a composition model to combine them and render a single image. While the file format supports several ISO and ITU-T compression methods for continuous tone and bi-level images, JPEG 2000 is the preferred method for continuous tone image compression in this file.

This part specifies the member of the JPEG 2000 File Format family that enables efficient processing, interchange and archiving of raster-oriented pages containing a mixture of multi-level and bi-level images. This efficiency is realized by representing the mixed-content image as multiple layers (planes), as determined by image type, and applying image specific encoding, spatial and colour resolution processing. A rasterized page may contain one or more image types, such as: multi-level continuous-tone or palettized (contone) usually associated with naturally occurring images; bi-level detail associated with text and line-art; multi-level colours associated with the text and line-art. This Part makes provisions for processing, interchange, and archiving of these image types in multiple separate layers. Recombining the layers in a prescribed manner regenerates the desired image.

5.1 Mixed Raster Content Model

A file that conforms with this Recommendation | International Standard contains one or more “pages,” grouped into “page collections.” A page consists of an optional base colour object and one or more layout objects, which when imaged in the specified sequence, will give the desired page image. A layout object consists of a mask object M and an image object I. A mask object is an opacity image and has only one component; an image object can be grayscale or colour with one or more components.

To apply a layout object to the page when building up the desired page image, the image object I_k is applied to the page through the mask object M_k associated with it in the layout object. The initial page, the BasePage, can be transparent, white, black or have the colour indicated in the page base colour object, if it exists. The following equation shows the model for combining the BasePage and a sequence of n layout objects.

$$PageImage_0 = BasePage \tag{5.1}$$

$$PageImage_k = (1 - M_k) \times PageImage_{k-1} + M_k \times I_k \tag{5.2}$$

$$PageImage = PageImage_n \tag{5.3}$$

where I_k is the scaled image in the layout object k. If the layout object k does not contain a mask object then M_k is 1, otherwise M_k is the scaled mask image normalized so that the maximum possible value, $2^{\text{mask bit depth}}$-1, is 1. Full details of layout object compositing may be found in Annex 5.2.4.

The mask object is commonly binary, as in ITU-T Rec. T.44 | ISO 16485. A binary mask is a special case of a multi-bit mask, which allows blending the image object on to the page built up so far. Mask objects and image objects are instances of plane objects. The mask and image objects of a layout object can have different resolutions, bit depths, compression types and image sizes.

The image objects being combined to generate the PageImage may use any of the supported colourspaces. When using non-binary masks, the image objects should be converted to a common colourspace and bit depth before being combined to generate the PageImages. The recommended colourspace for combining the image objects is a linear colourspace such as the linear version of sRGB (see IEC 61966-2) or XYZ (see T.42). An alternative, but sub-optimal, choice of common colourspace would be the colourspace in which the final PageImage shall be rendered or displayed.

When combining image objects with a PageImage, Equations 5.1-5.3 are applied individually to each of the channels. For example, when combining in an sRGB colourspace, the R, G and B channels of the image object are individually combined with the R, G and B channels of the PageImage using Equations 5.1-5.3.

5.2 File elements and structure

The files that conform to the format defined in this Recommendation | International Standard are called JPM files. At its core, a JPM file is a sequence of pages, where each page in turn is a sequence of layout objects. A layout object normally consists of a mask object and an image object. Mask and image objects are composited to build up the final page image according to the equation in Section 5.1. The key elements or boxes of a JPM file are: page, layout object, image object and mask object. An object point to its image data (codestream fragments) indirectly via a fragment table. Like all members of the JPEG 2000 file format family, a JPM file begins with a JPEG 2000 Signature Box and a File Type Box. They are followed by a Compound Image Header Box, which contains general information about the file. There is then a Page Collection Box, which can be used to navigate the sequence of pages that then follow. The hierarchical organization of these elements in a JPM file is shown below.

```
JPEG 2000 Signature
File Type
Optional JP2 Header for Document Thumbnail
Compound Image Header
Page Collection
Page
    Layout Object
        Mask Object
        Image Object
Fragment Table
Codestream Fragment
```

Fragment Table
Codestream Fragment

A JPM file can have other, optional elements, not shown here but described in the Parts and Annexes of this Recommendation | International Standard. Each element of a JPM file is represented as a box and a JPM file consists of a sequence of boxes, some of which can contain other boxes. Annex A.3 gives the definition of a box.

5.2.1 Page Collections

A JPM compatible file consists of a sequence of pages, each of which occurs at the top level of the file and each of which can be imaged independently of any other page. Pages optionally may be grouped into collections. Page Collection boxes can be nested, so that a page collection can itself consist of one or more page collections and/or one or more pages. A Page Collection box contains an optional Label box, one or more metadata (XML and/or UUID) boxes for the page collection, and a Page Table box that contains the locations of the pages and of subsidiary page collections belonging to the collection.

A JPM file may or may not have a page collection. If it does have a page collection, it may be located anywhere at the file level. It is recommended that optimized files have at least one page collection and that it be placed near the beginning of the file as specified in Section B. While pages and page collections typically occur at the top level of the file, they may be located in external files. All such pages which are part of a JPM file must be referenced from a page collection, which at this point is mandatory.

The purpose of a primary page collection is to enable an application to guide navigation through the primary document to which the page belongs using the sequential order specified by the primary page collection, thus providing support for previous page and next page commands.

Every Page Collection box contains a Page Table box. The entries point to the locations of the Page and Page Collection boxes within the page collection. A flag for each entriy specifies whether the location is that of a page or page collection as well as indicating whether the box contains a thumbnail or metadata.

By walking the tree of pages and "nested" page collections in a page's primary page collection, all pages can be reached. Every page has a PageID which is unique within the file where the primary page collection for the page is located. The data reference for the page's primary page collection, when combined with the PageID, forms a unique number within the current JPM file. This unique number may be used to find the current page within the page collections of the current JPM file, and then the next page and previous page can be found by walking one page forwards or backwards from the current page. The PageID alone can be used to find the current page within its primary page collection, where it is guaranteed to be unique.

Multiple page collections can exist in a JPM file. Some may have functions other than basic navigation. A table of figures could point to those pages containing figures. A section table or chapter table might contain subsets of the entire set of pages and be referenced in other page collections. A search results page collection could occur in a JPM file that has only snippets of pages where search hits on metadata occurred.

Two examples of the use of page collections follow:

Search results example

A search on metadata in a JPM file on a server might result in the server returning a search results JPM file related to the primary file. The primary page collection for any given page in the search results page may not actually be located directly in the file, but could be reached using the primary page collection data reference in the page's Page box. The search results page collection would include only pages having search hits on them. If the user then wanted to look at the next page, one following a search hit page, but which does not occur in the search results page collection, the client software would go to the data reference to the primary page collection found in that page's Page box. This entry (if present) has a pointer to the primary Page Collection box back in the main JPM file. This primary Page Collection box will then be used by the client software to walk the tree, comparing PageIDs until the current page is found. By that means, the previous and next pages can be found. Full details of these boxes may be found in Annex B.

Encyclopedia example

The primary page collection of a very large multi-volume set of books such as an encyclopedia would not want to point to each page individually; such a data structure would be too large to be brought over to a client before even the first page can be displayed. Instead, the primary page collection might serve as a table of contents and point to each of the front matter pages individually, then to page collections that each cover entire volumes: Volume A, Volume B and so on. Each of these "volume" page collections would reference "section" page collections that would reference "article" page collections.

A search of metadata for a term occurring on a few of the pages would return a new "search results" JPM file containing a single "search results" page collection and a handful of pages. The first entry in this page collection box would point directly to the first search result page-actually one of the pages in the encyclopedia via an external reference. By navigating the "search results" page collection, the application can let the user go to next search hit and previous search hit. If the user then finds the article of interest and wants to continue reading onto a page back in the remote encyclopedia where no search hit occurred, they would then need a next page function as well. The application then retrieves the primary page collection for that page and begins a search of that tree for the PageID of the current page. It can then provide to the user the very next page in the encyclopedia.

5.2.2 Pages

A page is contained in a Page box, which is a superbox that consists of a Page Header box, containing general information about the page, a Resolution box, which gives the resolution of the page, an optional Base Colour box, which describes the page colour, optional Metadata boxes, and Layout Object boxes, one for each layout object on the page. Together, these boxes define the “canvas” on which the page’s layout objects are imaged. Full details of these boxes may be found in Annex B.2.

5.2.3 Layout Objects

A Layout Object box contains a Layout Object Header box with general information about the layout object, optional boxes with metadata that applies to the layout object, and one or two Object boxes - an image Object box and/or mask Object box, or a combined image/mask Object box. The Layout Object Header box defines a rectangular region on the page, as illustrated in Figure 5-1.

Each Object box contains an Object Header box, which identifies whether thisas an image, a mask or a combined image-mask object and gives the position of either a Contiguous Codestream box or a Fragment Table box which is used to locate the codestream data for the object.

When the codestream for the object is defined then the Object Header box is followed by an optional Object Scale box and a JP2 Header box that contains boxes describing the object data: an Image Header, Colour Specification, optional Bits Per Component, Palette, Component Mapping, Channel Definition and Resolution boxes. Either these boxes, or a reference to them, are placed in the Object box. These are the same boxes that would occur in a JP2 or JPX file. For example, an image object can be a JP2 file, suitably referenced from within a JPM file. The Object Scale box describes the scaling which must be applied to the object before applying it to the page. The vertical scaling is by the ratio $\frac{VRN}{VRD}$, and the horizontal scaling is by the ratio $\frac{HRN}{HRD}$. If an Object box does not contain an Object Scale box then the vertical and horizontal scaling ratios are both 1, i.e. no scaling. The Object Header box contains offset fields, OHoff and OVoff, which describe a position in the scaled object. This position is lined up with the top-left corner of the layout object region and clipped against this region. If a base colour box exists in the object box then this colour is used in any areas of the layout object region where the scaled object is not defined. Figure 5-2 illustrates the scaling and offset position.

The scaling algorithm used will affect the appearance of the re-composed image. Bitonal mask images should be scaled to a grey image using bi-linear interpolation for proper results. The scaling algorithm used on image objects is left to the implementor, but will also affect the image quailty, if perhaps to a lesser degree.

Within a Page Box, there are as many Layout Object boxes as there are layout objects on the page. Full details of these boxes may be found in Annex B.3.

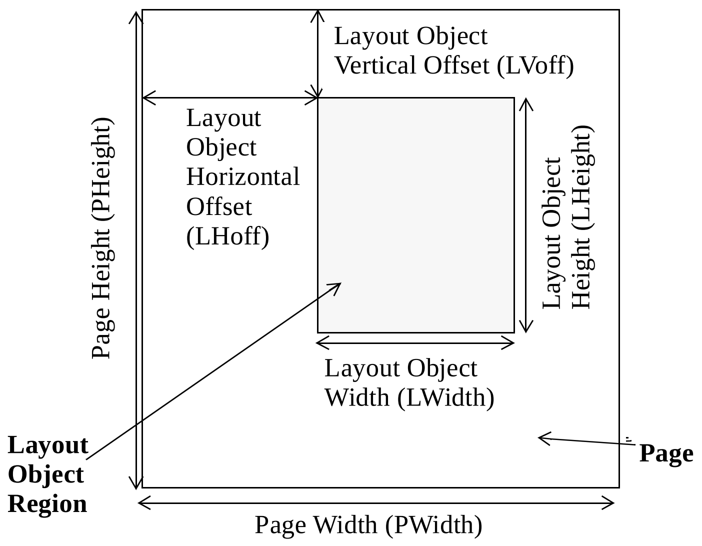

Layout Object Vertical Offset (LVoff)
Layout Object Horizontal Offset (LHoff)
Page Height (PHeight)
Layout Object Height (LHeight)
Layout Object Width (LWidth)
Layout Object Region
Page
Page Width (PWidth)

Figure 5-1 — Layout Object Region

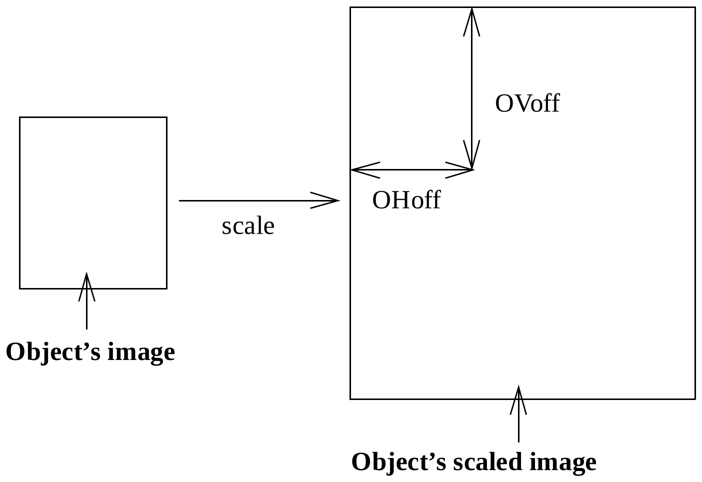


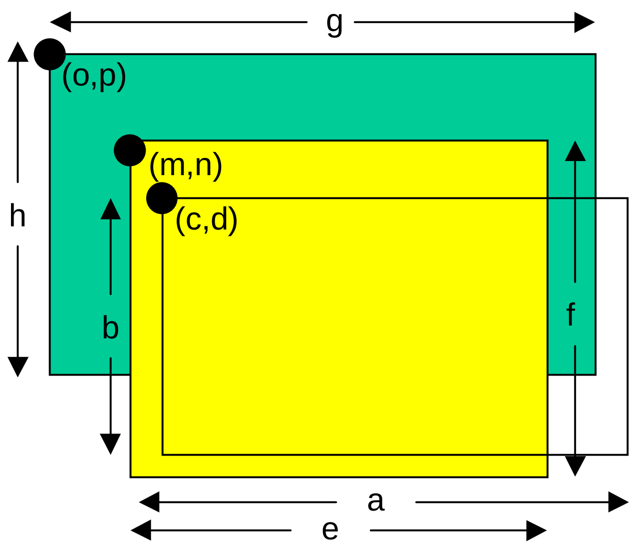
Figure 5-2 — Object Scaling and Positioning

5.2.4 Layout Object Compositing

A layout object typically comprises an image object and a mask object. They can occur as separate objects, each with its own codestream, or as a single object with both the mask and image data in the same codestream. The latter would occur in cases where there is an image with an opacity channel or a JBIG2 codestream with color tags. It is also possible to have an image object only or a mask object only. An image object typically has a codestream associated with it, but may not, in which case the NoCodestream field in the image Object header is set to 1. An image object may also have an associated constant colour or image base colour. If an image object does not have an associated codestream (NoCodestream=1), then it must have a defined image base color. A mask object, if present, must have a codestream associated with it and have NoCodestream=0. The legal combinations of image and mask objects are given in the following table.

Case	Mask	Image	Result	Style
1	Present: NoCodestream=0	Absent	Mask object only. Use black as the image base color; apply it through the mask	3
2	Absent	Present: NoCodestream=0	Image object only. Image object is self masking	2
3	Absent	Present: NoCodestream=1	Image object only; no codestream. Image Base coor must be defined; fill layout object with Image Base Color	2
4	Present: NoCodestream=0	Present: NoCodestream=0	Both Mask and Image objects present; possibly in the same codestream (Style=1). Apply image object data through Mask: use image data where Mask and Image extents overlap; use Image Base Color outside Image object extent but within Mask extent	0, 1
5	Present: NoCodestream=0	Present: NoCodestream=1	Both Mask and Image objects present, but image object contains Image Base Color only; apply Image Base Color through Mask	0

The following figure shows the general arrangement of mask and image objects in Case 4 in the table.

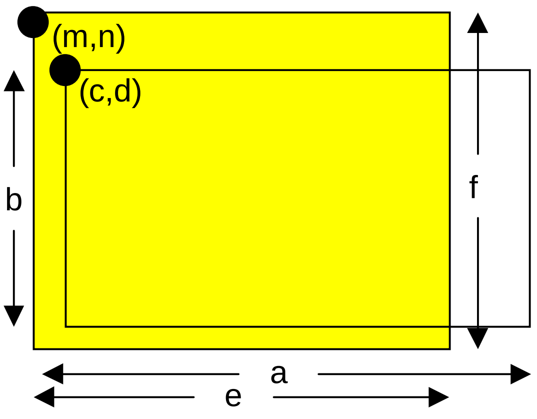


The layout object has dimensions a and b and is located at (c, d) with respect to the page. The scaled mask object has dimensions e and f, and the scaled image object has dimensions g and h. Both have offsets that position them as shown in the figure with respect to the layout object. Within the boundaries of the layout object, there are 4 regions: mask and image overlap, mask only, image only, and no image or mask. The page image is only affected in the first two regions where there is mask data. Where there is image data overlapping the mask data, then the result is described as follows:

$$PageImage_k = (1 - M_k) \times PageImage_{k-1} + M_k \times I_k$$

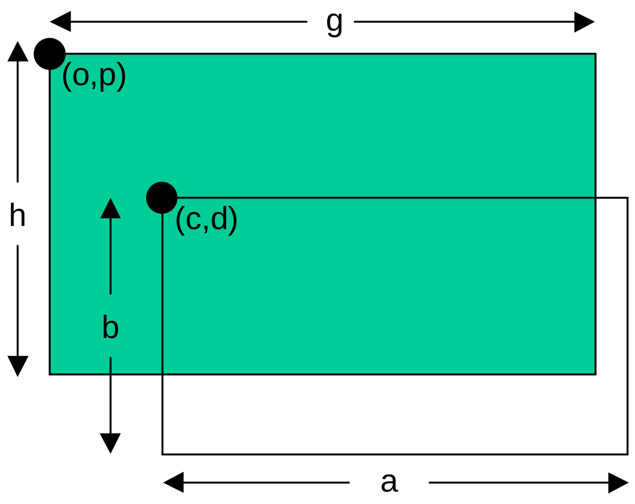
When there is no image data defined, then the image base color is used as I_k. The image base color is either explicitly defined in the Image Object Box, or a standard default value (black) is used.

The following figure shows the general arrangement of the mask object in Cases 1 and 4 in the table.



In these cases, codestream data is only present for the mask, which has dimensions e and f and is positioned as shown with respect to the layout object, which has dimensions a and b. Where there is mask codestream data within the layout object boundaries, then the Image Base color, defined in the Image Object Box, is used as I_k (Case 4). When there is no Image Object Box, then the standard default value (black) is used (Case 1).

The following figure shows the general arrangement of the image object in Case 2 in the table.



In this case, codestream data is only present for the image, which has dimensions g and h and is positioned as shown with respect to the layout object, which has dimensions a and b. Where there is image codestream data within the layout object boundaries, then a mask is taken to have the same dimensions and position as the image, and to have a value of 1. Any Image Base Color that may be defined in the Image Object Box is ignored. Therefore, within the bounaries of the layout object where image codestream data is defined, the result is described as follows:

$$PageImage_k = I_k$$

The last case is Case 3, where there is no mask object. An the image object exists, with dimensions g and h and positioned as shown in the previous figure with respect to the layout object. This image object has no codestream data, only an explicitly defined Image Base Color. This case is equivalent to Case 2, but with the Image Base Color used as I_k and laid down as the Page Image where the image object and layout object boundaries overlap.

These 5 cases correspond to the values of the Style field shown in the table. The Style field is contained in the Layout Object Header Box. The general rule is an image color is combined with the page image within the layout object boundaries only where the mask is defined, either explicitly with codestream data (there is a mask object with NoCodestream=0) or implicitly, where there is no mask object but there is an image object, which serves as its own mask. The image object can be defined by codestream data, an Image Base Color, or both. In cases where there is a mask object but no image object, then black is used as the Image Base Color.

5.2.4.1 JBIG2 Objects

In a JPM file, the mask object can be compressed using JBIG2 (ISO 14492 | ITU-T Rec. T.88). If JBIG2 is used to encode the mask object, then and only then may the image object be an encoding of color tags using ITU-T Rec. T.45, where the color tags are applied to the symbol occurrences in the JBIG2 codestream as described in ITU-T Rec.T.45. In this case, the Layout Object has Style=2 and contains a single object with an associated JBIG2 codestream containing an embedded T.45 codestream.

The color specification of the JP2 Header Box of this single object will give the color space for the tags. If a palette box is used, then the palette will be used to map single component labels in the T.45 codestream to multiple component values. The Bits per Component value will be the same as in the BPC Box.

5.2.5 Thumbnails

Thumbnails are an optional feature which allow an overview of a document or a page to be presented to an end user. There are two types of thumbnails: document thumbnails and page thumbnails.

A document thumbnail offers an overview representation of an entire document, much as the cover or spine of a book serves to do. It may simply be a repetition of the page thumbnail for the title page. Document thumbnails may be JP2 compatible or not.

A document thumbnail is created by adding a JP2 Header box for the thumbnail image immediately after the File Type box and using JPEG 2000 as the coder. The codestream for the thumbnail image must be contained in the first Contiguous Codestream box following the JP2 Header box.

A JP2 compatible document thumbnail appears to a JP2 reader to be a valid JP2 file. The JP2 compatibility flag is set in the File Type box by adding the string "jp2\40" to the compatibility string. A JP2 reader would then look for a JP2 Header box and jump over any following boxes until it finds a Contiguous Codestream box. It would then display the image represented by the data in that box. When a document thumbnail is present, the file would contain a JP2 Header Box and a Contiguous Codestream Box near the beginning of the file, somewhere after the Compound Image Header Box and before the first Page Box, usually before the Page Collection Box.

Page thumbnails allow an overview of each individual page to be displayed. This can be used during navigation, for example to render a small icon near each node of a tree-like table of contents. It can also be used to lay down a more useful initial image onto a new page than simply the page's background colour while waiting for additional content to arrive.

A page thumbnail is simply the first layout object in a page. Its presence is signalled by a flag in the page table which indicates that a given page pointed to from the page table contains a document thumbnail. Thumbnails are optional.

5.2.6 Contiguous and Fragmented Codestreams

The codestream data for the objects of a page can follow the Page Box in the file. This data may or may not be found in the file before the next Page box. The codestream data may either be a single contiguous codestream, or be composed of one or more codestream fragments. If the codestream consists of codestream fragments then the fragments are accessed via a Fragment Table box, which contains a Fragment List box giving the locations in the file of the codestream fragments, or pointers to them if the fragments are external and not contained within the file. Each codestream fragment is contained within a Fragment box. If the codestream is not fragmented then the object will contain the location of a Contiguous Codestream box containing the codestream data. Full details of these boxes may be found in Annex B.

The fragment table mechanism permits any codestream to be broken up into an ordered list of fragments, each pointed to by its own offset and data reference. A data reference indexes into the data reference table and produces a string listing the filename or URL of an external file where the fragment is to be found. If the data reference is 0, the fragment is located in the current file. This mechanism allows fragments to be placed in an optimal position to support streaming of the data to a client application from a server, or to allow progressive refinement of multiple codestreams simultaneously.

Fragments may start or stop at any byte boundary of the codestream. The length of each fragment is specified by the length parameter retrieved from the fragment table. If a fragment does not end at a point in the codestream where a full code block or full raster line can be decoded, the decoding application holds this additional information until the next fragment has been retrieved. The next fragment of the byte stream is then appended to the residual data from the prior fragment and then decoding is resumed. Codestreams having any compression type may de decoded in this way.

Fragment table entries always point to bare codestreams, not to files wrapped in file formats. All header information required to decode the codestream such as width and height, compression type or bit depth is stored into fields of the Object box or Layout Object box.

5.2.7 Shared Data

Objects or fragments that occur explicitly in one part of a JPM file can be included implicitly in another part of a JPM file by means of a Shared Data reference. Wherever a Shared Data Reference occurs at a point in the file, then the data it references is treated as if it exists at that point in the file. The shared data reference can use either an identifier to a Shared Data Entry Box or a data reference, including an offset and length pointing to an occurence of the shared data. Full details of the Shared Data Entry and Shared Data Reference Boxes may be found in Annex B.1

5.3 Self-Contained Files and Files With External References

The previous section outlined the logical structure of a JPM file. In a local file, typically all image content and parameter boxes are contained in a single file for ease of tracking and maintenance, and because local storage is cheap and local bandwidths are high.

The multi-layer model is a method of representing and constructing document images, which allows access paradigms based on the multiple layers in addition to providing compression advantages due to applying the optimal compression method to each layer. In addition, document images have an internal structure that implies a reading order, unlike photographic images, so that breaking them into layout objects can bring streaming advantages where only the small area of the page (a column, say, or a paragraph) that is of interest is brought over and refined.

The fragment table and cross reference features of the JPEG 2000 family of file formats provide enormous flexibility in the construction of files which are distributed across multiple files, some of which may be located remotely on networks such as the Internet. In this section, we examine some of the file organizations that these features enable.

Useful JPM files tend to have structural information boxes at the top and large codestream objects either at the bottom of the file or remotely located or both. This allows rapid parsing and seeking among the component parts of the file by first getting the necessary structural information and then seeking to the offsets specified in the local or remote files.

Structural information includes such information as Page Table Boxes, Page Collection Boxes, Label Boxes and Fragment Table Boxes.

Page Table Boxes are a key structural feature. They refer to Page Boxes via page references. These references contain an offset, length and data reference for the Page Box associated with each page in the complete file. Each of these data references may be zero, indicating a page reference to a Page Box in the current file, or non-zero, indicating that the data reference value is to be used as an index into the Data Reference Box. The Data Reference Box contains an enumerated list of strings, each string being a file name or a URL pointing to an Internet file.

In a client/server system, a common case might be that a user wishes to browse into a multi-page JPM file on a server, and wishes to begin on a page in the middle of that remote file. A common reason for this would be that a search engine found a hit on the JPM file metadata associated with that page. The URL returned by the search engine would pointto the JPM file and could contain a construction such as http://webimaging.org/test.jpm#page17. This would instruct the JPM decoder subsystem of the browser to abort the download of the full JPM file as soon as the key structural boxes at the top of the box have completed downloading. Then, byteserving protocols would be used to retrieve only the ranges of bytes needed to render the data in the Page Box associated through the Label Box to label "page17". Each Object box in each Layout Object box of a given Page box may contain a fragment table reference to a Fragment Table Box. By retrieving only enough of each layout object's Fragments to render a rough initial version of the page, a full rendering of all the layout objects can be deferred. The end user could in parallel begin moving the mouse over the browser's display window to indicate which layout objects might better be refined next based on their having been indicated of interest.

During this interactive browsing session, fragments are streaming over to the client copy of the JPM file in an order that is quite different than the order in the server copy. Page 17 comes first in the client copy, where page 1 was more likely first in the server copy. Similarly, the layout objects within page 17 will have a different order in the client version of the page, since the user's mouse movements dictated their retrieval. The client version can nevertheless form a perfectly equivalent surrogate for the server version because the page tables and fragment tables in the client copy will have started out pointing entirely to fragments on the server. As fragments and Page Boxes stream partially over to the client, the appropriate entries in the Page Table Boxes and Fragment Table Boxes are updated to point to the appropriate segments of the client copy of the file whether it is just in the file cache or explicitly saved.

When wholesale changes are made to the fragment table, then the fragment table pointer can be changed to point to a new, post-pended fragment table at the end of the file, and the old fragment table can be turned into a free box that can be recovered during a later garbage collection step on the file.

All the fragments of the data in the file could be retrieved either from another file located on the web server external to the server copy of the JPM file or from a file located on another web server, or from an external file locate d on the client machine. The fragment reference mechanism supports all of these sources of data fragments external to the main JPM file.

Similarly, the cross reference mechanism provides a means for JPM boxes (as opposed to data fragments) to be located in any of those possible places as well. The Cross Reference Box has an offset, length and data reference that points to an internal or external location so that a box can be found. For cross references internal to the main file, this mechanism permits repeated boxes to appear only once, with the second and subsequent instances being included by reference to the first instance. The key difference between fragment tables and cross references is that cross references point to the beginning of the referenced box, whereas fragment table entries point into the fragment box, beyond its contained Media Data Box and directly to the first byte of codestream in the codestream fragment. In addition, shared references may be used to reference boxes contained within the same file.

In addition to supporting a variety of streaming and remote browsing behaviours, the file format can support datastreams coming from high-speed document scanners, where the images are not yet optimised for viewing but must meet other stringent constraints, such as timing. These files are not heavily processed at scan time, but rather capture all the available image data (perhaps multiple images per page) coming from the scanner and capture software and save this data to make it available to downstream post-processing software that can convert it to a web-optimised form.

Annex A

Compound Image File Structure

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

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

A.1 File Identification

The brand shall be ‘jpm\040’ for files that are completely defined by this Part of this Recommendation | International Standard; brand is defined in Annex I of Part 1 of this Recommendation | International Standard. JPM files are files that conform to this Part of this Recommendation | International Standard. JPM files can be identified using several mechanisms. When stored in traditional computer file systems, files that conform to this Part should have the file extension .jpm (readers should allow mixed case). On Macintosh file systems, the type code should be ‘jpm\040’. The MIME type is image/jpm.

A.2 File Organization

A JPM file uses the file format architecture specified in Part 1, Annex I of this Recommendation | International Standard. Therefore, a JPM file is a collection of boxes. The binary structure of the file is a contiguous sequence of boxes. The start of the first box shall be the first byte of the file and the end of the last box shall be the last byte of the file. The binary structure of a box in a JPM file is identical to that defined in the JP2 file format (ITU-T T.800 | IS 15444-1 Annex I.6).

This Recommendation | International Standard defines boxes required by a JPM file, as well as several optional boxes. Other Recommendations | International Standards may define other boxes that may also be found within a JPM file. In all cases, the information contained within a JPM file shall be in the box format; byte-streams not in the box format shall not be found in a JPM file.

A JPM file may be self-contained in that it contains all the image data needed to composite the page or pages in the file. A JPM file may also reference images in external files, or layout objects, pages or page collections in an external file. References to the content of an external file are by means of the Data Reference box (Annex B.1.5).

Schematically, the hierarchical organization of boxes in a JPM file is shown in Figure A-1. Boxes with dashed borders are optional in conforming JPM files. However, an optional box may define mandatory boxes within that optional box. In that case, if the optional box exists, those mandatory boxes within the optional box shall exist. This illustration specifies only the containment relationship between the boxes in the file. A particular order of those boxes in the file is not generally implied. The definition of a box will include whether or not that box is required to be found at a specific location within the file or another box.

.

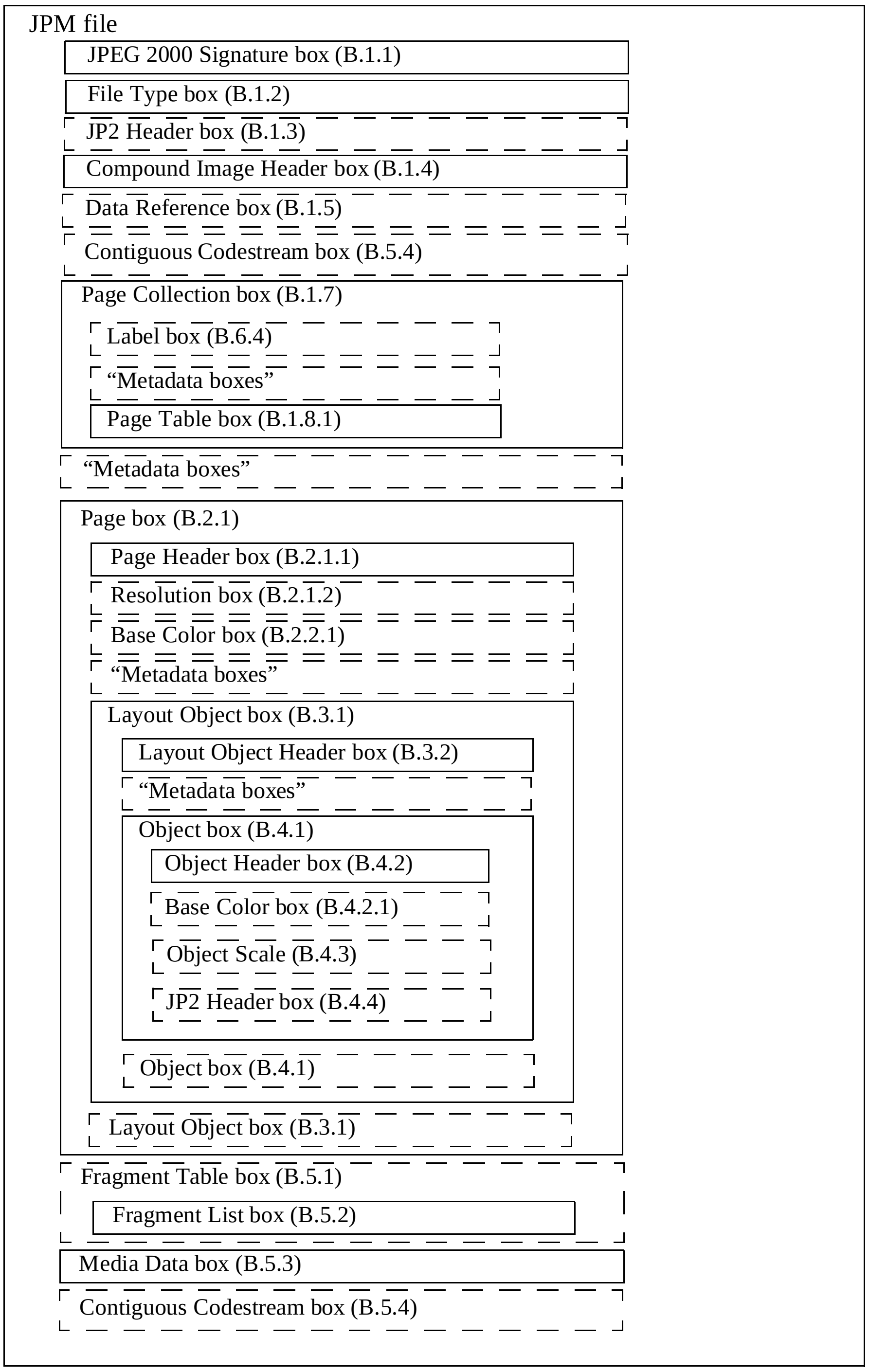

JPM file
JPEG 2000 Signature box (B.1.1)
File Type box (B.1.2)
JP2 Header box (B.1.3)
Compound Image Header box (B.1.4)
Data Reference box (B.1.5)
Contiguous Codestream box (B.5.4)
Page Collection box (B.1.7)
Label box (B.6.4)
“Metadata boxes”
Page Table box (B.1.8.1)
“Metadata boxes”
Page box (B.2.1)
Page Header box (B.2.1.1)
Resolution box (B.2.1.2)
Base Color box (B.2.2.1)
“Metadata boxes”
Layout Object box (B.3.1)
Layout Object Header box (B.3.2)
“Metadata boxes”
Object box (B.4.1)
Object Header box (B.4.2)
Base Color box (B.4.2.1)
Object Scale (B.4.3)
JP2 Header box (B.4.4)
Object box (B.4.1)
Layout Object box (B.3.1)
Fragment Table box (B.5.1)
Fragment List box (B.5.2)
Media Data box (B.5.3)
Contiguous Codestream box (B.5.4)

Figure A-1 — Conceptual structure of a JPM file

A JPM file contains a JPEG 2000 Signature Box, a File Type Box, a Compound Image Header Box, a Page Collection Box, one or more Page Boxes.

The JPEG 2000 Signature Box identifies the file as being part of the JPEG 2000 family of file formats.

The File Type box specifies file type, version and compatibility, including specifying if this file is a conforming JPM file or if it may be read, at least in part, by a conforming JP2 or JPX reader.

An optional JP2 Header box may follow the File Type box, describing a document thumbnail image for the JPM file.

The Compound Image Header box contains global information that describes the Compound Image file. It is analogous to the Start of Page marker segment in a T.44 datastream, as defined in ITU-T Rec. T.44 | ISO 16485.

The Page Collection box contains a Page Table box for locating the individual pages of a JPM file.

The Page box describes the page onto which its constituent layout objects are composited using the imaging model. It may also contain references to metadata associated with the page. The Page box also contains one or more layout objects.

A Compound Image File must contain a JPEG 2000 Signature box, a File Type box, a Compound Image Header box, and a Page Collection box, in that order. The file may or may not contain image data. Figure A-1 shows the data stored with the Compound Image File.

The JPM format allows the codestreams in the image layers to be non-contiguous. This enables interleaving codestream fragments among images and inter-image progressive rendering, where multiple images are rendered progressively at the same time.

The JPM format also allows the codestreams in the image layers to be stored outside but referenced from the Compound Image File. This means that the images used in a compound image can be stored separately from their parent compound image or JPM file.

Where not specified, all integer values are assumed to use the big-endian byte order.

A.3 Box Definition

Physically, each object in a JPM file is encapsulated within a binary structure called a box. That binary structure is as follows:

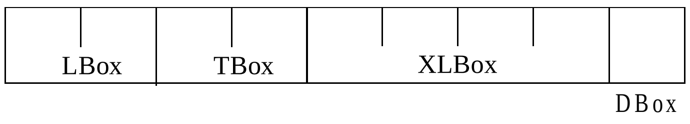


Figure 5-3 — Organization of a Box

LBox: Box Length. This field specifies the length of the box, stored as a 4-byte big endian unsigned integer. This Value includes all of the fields of the box, including the length and type. If the value of this field is 1, then the XLBox field shall exist and the value of that field shall be the actual length of the box. If the value of this field is 0, then the length of the box was not known when the LBox field was written. In this case, this box contains all bytes up to the end of the file. If a box of length 0 is contained within another box (its superbox), then the length of that superbox shall also be 0. This means that this box is the last box in the file. The values 2-7 are reserved for ISO use.

TBox: Box Type. This field specifies the type of information found in the DBox field. The value of this field is encoded as a 4-byte big endian unsigned integer. However, boxes are generally referred to by an ISO 646 character string translation of the integer value. For all box types defined within this Recommendation | International Standard, box types will be indicated as both character string (normative) and as 4-byte hexadecimal integers (informative). Also, a space character is shown in the character string translation of the box type as “\40”. All values of TBox not defined within this Recommendation | International Standard are reserved for ISO use.

XLBox: Box Extended Length. This field specifies the actual length of the box if the value of the LBox field is 1. This field is stored as an 8-byte big endian unsigned integer. The value includes all of the fields of the box, including LBox, TBox and XLBox fields.

DBox: Box Contents. This field contains the actual information contained within this box. The format of the box contents depends on the box type and will be defined individually for each type.

Annex B defines the boxes used in this Part of this Recommendation | International Standard. The following template is in the box definitions.

Box type:
Container:
Mandatory:
Quantity:
Location:

Box type is the value of the TBox field of the box. Mandatory indicates whether the box is required to be present in the box that contains it. Quantity is the number of boxes being defined that can occur in the Container. Location describes where in the Container the box being defined can occur.

A.4 Box Used in a Compound Image File

Table A-1 lists the boxes used in a JPM file and defined within this Recommendation | International Standard.

Table A-1 — Boxes defined within this Recommendation | International Standard

Box name	Type	Superbox	Required?	Comments
JPEG 2000 Signature	‘jP\040\040’ (0x6A50 2020)	No	Yes	This box uniquely defines the file as being part of the JPEG 2000 family of files.
File Type	‘ftyp’ (0x6674 7970)	No	Yes	This box specifies file type, version and compatibility information, including specifying if this file is a conforming JPM file.
Compound Image Header	‘mhdr’ (0x6D68 6472)	No	Yes	This box contains general information about the compound image file, such as profile and version.
Data Reference	‘dtbl’ (0x6474 626C)	No	No	This box contains a set of pointers to other files or data streams not contained within the JPM file itself.
Data Entry URL	‘url\040’ (0x7572 6C20)	No	No	This box specifes a URL.
Page Collection	‘pcol’ (0x70636F6C)	Yes	Yes	This box groups together the locations of a set of pages so that they are treated as related and associated with each other.
Label	‘lbl\040’ (0x6C62 6C20)	No	No	This box specifies a textual label for either a Page box or a Page Collection box.
Page Table	‘pagt’ (0x7061 6774)	No	Yes	This box gives the locations of the pages in a page collection and enables random access of pages in a file.

Table A-1 — Boxes defined within this Recommendation | International Standard

Box name	Type	Superbox	Required?	Comments
Page	‘page’ (0x7061 6765)	Yes	Yes	This box contains all the information needed to image a page including references to codestream data.
Page Header	‘phdr’ (0x7068 6472)	No	Yes	This box describes the properties of a page and gives the number of layout objects in the page.
Resolution	‘res\040’ (0x7265 7320)	Yes	Yes	This box specifies contains the grid resolution of a page.
Default Display Resolution	‘resd’ (0x7265 7364)	No	Yes	This box specifies the default grid resolution at which the image should be displayed.
Base Colour	‘bclr’ (0x6263 6272)	No	No	This box specifies the colour of a page or of an object for which no image data is available.
Layout Object	‘lobj’ (0x6C6F 626A)	Yes	Yes	This box contains the information and image data needed to composite a mask-image object pair.
Layout Object Header	‘lhdr’ (0x6C68 6472)	No	Yes	This box describes the properties of the layout object and assigns each a unique identifier within the page.
Object	‘objc’ (0x6F62 6A63)	Yes	Yes	This box contains all the image data and information for one object in a layout object.
Object Header	‘ohdr’ (0x6F68 6472)	No	Yes	This box describes the properties of an object, identifying it as a mask, an image or a combined mask/ image object, and assigns each object a unique identifier within the file.
Object Scale	‘scal’ (0x7363 616C)	No	No	This box describes the scaling for an object before applying it to the page.
JP2 Header	‘jp2h’ (0x6A70 3268)	Yes	No	This box contains a series of boxes that contain header-type information about a file containing a single image.
Image Header	‘ihdr’ (0x6968 6472)	No	Yes	This box specifies the size of the image and other related fields.
Bits Per Component	‘bpcc’ (0x6270 6363)	No	No	This box specifies the bit depths of the components of an image when the bit depth is not the same for all components.
Colour Specification	‘colr’ (0x636F 6C72)	No	Yes	This box specifies the colourspace of an image.
Palette	‘pclr’ (0x7063 6C72)	No	No	This box specifies the palette which maps a single component in index space to a multi-component image in color space.
Component Mapping	‘cmap’ (0x636D 6170)	No	No	This box specifies the mapping between a palette and codestream components.
Resolution	‘res\040’ (0x7265 7320)	Yes	No	This box specifies the palette which maps a single component in index space to a multi-component image in color space.

Table A-1 — Boxes defined within this Recommendation | International Standard

Box name	Type	Superbox	Required?	Comments
Fragment Table	‘ftbl’ (0x6674 626C)	Yes	No	This box describes how a codestream has been split up or fragmented and then stored within the file.
Fragment List	‘flst’ (0x666C 7374)	No	Yes	This box specifies a list of fragments that make up one particular codestream within the file.
Media Data	‘mdat’ (0x6D64 6174).	No	Yes	This box contains generic media data, which is referenced through the Fragment box.
Cross-Reference	‘cref’ (0x6372 6566)	No	No	This box specifies that a box found in another location (either within the JPM file or within another file) should be considered as if it was directly contained at this location in the JPM file.
Contiguous Codestream	‘jp2c’ (0x6A70 3263).	No	No	This box contains a valid and complete JPEG 2000 codestream.
Free	‘free’ (0x6672 6565)	No	No	This box contains data that is no longer used and may be overwritten when the file is updated.
Shared Data Entry	‘sdat’ (0x7264 6174)	Yes	No	This box contains a box that can be referenced by an identifier from multiple places within a file.
Shared Data Reference	‘sref’ (0x7372 6566)	No	No	This box can be used to insert a box in the file by reference to a previous occurence of the box in the same file.

Annex B

Box Definitions

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

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

The following boxes shall be interpreted by all conforming readers. Each of these boxes conforms to the standard box structure as defined in Annex A.3.

B.1 File Level Boxes

B.1.1 JPEG 2000 Signature box

Box type: 'jP\040\040' (X'6A502020')
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: First box in file

The JPEG 2000 Signature box identifies that the format of this file was defined by the JPEG 2000 Recommendation | International Standard, as well as providing a small amount of information which can help determine the validity of the rest of the file. The JPEG 2000 Signature box shall be the first box in the file, and all files shall contain one and only one JPEG 2000 Signature box.

The type of the JPEG 2000 Signature box shall be 'jP\040\040' (X'6A502020'). The length of this box shall be 12 bytes. The contents of this box shall be the 4-byte character string '<CR><LF><X'87'><LF>' (X'0D0A870A'). For file verification purposes, this box can be considered a fixed-length 12-byte string which shall have the value: X'0000000C 6A502020 0D0A870A'.

The combination of the particular type and contents for this box enable an application to detect a common set of file transmission errors. The CR-LF sequence catches bad file transfers that alter newline sequences, the final linefeed checks for the inverse of the CR-LF translation problem and the third character of the box contents has its high-bit set to catch bad file transfers that clear bit 7.

B.1.2 File Type box

Box type: 'ftyp' (X'66747970')
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: Immediately after the JPEG 2000 Signature box

The File Type box specifies the Recommendation | International Standard which completely defines all the contents of this file, as well as a separate list of readers, defined by other Recommendations | International Standards, with which this file is compatible, and thus the file can be properly interpreted within the scope of that other standard. This box shall immediately follow the JPEG 2000 Signature box. All files shall contain one and only one File Type box.

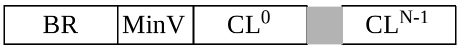


Figure B-1 — Organization of the contents of a File Type box

BR: Brand. This field specifies the Recommendation | International Standard which completely defines this file. This field is specified by a 4-byte string of ISO 646 characters. The value of this field is defined in Table B-1. In addition, the Brand field shall be considered functionally equivalent to a major version number. A major version change (if there ever is one), representing an incompatible change in the JPM file format, shall define a different value for the Brand field.

If the value of the Brand field is not 'jpm\040', then a value of 'jpm\040' in the Compatibility list indicates that a JPM reader can interpret the file in some manner as intended by the creator of the file.

Table B-1 — Legal Brand values

Value	Meaning
'jpm\040'	IS 15444-6 (This Recommendation | International Standard)
other values	Reserved for other ISO uses

MinV: Minor version. This parameter defines the minor version number of the JPM specification for which the file complies. The parameter is defined as a 4-byte big endian unsigned integer. The value of this field shall be zero. However, readers shall continue to parse and interpret this file even if the value of this field is not zero.

CL^i: Compatibility list. This field specifies a code representing this Recommendation | International Standard, another standard, or a profile of another standard, to which the file conforms. This field is encoded as a 4-byte string of ISO 646 characters. A file that conforms to this Recommendation | International Standard shall have at least one CL^i fields in the File Type box, and shall contain the value 'jpm\040' in one of the CL^i fields in the File Type box, and all conforming JPM readers shall properly interpret all files with 'jpm\040' in one of the CL^i fields.

Other values of the Compatibility list field are reserved for ISO use.

The number of CL^i fields is determined by the length of this box.

Table B-2 — Format of the contents of the File Type box

Field name	Size (bits)	Value
BR	32	$0 - (2^{32}-1)$
MinV	32	0
CL^i	32	$0 - (2^{32}-1)$

B.1.3 JP2 Header box (superbox) after File Type box

Box type: ‘jp2h’ (X'6A70 3268')
Container: File

Mandatory: No
Quantity: At most one
Location: Immediately after the File Type box

A JP2 Header box immediately following the File Type box describes a document level thumbnail of the JPM file. The first contiguous codestream following a File level JP2 Header box shall contain the associated codestream. A JP2 compatible JPM file may be created by adding 'jp2\40' to the compatibility list in the File Type box and using a document level thumbnail with a JPEG 2000 coder.

This box contains generic information about the document thumbnail, such as number of components, colourspace, and grid resolution. This box is a superbox. Within a JPM file, there shall be at most one File level JP2 Header box. The type of the JP2 Header box shall be 'jp2h' (X'6A70 3268'). The box is defined in Annex B.6.3.

B.1.4 Compound Image Header box

Box type: 'mhdr' (X'6D686472')
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: Immediately after the File Type box and any File level JP2 Header box.

The Compound Image Header box contains generic information about the compound image. This box contains the following fields.

VERS	NP	P	MC	IC	IPR

Figure B-2 — Organization of the contents of an Compound Image Header box

VERS: Version. This version defines the version number of this specification with which this file complies. This parameter is defined as a 2-byte unsigned big endian integer with the most significant byte containing the major version number (currently defined as 1) and the least significant byte containing a minor revision number (currently defined as 0). The value of this field is X'0100'.

NP: Number of pages. This parameter specifies the number of Page boxes in the compound image file and is stored as a 2-byte unsigned big endian integer. If not known, the value is 0.

P: Profile ID; application profile or mode that must be supported to read this file. This value is stored as a 1-byte unsigned integer. Values defined in this specification are specified in Table B-3.

Table B-3 — Legal P values

Value	Meaning
0	No profile specified
1	Web profile
2	T.44-compatible profile
other values	Reserved for ISO use.

MC: Mask coders; the coder or coders that are used to compress the mask objects of the layout objects in this file. This field can be one or more bytes long, with values set bit-by-bit as specified in Table B-4.

NOTE — Bit 7, the extend bit, would be set when adding another byte to accommodate additional coders, such as an eighth, which would be assigned to bit number 8.

Table B-4 — Mask Coders

Field Bit Number	Coder used
LSB 0	One dimensional T.4 (MH) coding
1	Two dimensional T.4 (MR) coding
2	T.6 (MMR) coding
3	T.82 (JBIG) coding applying Recommendation T.85
4	T.88 (JBIG2) coding applying ITU-T Rec, T.89
5	T.800 (JPEG 2000) coding
6	Uncompressed
MSB 7	Extend, add another octet that follows immediately

IC: Image coders; the coder or coders that are used to compress the image objects of the layout objects in this file. This field can be one or more bytes long, with values set bit-by-bit as specified in Table B-5.

Table B-5 — Image Coders

Field bit number	Coder used
LSB 0	T.81 (JPEG) coding
1	T.82 (JBIG) coding applying Recommendation T.43
2	T.800 (JPEG 2000) coding
3	T.87 (JPEG-LS) coding
4	T.45 (Run-Length Colour Encoding)
5	Uncompressed
6	Reserved
MSB 7	Extend, add another octet that follows immediately

NOTE — Bit 7, the extend bit, would be set when adding anotherbyte to accommodate additional coders, such as an eighth, which would be assigned to bit number 8.

IPR: Intellectual Property. This parameter indicates whether this JPM file contains intellectual property rights information. This value is stored as a 1-byte unsigned integer. If the value of this field is 0, this file and the image files it contains do not contain rights information, and thus the file does not contain an IPR box. If the value is 1, then the file does contain rights information and thus does contain an IPR box as defined in Annex I.6 of Part 1 of this Recommendation | International Standard. Other values are reserved for ISO use.

Table B-6 — Format of the contents of the Compound Image Header box

Field name	Size (bits)	Value
VERS	16	0x0100
NP	16	0 - (2^{16}-1)

Table B-6 — Format of the contents of the Compound Image Header box

Field name	Size (bits)	Value
P	8	see Table B-3
MC	variable	see Table B-4
IC	variable	see Table B-5
IPR	8	0 - 1

B.1.5 Data Reference box

Box type: ‘dtbl’ (X'6474 626C')
Container: File
Mandatory: No
Quantity: At most one
Location: Immediately after the Compound Image Header box

The Data Reference box contains an array of URLs that are referenced by this file. Many of these references will be from Fragment Table boxes, specifying the location of the codestream fragments. A JPM file may contain at most one Data Reference box and this box shall be at the top level of the file.

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

Data Reference box entries are used by Page Table boxes, Object Header boxes and Fragment List boxes to refer to data external to the JPM file.

A Data Reference and offset in a Page Table box is used to refer to an external Page box or Page Collection box.
A Data Reference and offset in an Object Header box is used to refer to an external Fragment Table box.
A Data Reference and offset in a Fragment List box within a Fragment Table box is used to refer to an external codestream fragment.
A Data Reference and offset in a Fragment List box within a Cross Reference box is used to refer to an external shared header or metadata fragment.

The type of the Data Reference box shall be ‘dtbl’ (X'6474 626C'), and its contents shall be as follows:

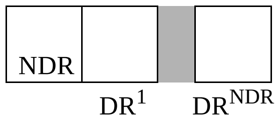


Figure B-3 — Organization of the contents of a Data Reference box

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 Annex C.2.3.2. 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 DRi fields is 1 based; a data reference of 1 in a DRi field within a Fragment List box or Page Table 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 file itself.

Table B-7 — Format of the contents of the Data Reference box

Field name	Size (bits)	Value
NDR	16	0—(2^{16}–1)
DR^i	Variable	Variable

B.1.6 Data Entry URL box

Box type: 'url\040' (X'7063 6F6C')
Container: File
Mandatory: No
Quantity: Any number
Location: Immediately after NDR field of Data Reference box

This box contains a URL which can be used by an application to aquire externally stored data. The URL type should be of a service which delivers a file (e.g. URL's of type file, http, ftp, etc.), which ideally also permits random access. Relative URL's are permissable and are relative to the file containing the Data Entry URL box.

The type of a Data Entry URL box shall be 'url\040' (X'75726C20'). The contents of a Data Entry URL box are defined in Annex C.2.3.2.

B.1.7 Page Collection box (superbox)

Box type: 'pcol' (X'7063 6F6C')
Container: File
Mandatory: No
Quantity: Any number
Location: The first Page Collection box may be located anywhere at the file level but for efficient access, an optimized file would have it located following the Data Reference box, if it exists, otherwise it would immediately follow the Compound Image Header box

A Page Collection box is a superbox that contains an optional Label box, optional metadata boxes, and a mandatory Page Table box. A Page Collection Box associates a collection of pages, whose locations are obtained using the Page Table box with an optional label and optional metadata.

A JPM file may or may not have a page collection. If it does have a page collection, it may be located anywhere at the file level. It is recommended that optimized files have at least one page collection and that it be placed near the beginning of the file as specified above.

Every page in a JPM file may belong to what is called its primary page collection. The role of the primary page collection is discussed in Section 5.2.1..

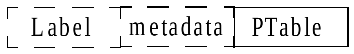


Figure B-4 — Organization of the contents of a Page Collection box

Label: Optional Label box

metadata: Optional metadata boxes

PTable: Page Table box

B.1.8 Label box in Page Collection box

Box type: 'lbl\040' (X'6C62 6C20')
Container: Page Collection box
Mandatory: No
Quantity: Any number

This box contains a textual label that may be associated with a Page Collection. The type of a Label box shall be ‘lbl\040’ (X'6C62 6C20'). The Label box is defined in Annex B.6.4.

B.1.8.1 Page Table box

Box type: ‘pagt’ (X'70616774')
Container: Page Collection box
Mandatory: Yes
Quantity: Exactly one in each Page Collection box
Location: Anywhere

The Page Table box contains the offsets to the Page boxes and Page Collection boxes within the page collection. The type of the Page Table Box shall be ‘pagt’ (X'70616774').

The box data contains the number of pages in the page collection and the number of entries in the page table. Each entry in the table is a reference to a Page box or a Page Collection box. A flag in each table entry indicates the type of box being referenced. The data field of the box is:

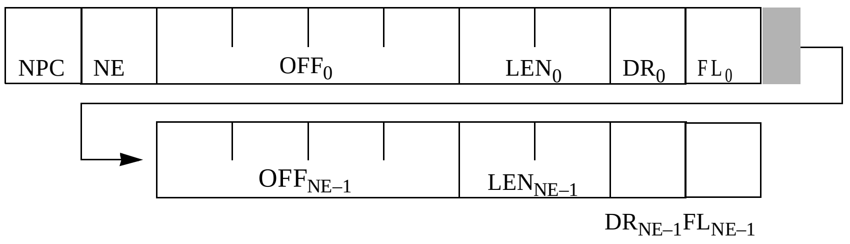


Figure B-5 — Organization of the contents of a PageTable box

NPC: Number of pages in the collection. This field specifies the number of pages in the page collection.

NE: Number of entries in the page table. The number of {OFF, LEN, DR, FL} tuples in the Page Table box shall be the same number as the value of the NE field.

OFF_i: Offset. This field specifies the offset to the first byte of the referenced Page or Page Collection box. The offset is relative to the first byte of the file (the first byte of the length field of the JPEG 2000 Signature box). This field is encoded as a 64-bit big endian unsigned integer.

LEN_i: Length of the page. This field specifies the length of the Page Box containing the page. This field is encoded as a 32-bit big endian unsigned integer.

DR_i: Data reference. This field specifies the data file or resource that contains this page. If the value of this field is zero, then the page is contained within this file. If the value is not zero, then the page is contained within the file specified by this index into the Data Reference box. This field is encoded as a 16-bit big endian unsigned integer.

FL_i: Flag. This field indicates the type of entry. This field is encoded as a 8-bit unsigned integer. Legal values of this field are as follows:

Table B-8 — Legal FL values

Value	Meaning
xxxx x001	Offset to Page box
xxxx x101	Offset to Page box containing thumbnail.
xxxx x010	Offset to Page Collection box.
xxxx 1xxx	Offset to Page or Page Collection box containing metadata
other values	Reserved for ISO use.

Table B-9 — Format of the contents of the Page Table box

Parameter	Size (bits)	Value
NPC	16	1—(2^{16}–1)
NE	16	1—(2^{16}–1)
OFF^i	64	12—(2^{64}–1)
LEN^i	32	0—(2^{32}–1)
DR^i	16	0—(2^{16}–1)
FL^i	8	see Table B-8

B.1.9 Shared Data Entry box

Box type: ‘sdat’ (X'7364 6174')
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere after the Compound Image Header box

The Shared Data Entry box contains an Identifier, a Type, a LastPageUsed field, and Shared Data, which is a box that can occur multiple times in the file. An example of Shared Data is a JP2 Header Box. Rather than replicate that box each time it is used, a reference to the box is used. The reference is contained in a Shared Data Reference Box. When this reference is encountered, it is replaced by the Shared Data box from the Shared Data Entry Box with the same identifier as used in the Shared Data Reference Box.

The type of a Shared Data Entry box shall be 'sdat' (X'73646174'). The contents of a Shared Data Entry box shall be as follows:

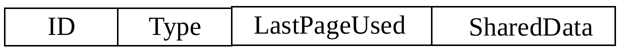


Figure B-6 — Organization of the contents of a Shared Data Entry box

ID: Identifier. This field specifies a number that is the unique identifier for the shared data contained in the box. It is encoded as a 2-byte unsigned integer.

LastPageUsed: The 1-based page ordinal of the last page that references this shared data. The page ordinal shall be based on the order in which pages are retrieved or accessed. A value of 0 indicates that the last page to use the shared data is not known. It is encoded as a 4-byte unsigned integer.

SharedData: This field contains the box that can be used multiple places in the file**.**

Table B-10 — Format of the contents of a Shared Data Entry box

Field name	Size (bits)	Value
ID	16	0-(2^{16}–1)
LastPageUsed	32	1-(2^{32}–1)
Shared Data	Variable	Variable

B.1.10 Shared Data Reference box

Box type: ‘sref’ (X'73726566')
Container: Not restricted
Mandatory: No
Quantity: Any number
Location: Anywhere after the Shared Data Entry Box with the same ID

The Shared Data Reference box contains a reference to the shared data that is to be used or inserted at the point in the file where the Shared Data Reference box occurs. When a Shared Data Reference box is encountered, it is replaced by the shared data from the Shared Data Entry Box with the ID value.

The type of a Shared Data Reference box shall be 'sref' (X'73726566'). The contents of a Shared Data Reference box shall be as follows:

Figure B-7 — Organization of the contents of a Shared Data Reference box

ID: Identifier. This field specifies a number that is the unique identifier of the shared data to be used. It is encoded as a 2-byte unsigned integer. It is a reference to the shared data in the Shared Data Entry Box with the same ID field value.

OFF: Offset. This field specifies the file offset to the start of the shared data. The offset is relative to the first byte of the file. This field is encoded as a 8-byte big endian unsigned integer.

LEN: Length. This field specifies the length of the referenced Shared Data Entry box. This field is encoded as a 32-bit bi endian unsigned integer.

Table B-11 — Format of the contents of the Shared Data Reference box

Field name	Size (bits)	Value
ID	16	0—(2^{16}–1)
OFF	64	0—(2^{64}–1)

Table B-11 — Format of the contents of the Shared Data Reference box

Field name	Size (bits)	Value
LEN	32	0—(2^{32}–1)

B.2 Page Level Boxes

B.2.1 Page box

Box type: ‘page’ (X'70616765')
Container: File
Mandatory: Yes
Quantity: At least one
Location: Anywhere after the Compound Image Header box

The Page box is a superbox that contains information about the page on which the layout objects are rendered, followed by the layout objects that make up the page contents. The type of the Page box shall be ‘page’ (X'70616765'). The data field of the box is:

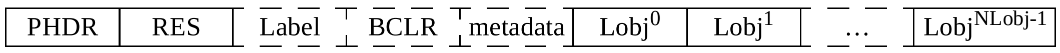


Figure B-8 — Organization of the contents of a Page box

PHDR: Page Header box.

Res: Resolution box. This box gives the resolution of the page.

Label: Label box. This box contains a label associated with the page; this box is optional

BCLR: Base Color box. The color of the page; this box is optional.

metadata: Optional metadata boxes.

$Lobj^i$: Layout Object box i, where i = 0, NLobj-1; the Layout Object boxes are specified in Annex B.3.

B.2.1.1 Page Header box

Box type: ‘phdr’(X'7068 6472')
Container: Page box
Mandatory: Yes
Quantity: Exactly one
Location: First box in a Page box

A Page Header box specifies a page identifier (unique within a file), a number of layout objects on the page and describes the height, width, orientation and colour of the page, i.e. the surface on which the layout objects are rendered. It also may optionally contain a set of 3 parameters that can point to the primary page collection to which the page belongs. The data field of the box is:

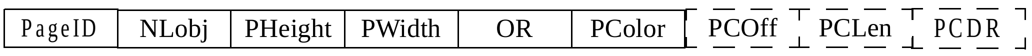


Figure B-9 — Organization of the contents of a Page Header box

PageID: Page Identifier; this field is 2 bytes long. Each page in the file has a unique Page ID.

NLobj: Number of layout objects on this page; this field is 2 bytes long.

PHeight: Page height. This field indicates the height, in page grid units and before any rotation, of the region onto which the layout objects on this page are rendered. This value is stored as a 4-byte big endian unsigned integer.

PWidth: Page width. This field indicates the width, in page grid units and before any rotation, of the region onto which the layout objects on this page are rendered. This value is stored as a 4-byte big endian unsigned integer.

OR: Page orientation. This field indicates the orientation of the page. The value is a 2-byte big endian unsigned integer. Values defined in this specification are:

Table B-12 — Legal OR values

Value	Meaning
0	Orientation not specified
1	Rotate 0° clockwise for a right-reading image
2	Rotate 90° clockwise for a right-reading image
3	Rotate 180° clockwise for a right-reading image
4	Rotate 270° clockwise for a right-reading image
other values	Reserved for ISO use.

PColorPage Color. This field indicates if the page is transparent, in which case the layout objects are imaged onto whatever exists within the boundaries of the page, or if the page has a color, in which case, the page color is applied everywhere within the boundaries of the page and then the layout objects are imaged onto the page. The value is a 2-byte big endian unsigned integer. Values defined in this specification are:

Table B-13 — Legal Pcolor values

Value	Meaning
0	Page is transparent
1	Page is white
2	Page is black
255	Page color is specified in Base Colour Box
other values	Reserved for ISO use.

PCOFF$_i$: Page Collection Offset. This field specifies the offset to the first byte of the primary page collection to which this page belongs. The offset is relative to the first byte of the file (the first byte of the length field of the JPEG 2000 Signature box). This field is encoded as a 64-bit big endian unsigned integer. If the PCOff field exists then the PCLen and PCDR fields shall also exist.

PCLEN$_i$: Length of the Page Collection box. This field specifies the length of the primary Page Collection box for this page. This field is encoded as a 32-bit big endian unsigned integer. If the PCLen field exists then the PCOff and PCDR fields shall also exist.

PCDR$_i$: Data reference. This field specifies the data file or resource that contains this page. If the value of this field is zero, then the primary Page Collection box is contained within this file. If the value is not zero, then the page is contained within the file specified by this index into the Data Reference box. This field is encoded as a 16-bit big endian unsigned integer. If the PCDR field exists then the PCOff and PCLen fields shall also exist.

Table B-14 — Format of the contents of a Page Header box

Field name	Size (bits)	Value
PageID	16	0-(2^{16}–1)
NLobj	16	1-(2^{16}–1)
PHeight	32	1-(2^{32}–1)
PWidth	32	1-(2^{32}–1)
OR	16	see Table B-12
PColor	16	see Table B-13
PCOff	0 or 64	see above
PCLen	0 or 32	see above
PCDR	0 or 16	see above

B.2.1.2 Resolution box (superbox) in Page box

Box type: ‘res\040’ (X'7265 7320')
Container: Page box
Mandatory: Yes
Quantity: Exactly one
Location: Immediately follows the Page Header box

The Resolution box is defined in Annex B.6.1. The Resolution box within a Page box specifies the default display grid resolution of the page and shall contain a Default Resolution box only. The type of a Resolution box shall be 'res\040' (X'72657320').

B.2.2 Label box in Page box

Box type: 'lbl\040' (X'6C62 6C20')
Container: Page box
Mandatory: No
Quantity: Any number

This box contains a textual label that may be associated with a Page. The type of a Label box shall be ‘lbl\040’ (X'6C62 6C20'). The Label box is defined in Table B.6.4.

B.2.2.1 Base Color box in Page box

Box type: ‘bclr’(X'6263 6C72')
Container: Page box
Mandatory: No
Quantity: At most one
Location: If present, immediately follows the Resolution box

The Base Color box occurs in a Page box when the value of the PColor field in the Page Header box is 255. Within a Page Box it specifies the color that is applied prior to rendering any objects onto the page. The Base Colour box is defined in Annex B.6.1

B.3 Layout Object Level Boxes

B.3.1 Layout Object box (superbox)

Box type: ‘lobj’ (X'6C6F626A')
Container: Page box
Mandatory: Yes
Quantity: Any number

A Layout Object box contains the information needed to position and composite an image and optional mask on a page. It contains the Layout Object Header box, optional metadata, an Object Header box (for image data) and an optional Object Header box (for mask data).

The data field of the box is:

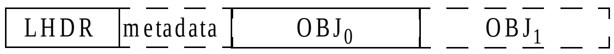


Figure B-10 — Organization of the contents of a Layout Object box

LHDR: Layout Header box

OBJ_0: Object box; the first Object box.

OBJ_1: Object box; the second Object box.

B.3.2 Layout Object Header box

Box type: 'lhdr' (X'6C68 6472')
Container: Layout Object box
Mandatory: Yes
Quantity: Exactly one

This box gives the layout object ID (unique within a page), height (in page grid units), width (in page grid units) and style of the layout object. The data field of the box is:

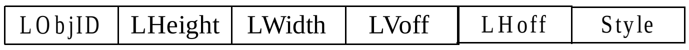


Figure B-11 — Organization of the contents of a Layout Object Header box

LObjID: Layout Object Identifier; this field is 2 bytes long. Each layout object in a given Page box has a unique LObjID value. A layout object with a lower LObjID value is imaged before a layout object with a higher LObjID value. Therefore, the LObjID specifies the sequence in which layout objects are imaged on the page.

LHeight: Layout object height. This field indicates the height, in page grid units, of the layout object before any page rotation. This value is stored as a 4-byte big endian unsigned integer.

LWidth: Layout object width. This field indicates the width, in page grid units, of the layout object before any page rotation. This value is stored as a 4-byte big endian unsigned integer.

LVoff: Layout vertical offset. This field indicates the vertical starting offset in page grid units of the layout object. This value is stored as a 4-byte big endian unsigned integer.

LHoff: Layout horizontal offset. This field indicates the horizontal starting offset in page grid units of the layout object. This value is stored as a 4-byte big endian unsigned integer.

Style: Layout Object Style. This field indicates the style of the layout object. A layout object can comprise of either a single image or an image and a mask pair. The image and mask of a layout object may consist of two separate objects or the mask may be the last component of the image object. The value is a 2-byte big endian unsigned integer. Values defined in this specification are:

Table B-15 — Legal Style values

Value	Meaning
0	Separate objects for image and mask components.
1	Single object for image and mask components.
2	Single object for image components only (no mask).
3	Single object for mask components only (no image).
255	Vendor specified layout object.
other values	Reserved for ISO use.

Table B-16 — Format of the contents of the Layout Object Header box

Field name	Size (bits)	Value
LObjID	16	0-(2^{16}-1)
LHeight	32	0-(2^{32}-1)
LWidth	32	0-(2^{32}-1)
LVoff	32	0-(2^{32}-1)
LHoff	32	0-(2^{32}-1)
Style	16	see Table B-15

B.4 Object Level Boxes

B.4.1 Object box (superbox)

Box type: ‘objc’
Container: Layout Object box
Mandatory: Yes
Quantity: At least one in each Layout Object box
Location: Anywhere after the Layout Object Header box

The Object Box is a superbox that contains information about the object of a layout object and pointers to the object's image data. The type of the Page Box shall be 'objc'. The data field of the box is:

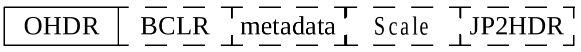


Figure B-12 — Organization of the contents of an Object box

OHDR: Object Header box.

BCLR: Optional Base Color box.

metadata: Optional metadata box.

Scale: Optional Object Scale box. Specifies the scaling for the object before applying it to the page

JP2HDR: Optional JP2 Header Box - always exists if the NoCodestream field in the Object Header box is 0

B.4.2 Object Header Box

Box type: 'ohdr' (X'6F61 6765')
Container: Object box
Mandatory: Yes
Quantity: Exactly one
Location: First box in the Object box

This box contains fields that describe general properties of the object. The fields are: ObjType (if this object is used only for a mask the type is '0', if only for the image the type is '1', if for the image and mask the value is '2') NoCodestream (if value of this parameter is '1', then this object is colored by a unique color specified by the Base Color Box in the Object box), OHoff (horizontal offset in page grid units), OVoff (vertical offset in page grid resolution), and a reference to the codestream data for the object - either an offset to and the length of a Fragment Table or Contiguous Codestream box. The fields are organized in the box as follows:

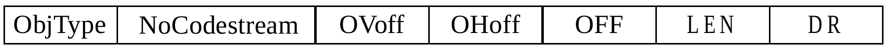


Figure B-13 — Organization of the contents of an Object Header box

ObjType: Object Type; Identifies whether the object is a mask object or an image object in the Layout Object. The value is a 2-byte big endian unsigned integer. Values defined in this specification are:

Table B-17 — Legal ObjType values

Value	Meaning
0	The object contains the mask of a layout object
1	The object contains the image of a layout object
2	The object contains the image and mask of a layout object
other values	Reserved for ISO use.

NoCodestream: Identifies whether or not the object contains a compressed codestream. If the object does not contain a codestream, then there is no image for this object and an "image" with a single uniform color value as specified in the Base Color box is used as the object. The value is a 2-byte big endian unsigned integer. Values defined in this specification are:

Table B-18 — Legal NoCodestream values

Value	Meaning
0	The object contains a compressed codestream
1	The object does not contain a compressed codestream
other values	Reserved for ISO use.

OVoff: Vertical offset in the object; the vertical offset into the object in page grid units. This field is 4-bytes big endian unsigned integer. If the NoCodestream field has a value of 1, then this field is ignored.

OHoff: Horizontal offset in the object; the horizontal offset into the object in page grid units. This field is 4-byte big endian unsigned integer. If the NoCodestream field has a value of 1, then this field is ignored.

OFF: Offset. This field specifies the file offset to the start of the Contiguous Codestream or Fragment Table box that is associated with this object's image. The Contiguous Codestream or Fragment Table box is used to access the actual codestream. The offset is relative to the first byte of the file. This field is encoded as a 8-byte big endian unsigned integer.

LEN: Length of the Contiguous Codestream or Fragment Table box. This value includes only the actual data and not any headers of an encapsulating box. This field is encoded as a 4-byte big endian unsigned integer. If value is 0, then the length of the Contiguous Codestream or Fragment Table box is not known. If the NoCodestream field has a value of 1, then this field is ignored.

DR: Data reference. This field specifies the data file or resource that contains the Fragment Table box.. If the value of this field is zero, then the fragment is contained within this file or the data is contained in a Contiguous Codestream box within the file. If the value is not zero, then the fragment table box is contained within the file specified by this index into the Data Reference box. This field is encoded as a 2-byte big endian unsigned integer. If the NoCodestream field has a value of 1, then this field is ignored.

Table B-19 — Format of the contents of the Object Header box

Field name	Size (bits)	Value
ObjID	16	0-(2^{16}-1)
ObjType	16	see Table B-17
NoCodestream	16	see Table B-18
OVoff	32	0-(2^{32}-1)
OHoff	32	0-(2^{32}-1)
OFF	64	0-(2^{64}-1)
LEN	32	0-(2^{32}-1)
DR	16	0-(2^{16}-1)

B.4.2.1 Base Color box in Object box

Box type: 'bclr'(X'6263 6C72')
Container: Object box
Mandatory: No

Quantity: At most one
Location: If present, immediately follows the Object Header box

The Base Color box within the Object box is used to specify the values to be used within the boundaries of the layout object in areas where no image is defined. This includes the situation where the object NoCodestream field is 1 and the situation when the scaled and positioned object image is smaller than the layout object. The Base Color box within a mask Object box must use the Grayscale enumerated colorspace. The Base Colour box is defined in Annex B.6.1.

B.4.3 Object Scale box

Box type: ‘scal’ (X'7363 616C')
Container: Object box
Mandatory: No
Quantity: At most one in each Object box
Location: Anywhere after the Object Header box

The Object Scale box optionally exists when the NoCodestream field of the Object Header box in the container Object box is 0. If an Object Scale box does not exist within an Object box then the vertcial and horizontal scaling ratios are assumed to be 1, i.e. no scaling. This box describes the scaling required to transform from the object units to the page grid units. This scaling must be used before applying the object to the page.

The vertical scaling is by the ratio $\frac{VRN}{VRD}$, and the horizontal scaling is by the ratio $\frac{HRN}{HRD}$. The scaling algorithm used will affect the appearance of the re-composed image. Bitonal mask images should be scaled to a grey image using bi-linear interpolation for proper results. The scaling algorithm used on image objects is left to the implementor, but will also affect image quality, if perhaps to a lesser degree

VRN	VRD	HRN	HRD

Figure B-14 — Organization of the contents of an Object Header box

VRN: Vertical scaling numerator; this parameter is encoded as a 2-byte big endian unsigned integer.

VRD: Vertical scaling denominator; this parameter is encoded as a 2-byte big endian unsigned integer.

HRN: Horizontal scaling numerator; this parameter is encoded as a 2-byte big endian unsigned integer.

HRD: Horizontal scaling denominator; this parameter is encoded as a 2-byte big endian unsigned integer.

Table B-20 — Format of the contents of the Object Scale box

Field name	Size (bits)	Value
VRN	16	0-(2^{16}-1)
VRD	16	1-(2^{16}-1)
HRN	16	0-(2^{16}-1)
HRD	16	1-(2^{16}1)

B.4.4 JP2 Header box (superbox) in Object box

Box type: ‘jp2h’ (X'6A70 3268')
Container: Object box

Mandatory: No
Quantity: At most one in each Object box - Always exists if the NoCodestream field in the Object Header box is 0
Location: Immediately after the Object Scale box

A JP2 Header box exists within an Object box when the NoCodestream field of the Object Header box in the container Object box is 0. This box contains generic information about an object, such as number of components, colourspace, and grid resolution. This box is a superbox. Within an Object box of a JPM file, there shall be one and only one JP2 Header box. The type of the JP2 Header box shall be ‘jp2h’ (X'6A70 3268'). The box is defined in Annex B.6.3.

B.5 Codestream Element Boxes

B.5.1 Fragment Table box

Box type: ‘ftbl’ (X'6672 626C')
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere

A Fragment Table box specifies the location of one of the codestreams in a JPM 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’ (X'6674 626C'), and its contents shall be as follows:

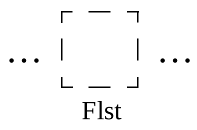


Figure B-15 — Organization of the contents of a FragmentTable box

FLst: Fragment List.This field contains a Fragment List box as specified in Annex B.5.2.

B.5.2 Fragment List box

Box type: ‘flst’ (X'666C 7374')
Container: Fragment Table box
Mandatory: Yes
Quantity: Exactly one
Location: Anywhere

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 of 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 box header of the referenced box.

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

The type of the Fragment List box shall be ‘flst’ (X'666C 7374') and it shall have the following contents:

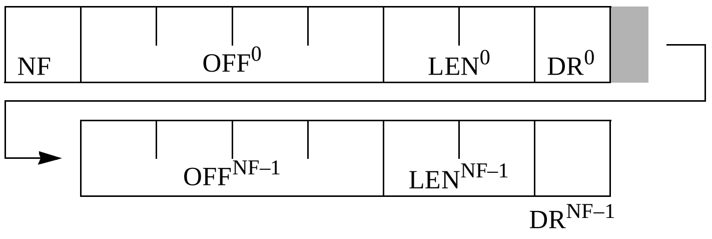


Figure B-16 — Organization of the contents of a Fragment List box

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

OFF^i: Offset. This field specifies the offset to the start of the fragment in the file. The offset is relative to the first byte of the file (the first byte of the length field of the JPEG 2000 signature box). This field is encoded as a 8-byte big endian unsigned integer. Only the first fragment in a Fragment List contained within a Cross-Reference box shall point to the first byte of a box header.

LEN^i: Length of fragment. This field specifies the length of the fragment. This value includes only the actual data and not any headers of an encapsulating box. This field is encoded as a 4-byte big endian unsigned integer.

DR^i: Data reference. This field specifies the data file or resource that contains this fragment. If the value of this field is zero, then the fragment is contained within this file. If the value is not zero, then the fragment is contained within the file specified by this index into the Data Reference box. This field is encoded as a 2-byte big endian unsigned integer.

Table B-21 — Format of the contents of the Fragment List box

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

B.5.3 Media Data box

Box type: 'mdat' (X'6D64 6174')
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere

The Media Data box contains fragments of the JPEG 2000 codestream or other media data. Applications should not access Media Data boxes directly, but instead use the fragment table to determine what parts of which Media Data boxes represent a valid JPEG 2000 codestream or other media stream.

The type of a Media Data box shall be ‘mdat’ (X'6D64 6174'). The contents of a Media Data box in general are not defined by this Recommendation | International Standard.

B.5.4 Contiguous Codestream box

Box type: 'jp2c' (X'6A70 3263')
Container: File
Mandatory: No
Quantity: Any number

The Contiguous Codestream box contains a valid and complete JPEG 2000 codestream. JPEG 2000 codestream data may be found in a Contiguous Codestream box rather than a Media Data box when the codestream data is contained in a JP2 file. The type of a Contiguous Codestream box shall be ‘jp2c’ (X'6A70 3263').

B.6 General / Common Boxes

B.6.1 Base Color box

Box type: ‘bclr’(X'6263 6C72')
Container: Page box or Object box

A Base Colour box specifies a base colour for an Object or a Page.

The type of a Base Colour box shall be ‘bclr’ (X'6263 6C72'). The data field of the box is:

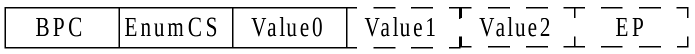


Figure B-17 — Organization of the contents of a Base Color box

BPC: Bits per component for the Base Color; Legal value of this field are as follows:

Table B-22 — Legal BPC values

Value	Meaning
x000 0000 - x010 0101	Component bit depth = value + 1. From 1 to 38 bits (counting the sign bit, if appropriate).
0xxx xxxx	Components are unsigned values.
1xxx xxxx	Components are signed values.
other values	Reserved for ISO use.

EnumCS: Enumerated colorspace; this field is defined in Annex B.6.3.6.

Values: The values of the Base Color in the enumerated color space. The number of values depends on the enumerated color space and is interpreted according to the BPC field. For EnumCS=17, there is one value; for EnumCS=3, 14 or 16, there are 3 values. Each value is a 2-byte big endian unsigned integer

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 colourspace has been encoded in the JPM file. 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. In JPM, a EP field may only be defined when EnumCS is 14. The format of the EP field when EnumCS is 14 is described in Annex B.6.3.8.

Table B-23 — Format of the contents of the Base Colour box

Field name	Size (bits)	Value
BPC	8	see Table B-22
EnumCS	32	see Table B-30
Values$_i$	variable	see above
EP	variable	see above

B.6.2 Resolution box (superbox)

Box type: ‘res\040’ (X'7265 7320')

This box specifies the capture and default display grid resolutions of an object or image. A Resolution box shall contain either a Capture Resolution box, a Default Display Resolution box or both. A Resolution box in a JPM file may be found within a JP2 Header box, Annex B.6.3, or within a Page box, Annex B.2.1.2.

The type of a Resolution box shall be 'res\040' (X'72657320'). The contents of the Resolution box are as follows:

RESC RESD

Figure B-18 — Organization of the contents of a Resolution box

RESC: Capture Resolution box. This box specifies the grid resolution at which this object was captured. The format of this box is specified in Annex B.6.2.1

RESD: Default Display Resolution box. This box specifies the default grid resolution at which this page/object should be displayed. The format of this box is specified in Annex B.6.2.2

B.6.2.1 Capture Resolution box

Box type: ‘resc’ (X'7265 7363')
Container: Resolution box
Mandatory: No
Quantity: At most one
Location: Anywhere

This box specifies the grid resolution at which the source was digitized to create the image samples specified by the codestream. The vertical and horizontal capture grid resolutions are calculated using the six parameters (Table B-24) stored in this box in the following two equations, respectively:

$$VRc = \frac{VRcN}{VRdD} \times 10^{VRcE} \tag{B.1}$$

$$HRc = \frac{HRcN}{HRcD} \times 10^{HRcE} \tag{B.2}$$

The values *VRc* and *HRc* are always in reference grid points per metre. If an application requires the grid resolution in another unit, then that application must apply the appropriate conversion.

The type of a Capture Resolution box shall be 'resc' (X'72657363'). The contents of the Capture Resolution box are as follows:

VRcN	VRcD	HRcN	HRcD	VRcE	HRcE

Figure B-19 — Organization of the contents of a Capture Resolution box

VRcN: Vertical Capture grid resolution numerator. This parameter specifies the VRcN value in Equation B.1, which is used to calculate the vertical capture grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

VRcD: Vertical Capture grid resolution denominator. This parameter specifies the VRcD value in Equation B.1, which is used to calculate the vertical capture grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

HRcN: Horizontal Capture grid resolution numerator. This parameter specifies the HRcN value in Equation B.2, which is used to calculate the horizontal capture grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

HRcD: Horizontal Capture grid resolution denominator. This parameter specifies the HRcD value in Equation B.2, which is used to calculate the horizontal capture grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

VRcE: Vertical Capture grid resolution exponent. This parameter specifies the VRcE value in Equation B.1, which is used to calculate the vertical capture grid resolution. This parameter is encoded as a twos-complement 1-byte signed integer.

HRcE: Horizontal Capture grid resolution exponent. This parameter specifies the HRcE value in Equation B.2, which is used to calculate the horizontal capture grid resolution. This parameter is encoded as a twos-complement 1-byte signed integer

Table B-24 — Format of the contents of the Capture Resolution box

Field name	Size (bits)	Value
VRcN	16	1 - (2^{16}-1)
VRcD	16	1 - (2^{16}-1)
HRcN	16	1 - (2^{16}-1)
HRcD	16	1 - (2^{16}-1)
VRcE	8	-128 - 127
HRcE	8	-128 - 127

B.6.2.2 Default Display Resolution box

Box type: ‘resd’ (X'7265 7364')
Container: Resolution box
Mandatory: No
Quantity: At most one
Location: Anywhere

This box specifies a desired display grid resolution for a page. The vertical and horizontal display grid resolutions are calculated using the six parameters (Tab leB-25) stored in this box in the following two equations, respectively:

$$VRd = \frac{VRdN}{VRdD} \times 10^{VRdE} \qquad \text{B.3}$$

$$Rd = \frac{HRdN}{HRdD} \times 10^{HRdE} \qquad \text{B.4}$$

The values *VRd* and *HRd* are always in reference grid points per metre. If an application requires the grid resolution in another unit, then that application must apply the appropriate conversion.

The type of a Default Display Resolution box shall be 'resd' (X'72657364'). The contents of the Default Display Resolution box are as follows:

VRdN	VRdD	HRdN	HRdD	VRdE	HRdE

Figure B-20 — Organization of the contents of a Default Display Resolution box

VRdN: Vertical Display grid resolution numerator. This parameter specifies the VRdN value in Equation B.3, which is used to calculate the vertical display grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

VRdD: Vertical Display grid resolution denominator. This parameter specifies the VRdD value in Equation B.3, which is used to calculate the vertical display grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

HRdN: Horizontal Display grid resolution numerator. This parameter specifies the HRdN value in Equation B.4, which is used to calculate the horizontal display grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

HRdD: Horizontal Display grid resolution denominator. This parameter specifies the HRdD value in Equation B.4, which is used to calculate the horizontal display grid resolution. This parameter is encoded as a 2-byte big endian unsigned integer.

VRdE: Vertical Display grid resolution exponent. This parameter specifies the VRdE value in Equation B.3, which is used to calculate the vertical display grid resolution. This parameter is encoded as a twos-complement 1-byte signed integer.

HRdE: Horizontal Display grid resolution exponent. This parameter specifies the HRdE value in Equation B.4, which is used to calculate the horizontal display grid resolution. This parameter is encoded as a twos-complement 1-byte signed integer

Table B-25 — Format of the contents of the Default Display Resolution box

Field name	Size (bits)	Value
VRdN	16	1 - (2^{16}-1)
VRdD	16	1 - (2^{16}-1)
HRdN	16	1 - (2^{16}-1)
HRdD	16	1 - (2^{16}-1)
VRdE	8	-128 - 127
HRdE	8	-128 - 127

B.6.3 JP2 Header box (superbox)

Box type: ‘jp2h’ (X'6A70 3268')

The JP2 Header box is a superbox with type ‘jp2h’ (X'6A70 3268'). A JP2 Header box in a JPM file may be found immediately after the File Type box, Annex B.1.3, or within an Object box, Annex B.4.4.

This box contains several boxes. Other boxes may be defined in other standards and may be ignored by conforming readers. Those boxes contained within the JP2 Header box that are defined within this Recommendation | International Standard are as follows:

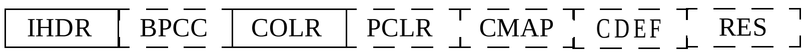


Figure B-21 — Organization of the contents of a JP2 Header box

IHDR: Image Header box. This box specifies information about the object, such as its height and width. Its structure is specified in Annex B.6.3.1. This box shall be the first box in the JP2 Header box.

BPCC: Bits Per Component box. This box specifies the bit depth of each component in the image. Its structure is specified in Annex B.6.3.2. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

COLR: Colour Specification box. This box specifies the colourspace of the decompressed image. Its structure is specified in Annex B.6.3.6. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

CMAP: Component Mapping box. This box defined in Annex B.6.3.4. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

CDEF: Channel Definition box. This box defined in Annex B.6.3.5. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

RES: Resolution box. This box specifies the capture and default display grid resolutions of the object. Its structure is specified in Annex B.6.1. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

The JP2 Header Box is a superbox that in a JPM file shall contain an Image Header box, a Color Specification box, an optional Resolution box, and an optional Palette box and Component Mapping box.

The JP2 Header box can be replaced by a Cross Reference box or a Shared Reference box, which specifies that a box found in another location, either within the JPM file or in another file, should be considered as if it were directly contained at this location in the JPM file.

B.6.3.1 Image Header box

Box type: ‘ihdr’ (X'6968 6472')
Container: JP2 Header box
Mandatory: Yes
Quantity: Exactly one
Location: First box in JP2 Header box

This box contains fixed length generic information about the object, such as the image size and number of components. The contents of the JP2 Header box shall start with an Image Header box. Objects that contain contradictory information between the Image Header box and the object codestream are not conforming files.

The type of the Image Header box shall be ‘ihdr’ (X'6968 6472') and the contents of the box shall have the following format

HEIGHT	WIDTH	NC	BPC	C	UnkC	IPR	CProfile

Figure B-22 — Organization of the contents of an Image Header box

HEIGHT: Image area height. The value of this parameter indicates the height of the image area. This field is stored as a 4-byte big endian unsigned integer.

WIDTH: Image area width. The value of this parameter indicates the width of the image area. This field is stored as a 4-byte big endian unsigned integer.

NC: Number of components. This parameter specifies the number of components in the codestream and is stored as a 2-byte big endian unsigned integer.

BPC: Bits per component. This parameter specifies the bit depth of the components in the image, minus 1, and is stored as a 1-byte field.
If the bit depth and the sign are the same for all components, then this parameter specifies that bit depth. If the components vary in bit depth and/or sign, then the value of this field shall be 255 and the JP2 Header box shall also contain a Bits Per Component box defining the bit depth of each component (as defined in Annex B.6.3.2).
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. See Table B-28 for legal values of C

UnkC: Colourspace Unknown. This field specifies if the actual colourspace if the image is known. This field is encoded as a 1-byte unsigned integer. Legal values for this field are 0, if the colourspace of the image is known and correctly specified in the Colourspace Specification box within the file, or 1, if the colourspace of the image is not known. A value of 1 will be used in cases such as the transcoding of legacy images where the actual colourspace of the image data is not known. Values other than 0 and 1 are reserved for ISO use.

IPR: Intellectual Property. This parameter indicates whether this JP2 file contains intellectual property rights information. If the value of this field is 0, this image does not contain rights information. If the value is 1, then the image does have associated right information and thus does contain, and thus the file This field specifies if the actual colourspace if the image is known. This field is encoded as a 1-byte unsigned integer. Legal values for this field are 0, if the colourspace of the image is known and correctly specified in the Colourspace Specification box within the file, or 1, if the colourspace of the image is not known. A value of 1 will be used in cases such as the transcoding of legacy images where the actual colourspace of the image data is not known. Values other than 0 and 1 are reserved for ISO use.

CProfile: Compression type profile. This parameter specifies the profile (compression options) of the compression algorithm used to compress the image data. See Table B-29 for legal values of CProfile It is encoded as a 2-byte big endian unsigned integer.

Table B-26 — 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)

Table B-26 — Format of the contents of the Image Header box

Field name	Size (bits)	Value
NC	16	1 - 16 384
BPC	8	See Table B-27
C	8	See Table B-28
UnkC	8	0-1
IPR	8	0-1
CProfile	16	see Table B-29

.

Table B-27 — BPC Values

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

Table B-28 — Legal C values for the Image Header box

Value	Meaning
1	One dimensional T.4 (MH) coding. This value is only permitted for bi-level images.
2	Two dimensional T.4 (MR) coding. This value is only permitted for bi-level images.
3	T.6 (MMR) coding. This value is only permitted for bi-level images.
4	T.82 (JBIG) coding.
5	T.81 (JPEG) coding.
6	T.87 (JPEG-LS) coding.
7	T.800 (JPEG 2000) coding.
8	T.88 (JBIG2) coding. This value is only permitted for bi-level images.
9	T.45 (Run Length) coding.

Table B-28 — Legal C values for the Image Header box

Value	Meaning
255	Uncompressed **Editor's note: Check against SPIFF value for uncompressed**
other values	Reserved for ISO use.

Table B-29 — Legal CProfile values for the Image Header box

Value of MS Byte	Value of LS Byte	Meaning
0	0	Undefined
1	0	One dimensional T.4 (MH) coding. EOLs are not byte aligned
1	4	One dimensional T.4 (MH) coding. EOLs are byte aligned
2	1	Two dimensional T.4 (MR) coding. EOLs are not byte aligned.
2	5	Two dimensional T.4 (MR) coding. EOLs are byte aligned.
3	0	T.6 (MMR) coding. Uncompressed mode not allowed in the encoding.
3	1	T.6 (MMR) coding. Uncompressed mode allowed in the encoding.
4	0	T.82 (JBIG) coding, using ITU-T T.85 profile
5	0	T.81 (JPEG) coding. Baseline JPEG.
6	0	T.87 (JPEG-LS) coding.
7	0	T.803 (JPEG 2000) Profile 0 coding.
7	1	T.800 (JPEG 2000) Profile 1 coding.
8		T.88 (JBIG2) coding. **Editor's note: Verify profile definitions with JBIG commitee.**
other values	other values	Reserved for ISO use.

B.6.3.2 Bits Per Component box

This box is defined in Part 1 Annex I of this Recommendation | International Standard.

B.6.3.3 Palette box

Box type: ‘pclr’ (X'7063 6C72')
Container: JP2 Header box
Mandatory: No
Quantity: At most one
Location: Anywhere after Image Header box

The palette specified by this box is applied to a single component to convert it into multiple components. The colourspace of the components generated by the palette is then interpreted based on the values of the colour specification boxes in the Component Mapping box. If the JP2 Header box contains a Palette box, then it shall also contain a

Component Mapping box. If the JP2 Header box does not contain a Palette box, then it shall not contain a Component Mapping box. The type of the palettized colour bopx shall be ‘pclr’ (X'7063 6C72'). The contents of the box shall be as follows:

This box is defined in Part 1 Annex I of this Recommendation | International Standard.

B.6.3.4 Component Mapping box

This box is defined in Part 1 Annex I of this Recommendation | International Standard.

B.6.3.5 Channel Definition box

This box is defined in Part 1 Annex I of this Recommendation | International Standard.

B.6.3.6 Colour Specification box

Box type: ‘colr’ (X'636F 6C72')
Container: JP2 Header box
Mandatory: Yes
Quantity: At least one
Location: Anywhere after Image Header box

A JP2 Header box in a JPM file may contain multiple Colour Specification boxes, but must contain at least one. A conforming JPM reader shall ignore all Colour Specification boxes after the first.

The type of a Colour Specification box shall be ‘colr’ (X'636F 6C72'). The contents of a Colour Specification box are as follows:

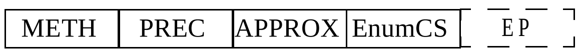


Figure B-23 — Organization of the contents of a Colour Specification box in JPM

METH: Specification method. This field specifies the method used by this Colour Specification box to define the colourspace of the image. This field is encoded as a 1-byte unsigned integer. The value for this field in a JPM file shall be 1, indicating an enumerated colourspace.

PREC: Precedence. This field is reserved for ISO use and the value shall be set to zero; however conforming readers shall ignore the value of the field. 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. The value of this field shall be set to zero.

EnumCS: Enumerated colourspace. This field specifies the colourspace of the image using integer codes. 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 defined in Table B-30. A mask object must use the Grayscale enumerated colorspace.

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 colourspace has been encoded in the JPM file. 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..

Table B-31 — Format of the contents of the Colour Specification box

Field name	Size (bits)	Value
METH	8	1
PREC	8	0
APPROX	8	0
EnumCS	32	See Table B-30
EP	variable	variable

Table B-30 — Legal EnumCS fields for the Colour Specification Box

Value	Meaning
16	**sRGB** as defined in IEC 61966-2
17	**Greyscale**: A greyscale space where image luminance is related to code values using the sRGB non-linearity given in Eqs. (2) through (4) of IEC 61966-2-1 (sRGB) specification: $Y' = Y_{8bit}/255$ B.5 $for(Y' \le 0,04045), Y_{lin} = Y'/12,92$ B.6 $for(Y' > 0,04045), Y_{lin} = \left(\frac{Y' + 0,055}{1,055}\right)^{2,4}$ B.7 where Y_{lin} is the linear image luminance value in the range 0.0 to 1.0. The image luminance values should be interpreted relative to the reference conditions in Section 2 of IEC 61966-2-1.
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 colourspcae, additional Enumerated parameters are specified in the EP field as specified in Annex B.6.3.8
18	**YCbCr(2)**: This is the most commonly used format for image data that was originally captured in RGB (uncalibrated format). The colourspace is based on Recommendation ITU-R BT.601-5. The valid ranges of the YC_bC_r components in this space are [0,255] for Y, and [-128,127] for C_b and C_r (stored with an offset of 128 to convert range to [0,255]). These ranges are different from the ones defined in Recommendation ITU-R BT.601-5. Recommendation ITU-R BT.601-5 specifies a 3x3 matrix transform that can be used to convert these samples into RGB.

Table B-30 — Legal EnumCS fields for the Colour Specification Box

Value	Meaning
3	**sRGB YCC (sRGB-based YCbCr)**: YCbCr which is related to sRGB by Equations B.8, B.9 and B.10. For N bits per component representation, the valid ranges of the YCbCr components in this space is [0, 2^N-1] for Y, and [-2^N-1,2^N-1-1] for Cb and Cr, with Cb and Cr encoded with an offset of 2^N-1 to convert the encoded range to [0, 2^N-1]. The colorimetry corresponding to encoded sRGB YCC values is defined according to PIMA 7667.
other values	Reserved for other ISO uses

B.6.3.7 sRGB Based YCbCr to sRGB Transform

$$R = Y + 1,402 \times C_r \tag{B.8}$$

$$G = Y - 0,34413 \times C_b - 0,71414 \times C_r \tag{B.9}$$

$$B = Y + 1,772 \times C_b \tag{B.10}$$

B.6.3.8 EP field format for the CIELab colourspace

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

RL	OL	RA	OA	RB	OB	IL

Figure B-24 — Organization of the contents of the EP field for the CIELab (EnumCS = 14)

The RL, OL, RA, OA, RB, and OB fields describe how to convert between the unsigned values N_L, N_a, N_b, as defined by ITU-T T.42, that are sent to the compressor or received from the decompressor and the signed CIELab values L*, a*, b* as defined by the CIE. According to ITU-T 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:

$$N_L = \frac{2^{n_L} - 1}{RL} \times L^* + OL \tag{B.11}$$

$$N_a = \frac{2^{n_a} - 1}{RA} \times a^* + OA \tag{B.12}$$

$$N_b = \frac{2^{n_b} - 1}{RB} \times b^* + OB \tag{B.13}$$

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 B.11. It is encoded as a 4-byte big endian unsigned integer.

OL: Offset for L*. This field specifies the OL value from Equation B.11. It is encoded as a 4-byte big endian unsigned integer.

RA: Range for a*. This field specifies the RA value from Equation B.12. It is encoded as a 4-byte big endian unsigned integer.

OA: Offset for a*. This field specifies the OA value from Equation B.12. It is encoded as a 4-byte big endian unsigned integer.

RB: Range for b*. This field specifies the RB value from Equation B.13. It is encoded as a 4-byte big endian unsigned integer.

OB: Offset for b*. This field specifies the OB value from Equation B.13. 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 B-32 — 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 illiminant is represented by the 4 bytes 0x4354 1D4C.

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

Other applications may use other range values by specifyiong 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 encdoing 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 B-33 — Format of the Contents of the EP field for CIELab (EnumCS = 14)

Field name	Size (bits)	Value
RL	32	0 - (2^{32}-1)

Table B-33 — Format of the Contents of the EP field for CIELab (EnumCS = 14)

Field name	Size (bits)	Value
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

B.6.4 Label box

Box type: 'lbl\040' (X'6C62 6C20')

This box contains a textual label that may be associated with a Page Collection, Annex B.1.8, or a Page Annex B.2.2.

The type of a Label box shall be 'lbl\040' (X'6C62 6C20'). The contents of the Label box are as follows:

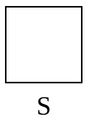


Figure B-25 — Organization of the contents of a Label box

S: Label string. A textual label associated with the entity or entities referred to by the Number List in the same Association box. This value is stored ISO 10646 characters in the UTF-8 encoding. Also, non-printable characters 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.

B.6.5 Cross-Reference box

Box type: 'cref' (X'6372 6566')
Container: See details below
Mandatory: No
Quantity: Any number

If a JPM file contains multiple codestreams or Layout Objects, it may be useful to share header and metadata information to minimize file size. One mechanism to share such data is to use a cross-reference to the actual header or metadata box in place of the actual data. This is done using a Cross-Reference box. A JPM file may contain zero or more Cross-Reference boxes and a a Cross-Reference box shall not point to another Cross-Reference box.The type of the Cross-Reference box shall be 'cref' (X'6372 6566') and it shall have the following contents:

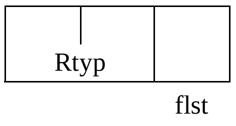


Figure B-26 — Organization of the contents of a Cross-Reference box

Rtyp: Referenced box type. This field specifies the actual type of the box referenced by this Cross-Reference box. This field is encoded as a 4-byte value.

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 entire referenced box, including all box header fields. The format of the Fragment List box is specified in Annex B.5.2.

Table B-34 — Format of the contents of the Cross-Reference box

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

B.6.6 Free box

Box type: 'free' (X'6672 6565')
Container: Any
Mandatory: No
Quantity: Any number

The Free box specifies a section of the JPM file that is not currently used and may be overwritten when editing the file. Boxes containing unused data should be altered to have the box type ‘free’ (X'6672 6565'). Decoders shall ignore Free boxes and their contents need not be preserved when re-writing the file. The contents of a Free box in general are not defined by this Recommendation | International Standard.

Annex C

Metadata

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

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

A JPM may contain metadata boxes with intellectual property right information or vendor specific information, as defined in this Annex. Metadata boxes are optional in a JPM file and may be igored by conforming readers. In addition, a JPM file may contain other metadata boxes not defined in this Annex, but by an external standards organization or industry group.

C.1 Adding intellectual property rights information in JPM

This Recommendation | International Standard specifies a box type for a box that is devoted to carrying intellectual property rights information within a JPM file. Inclusion of this information in a JPM file is optional for conforming files. The definition of the format of the contents of this box is reserved for ISO. However, the type of this box is defined in this Recommendation | International Standard as a means to allow applications to recognize the existence of IPR information. Use and interpretation of this information is beyond the scope of this Recommendation | International Standard.

The type of the Intellectual Property Box shall be ‘jp2i’ (0x6A70 3269).

C.2 Adding vendor specific information to the JPM file format

The following boxes provide a set of tools by which applications can add vendor specific information to the JPM file format. All of the following boxes are optional in conforming files and may be ignored by conforming readers.

C.2.1 XML boxes

An XML box contains vendor specific information (in XML format) other than the information contained within boxes defined by this Recommendation | International Standard. There may be multiple XML boxes within the file, and those boxes may be found anywhere in the file except before the File Type box.

The type of an XML box is ‘xml\040’ (0x786D 6C20). The contents of the box shall be as follows:

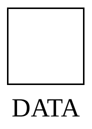


Figure C-1 — Organization of the contents of the XML box

DATA: This field shall contain a well-formed XML document as defined in REC-xml-19980210.

The existence of any XML boxes is optional for conforming files. Also, any XML box shall not contain any information necessary for decoding the image to the extent that is defined within this part of this Recommendation | International Standard, and the correct interpretation of the contents of any XML box shall not change the visual appearance of the image. All readers may ignore any XML box in the file.

C.2.2 UUID boxes

A UUID box contains vendor specific information other than the information contained within boxes defined within this Recommendation | International Standard. There may be multiple UUID boxes within the file, and those boxes may be found anywhere in the file except before the File Type box.

The type of a UUID box shall be ‘uuid’ (0x7575 6964). The contents of the box shall be as follows:

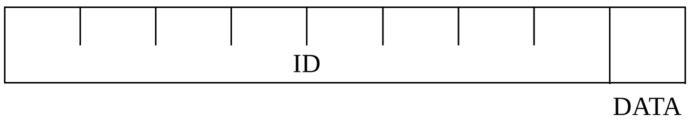


Figure C-2 — Organization of the contents of the UUID box

ID: This field contains a 16-byte UUID as specified by ISO/IEC 11578:1996. The value of this UUID specifies the format of the vendor specific information stored in the DATA field and the interpretation of that information.

DATA: This field contains the vendor specific information. The format of this information is defined outside of the scope of this standard, but is indicated by the value of the UUID field.

Table C-1 — Format of the contents of a UUID box

Field name	Size (bits)	Value
UUID	128	Variable
DATA	Variable	Variable

The existence of any UUID boxes is optional for conforming files. Also, any UUID box shall not contain any information necessary for decoding the image to the extent that is defined within this part of this Recommendation | International Standard, and the interpretation of the information in any UUID box shall not change the visual appearance of the image. All readers may ignore any UUID box.

C.2.3 UUID Info boxes (superbox)

While it is useful to allow vendors to extend JPM files by adding information using UUID boxes, it is also useful to provide information in a standard form which can be used by non-extended applications to get more information about the extensions in the file. This information is contained in UUID Info boxes. A JPM file may contain zero or more UUID Info boxes. These boxes may be found anywhere in the top level of the file (the superbox of a UUID Info box shall be the JPM file itself) except before the File Type box.

These boxes, if present, may not provide a complete index for the UUID’s in the file, may reference UUID’s not used in the file, and possibly may provide multiple references for the same UUID.

The type of a UUID Info box shall be ‘uinf’ (0x7569 6E66). The contents of a UUID Info box are as follows:

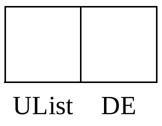


Figure C-3 — Organization of the contents of a UUID Info box

UList: UUID List box. This box contains a list of UUID’s for which this UUID Info box specifies a link to more information. The format of the UUID List box is specified in Annex C.2.3.1.

DE: Data Entry URL box. This box contains a URL. An application can acquire more information about the UUID's contained in the UUID List box. The format of a Data Entry URL box is specified in Annex C.2.3.2.

C.2.3.1 UUID List box

This box contains a list of UUID's. The type of a UUID List box shall be 'ulst' (0x756C 7374). The contents of a UUID List box shall be as follows:

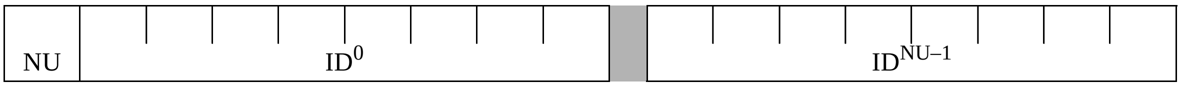


Figure C-4 — Organization of the contents of a UUID List box

NU: Number of UUID's. This field specifies the number of UUID's found in this UUID List box. This field is encoded as a 2-byte big endian unsigned integer.

ID^i: ID. This field specifies one UUID, as specified in ISO/IEC 11578:1996, which shall be associated with the URL contained in the URL box within the same UUID Info box. The number of $UUID^i$ fields shall be the same as the value of the NU field. The value of this field shall be a 16-byte UUID.

Table C-2 — UUID List box contents data structure values

Parameter	Size (bits)	Value
NU	16	0 — (2^{16}–1)
$UUID^i$	128	0 — (2^{128}–1)

C.2.3.2 Data Entry URL box

This box contains a URL that can be used by an application to acquire more information about the associated vendor specific extensions. The format of the information acquired through the use of this URL is not defined in this Recommendation | International Standard. The URL type should be of a service that delivers a file (e.g. URLs of type file, http, ftp, etc.), which ideally also permits random access. Relative URLs are permissible and are relative to the file containing this Data Entry URL box.

The type of a Data Entry URL box shall be 'url\040' (0x7572 6C20). The contents of a Data Entry URL box shall be as follows:

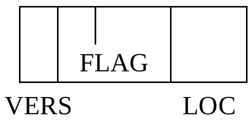


Figure C-5 — Organization of the contents of a Data Entry URL box

VERS: Version number. This field specifies the version number of the format of this box and is encoded as a 1-byte unsigned integer. The value of this field shall be 0.

FLAG: Flags. This field is reserved for other use to flag particular attributes of this box and is encoded as a 3-byte unsigned integer. The value of this field shall be 0.

LOC: Location. This field specifies the URL of the additional information associated with the UUIDs contained in the UUID List box within the same UUID Info superbox. The URL is encoded as a null terminated string of UTF-8 characters

Table C-3 — Data Entry URL box contents data structure values

Parameter	Size (bits)	Value
VERS	8	0
FLAG	24	0
LOC	varies	varies

Annex D

Profiles

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

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

D.1 JPM Profiles

The P field of the Compound Image Header Box, defined in Annex B.1.3, shows the profile to which the JPM file conforms. The profiles define legal values of various fields in a JPM field. This Annex gives the legal values for the P field values specified in the Compound Image Header Box.

D.1.1 Web Profile

The following table gives the legal values for the Web profile of JPM (P=1).

Table D-1 — Web Profile Values

Box	Field	Value	Comment
Compound Image Header	MC	0x04	T.6
		0x10	T.88 (JBIG2 profile)
		0x20	T.800 (JPEG 2000 Part 1)
	IC	0x00	T.81 (JPEG)
		0x20	T.800 (JPEG 2000 Part 1)
Image Header (in Mask Object Box)	BPC	0x01	1 bit per component
		0x02	2 bits per component (only when MC is T.800)
		0x03	3 bits per component (only when MC is T.800)
		0x04	4 bits per component (only when MC is T.800)
		0x08	8 bits per component (only when MC is T.800)
Image Header (in Image Object Box)	BPC	0x01	1 bit per component
		0x02	2 bits per component (only when MC is T.800)
		0x03	3 bits per component (only when MC is T.800)
		0x04	4 bits per component (only when MC is T.800)
		0x08	8 bits per component (only when MC is T.800)

D.1.2 T.44-compatible Profile

The following table gives the legal values for the T.44 compatible profile of JPM (P=2).

In addition, a T.44-compatible JPM file imposes some constraints on the layout objects. In a T.44 Mode 0, there are 3 "layers," called Background, Mask and Foreground. These layers would correspond to the Image object of the first layout object, and the Mask and Image objects of the second layout object, respectively. In this case, the Mask object of the first layout object must be all black. Further, the Mask object of the second layout object must be the same size as the Page. In N-layer T.44-compatible JPM files, additional Mask-Foreground pairs are added, each corresponding to a layout object.

Table D-2 — T.44-Compatible Profile Values

Box	Field	Value	Comment
Compound Image Header	MC	0x01	T.4 (MH)
		0x02	T.4 (MR)
		0x04	T.6 (MMR)
		0x08	T.82 (JBIG)
		0x10	T.88 (JBIG2)
	IC	0x00	T.81 (JPEG)
		0x20	T.800 (JPEG 2000 Part 1)
Image Header (in Mask Object Box)	BPC	0x01	1 bit per component
Image Header (in Image Object Box)	BPC	0x08	8 bits per component
Metadata	-	-	Not allowed

D.2 Compression Profiles

There are also profiles that determine the use of the compression methods allowed in a JPM file. The following table gives the compression profiles that are legal in a JPM file.

Table D-3 — Compression Profile Values

Compression Method	Profile
JPEG 2000	Part 1 & (Level 0 or Level 1) only
JPEG	Baseline JPEG, per ITU-T Rec. T.4, Annex E
JBIG2	ITU-T T.88 Profiles F5, F6 and F7 When using F6 and F7, no mixing of types within a stripe 2 or more stripes per page
JBIG	ITU-T T.85 Application Profile

Annex E

Example JPM Files (informative)

(This Annex forms an informative part of this Recommendation | International Standard.)

This Annex describes use scenarios and then illustrates them with some compliant JPM files.

Editor's note: (E-1) Add some examples showing how to get the codestream for a thumbnail, for an object using fragments, using contiguous codestream. Are there examples of working with fragments in Part 2?

E.1 Use Scenarios

This section describes some common use cases or scenarios for JPM files.

E.1.1 Scanned Files

Scanned files have not been prepared for efficient browsing over a network. They are intended to involve minimal encoder processing effort in order to meet the speed constraints imposed by high-speed scanning.

Scanned files may have metadata specific to the scanning process. This could include scanner metrics such as histograms, skew angle measurements, or cropping rectangle specifications. Other metadata might describe the scanner model or date and time of scanning and so on.

JPM files coming from a scanning subsystem would typically not have been decomposed into layers or layout objects, but may instead include an entire high-quality compressed color or grayscale image suitable for use by a subsequent decomposition system. If the scanning system also is capable of producing a bitonal (1 bit depth) image natively, it may wish to include both this image and the full color image in the JPM file to assist downstream decomposition processing. Two different scanned images of the same content which have not been pre-processed according to the MRC model might be placed into a JPM file as follows: place the color image as a layout object on one page and the bitonal image as a layout object on another page and associate the two pages by placing them together in a page collection. Appropriate Label Boxes could clarify the relationship between the two images and indicate that they represent the same page.

E.1.2 Files prepared for the Internet

JPM files prepared for use on the Internet will typically already have been decomposed into layers and perhaps also into separate layout objects. These preparation steps each give a measure of improved streaming performance in browsing remote JPM files.

Such files would likely have structural information placed near the top of the file, with codestreams occurring later. Codestreams and subsets of codestreams are accessible directly by seeking to a point in the file indicated by entries in the fragment table associated with that codestream.

E.1.3 Examples of Compliant Files

This section details the structure of several different styles of JPM files. It does not give detailed descriptions down to the byte level, but instead gives a hierarchical overview of each style of file.

E.1.3.1 Self-Contained File

This is the simplest type of file. All boxes and data streams are contained within the single file and it stands alone, without support from referenced external files.

Many boxes have a variety of allowed positions within the file. The most logical file organization places all the structural information at the top of the file.

Editor's note: (E-2) Make sure that this is accurate, including the optionals.

```
JPEG 2000 Signature Box
File Type Box
Compound Image Header Box
Data Reference Box [optional]
Metadata (document level) [optional]
Page Collection Box
Page Collection Box [optional]
        Label Box [optional]
        Metadata (outer page collection level) [optional]
        Page Collection Box [optional]
                Label Box [optional]
                Metadata (inner page collection level) [optional]
                Page Label Box [optional]
                        Page Table Box
        Page Collection Box [optional]
Page Box
        Page Header Box
        Label Box [optional]
        BaseColor Box  [optional] [present if not transparent]
        Metadata Box (page level) [optional]
                Layout Object Box
                        Layout Object Header Box
                        Metadata Box (layout object level) [optional]
                        Object Box (for the image object)
                        Object Box (for the mask object) [optional]
                                Object Header Box
                                BaseColor Box
                                JP2 Header Box [optional]
                                        Image Header Box (or cref)
                                        Bits Per Component Box (or cref) [optional]
                                        Color Specification Box
                                        Palette Box (or cref)
                                        [Note on proper ordering required.]
                                        Resolution Box
                Layout Object Box
Fragment Table Box
        Fragment List Box
Media Data Box
        Data chunk:  actual codestream fragment
Media Data Box
Page Box
Fragment Table Box
Page Box
Media Data Box
```

Annex F

XML Encoding of JPM Files (informative)

(This Annex forms an informative part of this Recommendation | International Standard.)

An alternative encoding of a JPM file is defined here. An example is given, element and attribute names corresponding to JPM boxes and parameters are defined and an XML schema is defined.

This encoding may be used in conjunction with XSLT (XML Stylesheet Language Transformations) on a web server to produce alternative JPM files derived from a server JPM file that are tailored to a device, or which contain only specific pages or layout objects.

F.1 XML Encoding Example

When stored in traditional computer file systems, XML-encoded JPM files that conform to this Part should have the file extension .xml (or .XML). On Macintosh file systems, the type code should be ‘xml\040’. The MIME type is xml.

Editor’s note: This needs to be updated.

```
<?xml version="1.0" ?>
<document xmlns="http://www.webimaging.org/schemas/2001/jpm" vers="1.00"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.webimaging.org/schemas/2001/jpm.xsd"
          num-pages="2"
          profile="web"
          ipr="0"
          mask-coders="JBIG JPEG2000"
          image-coders="JPEG JPEG2000">

  <!-- list of URIs to the image data -->
  <data-refs>
    <data-uri id="1">http://www.webimaging.org/images/testmask1.jp2</data-uri>
    <data-uri id="2">http://www.webimaging.org/images/testimage1.jp2</data-uri>
    <data-uri id="3">http://www.webimaging.org/images/testmask2.jbg</data-uri>
    <data-uri id="4">http://www.webimaging.org/images/testimage2.jpg</data-uri>
  </data-refs>

  <!-- locations of image codestream fragments in image files -->
  <fragments num-fragments="5">
    <fragment offset="100" length="20024" data-ref="1">
      <stream obj-id="3" frag-num="0"/>
    </fragment>
    <fragment offset="74776" length="14213" data-ref="1">
      <stream obj-id="3" frag-num="1"/>
    </fragment>
    <fragment offset="20124" length="34652" data-ref="1">
      <stream obj-id="5" frag-num="0"/>
    </fragment>
    <fragment offset="0" length="0" data-ref="2">
      <stream obj-id="2" frag-num="0"/>
    </fragment>
    <fragment offset="0" length="15243" data-ref="4">
```

```
      <stream obj-id="4" frag-num="0"/>
    </fragment>
    <fragment offset="0" length="0" data-ref="3">
      <stream obj-id="5" frag-num="0"/>
    </fragment>
    <fragment offset="15243" length="68423" data-ref="4">
      <stream obj-id="6" frag-num="0"/>
    </fragment>
  </fragments>

  <!-- collection of pages -->
  <pages num-pages="2">
    <page label="Cover" num-layout-objs="1" width="1700" height="2200"
          orientation="rotate0deg" page-color="white">
      <capture-res>
         <horz num="2" denom="1" exp="2"/>
         <vert num="2" denom="1" exp="2"/>
      </capture-res>
      <display-res>
         <horz num="1" denom="1" exp="2"/>
         <vert num="1" denom="1" exp="2"/>
      </display-res>
      <layout-box id="1" width="600" height="900" style="separate-objects">
        <object-box id="1" type="mask" no-image="1" width="600" height="900"
                    h-offset="550" v-offset="400">
          <base-color bpc="2" colorspace="YCbCr">43</base-color>
        </object-box>
        <object-box id="2" type="image" width="600" height="900"
                    h-offset="550" v-offset="400" res-ratio="2"/>
      </layout-box>
    </page>
    <page label="Page 1" num-layout-objs="2" width="1700" height="2200"
          orientation="rotate90deg" page-color="use-base-color">
      <base-color bpc="8" colorspace="sRGB">8388544</base-color>
      <capture-res>
         <horz num="4" denom="1" exp="2"/>
         <vert num="4" denom="1" exp="2"/>
      </capture-res>
      <display-res>
         <horz num="1" denom="1" exp="2"/>
         <vert num="1" denom="1" exp="2"/>
      </display-res>
      <layout-box id="2" width="1500" height="1100" style="separate-objects">
        <object-box id="3" type="mask" width="300" height="900"
                    h-offset="100" v-offset="200"/>
        <object-box id="4" type="image" width="1500"
                    height="800" h-offset="100" v-offset="300"/>
      </layout-box>
      <layout-box id="3" width="1200" height="700" style="separate-objects">
        <object-box id="5" type="mask" width="1200"
                    height="500" h-offset="200" v-offset="1200"/>
        <object-box id="6" type="image" width="1100"
                    height="600" h-offset="250" v-offset="1300"/>
      </layout-box>
    </page>
  </pages>
</document>
```

F.2 Mapping of Boxes and Parameters to Elements and Attributes

This section of this Recommendation | International Standard defines the XML element name that corresponds to any given box in a JPM file.

The top-level element in the XML encoding (document) has no equivalent box in the normal binary encoding of a JPM file since there is no containing superbox at the document level.

Editor's note: This needs to be updated.

Table 1 — JPM Boxes and Parameters and XML Equivalents

JPM Structure		**XML Equivalent**		
Box	**Parameter**	**Element Name**	**Attribute Name**	**Allowed Forms and Example**
--		jpm:document		`<jpm:document>`
	P		profile-id	profile-id="1" means the web profile
	MC		mask-coders	mask-coders="5"
	IC		image-coders	image-coders="4"
Page Box		jpm:page		<jpm:page>
	HEIGHT		height	height="1000"
	WIDTH		width	width="2000"
	RESD		resolution	resolution="200"
Layout Object Box		jpm:layout-box		<jpm:layout-box>
Object Box		jpm:object-box		

F.3 XML Schema for JPM Files

Editor's note: This needs to be updated.

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="http://webimaging.org/schemas/jpm"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:jpm="http://webimaging.org/schemas/jpm">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    An XML schema for the JPEG 2000 Compound Image File Format.
    Created By: David C. Johnson, iLecta, Inc.
    This schema is experimental and subject to change. Send comments to
    davidj@ilecta.com.
    </xsd:documentation>
  </xsd:annotation>

  <!-- global attributes and elements -->
  <xsd:attribute name="id" type="xsd:integer"/>
  <xsd:attribute name="width" type="xsd:positiveInteger"/>
  <xsd:attribute name="height" type="xsd:positiveInteger"/>
  <xsd:attribute name="offset" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="data-ref" type="xsd:integer"/>
  <xsd:element name="base-color" type="jpm:color-type"/>
  <xsd:element name="capture-res" type="jpm:resolution-type"/>
  <xsd:element name="display-res" type="jpm:resolution-type"/>
  <xsd:attribute name="num-pages" type="xsd:positiveInteger"/>

  <!-- color-type specifies a colorspace and color value-->
  <xsd:complexType name="color-type">
    <xsd:simpleContent>
      <xsd:extension base="xsd:integer">
        <xsd:attribute name="bpc" type="xsd:positiveInteger" use="required"/>
        <xsd:attribute name="colorspace" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="sRGB"/>
              <xsd:enumeration value="grayscale"/>
              <xsd:enumeration value="CIELAB"/>
              <xsd:enumeration value="YCbCr"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="ep">
          <xsd:simpleType>
            <xsd:list itemType="integer"/>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!-- resolution-type specifies resolution -->
  <xsd:complexType name="resolution-type">
    <xsd:sequence>
      <xsd:element name="horz" type="jpm:fraction-type"/>
      <xsd:element name="vert" type="jpm:fraction-type"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- fraction-type specifies a fractional value -->
  <xsd:complexType name="fraction-type">
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="num" type="xsd:positiveInteger"
                       use="required"/>
        <xsd:attribute name="denom" type="xsd:positiveInteger"
                       default="1"/>
        <xsd:attribute name="exp" type="xsd:nonNegativeInteger"
                       default="0"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- document contains one or more page collections and metadata -->
  <xsd:element name="document">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="data-refs" type="jpm:data-refs-type"/>
        <xsd:element name="fragments" type="jpm:fragments-type"/>
        <xsd:element name="pages" type="jpm:pages-type"/>
      </xsd:sequence>
      <xsd:attribute ref="jpm:num-pages"/>
      <xsd:attribute name="profile" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="none"/>
            <xsd:enumeration value="web"/>
            <xsd:enumeration value="T.44Compatible"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="mask-coders">
        <xsd:simpleType>
          <xsd:list>
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="MH"/>
                <xsd:enumeration value="MR"/>
                <xsd:enumeration value="MMR"/>
                <xsd:enumeration value="JBIG"/>
                <xsd:enumeration value="JBIG2"/>
                <xsd:enumeration value="JPEG2000"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:list>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="image-coders">
        <xsd:simpleType>
          <xsd:list>
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="JPEG"/>
                <xsd:enumeration value="JBIG"/>
                <xsd:enumeration value="JPEG2000"/>
                <xsd:enumeration value="JPEG-LS"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:list>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="ipr" type="boolean"/>
    </xsd:complexType>
  </xsd:element>

  <!-- data-refs contains a list of URIs containing the actual image data -->
  <xsd:complexType name="data-refs-type">
    <xsd:sequence>
      <xsd:element name="data-uri" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI">
              <xsd:attribute ref="jpm:id"/>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <!-- fragments contains metadata about codestream fragments -->
  <xsd:complexType name="fragments-type">
    <xsd:sequence>
      <xsd:element name="fragment" type="jpm:fragment-type"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="num-fragments" type="xsd:positiveInteger"/>
  </xsd:complexType>


  <!-- fragment contains metadata about a codestream fragment -->
  <xsd:complexType name="fragment-type">
    <xsd:sequence>
      <xsd:element name="stream" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:restriction base="xsd:anyType">
              <xsd:attribute name="obj-id"
                             type="xsd:integer"
                             use="required"/>
              <xsd:attribute name="frag-num"
                             type="xsd:nonNegativeInteger"
                             use="required"/>
            </xsd:restriction>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="jpm:offset"/>
    <xsd:attribute ref="jpm:length"/>
    <xsd:attribute ref="jpm:data-ref"/>
    <xsd:attribute name="num-streams" type="xsd:positiveInteger"/>
  </xsd:complexType>

  <!-- pages contains a collection of pages and metadata -->
  <xsd:complexType name="pages-type">
    <xsd:sequence>
      <xsd:element name="page" type="jpm:page-type" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="jpm:num-pages"/>
  </xsd:complexType>

  <!-- page contains one or layout boxes and metadata -->
  <xsd:complexType name="page-type">
    <xsd:sequence>
      <xsd:element ref="jpm:base-color" minOccurs="0"/>
      <xsd:element ref="jpm:capture-res"/>
      <xsd:element ref="jpm:display-res"/>
      <xsd:element name="layout-box" type="jpm:layout-box-type"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="label" type="xsd:string"/>
    <xsd:attribute ref="jpm:offset"/>
    <xsd:attribute ref="jpm:length"/>
    <xsd:attribute ref="jpm:data-ref"/>
    <xsd:attribute name="num-layout-objs" type="xsd:positiveInteger"/>
    <xsd:attribute ref="jpm:width"/>
    <xsd:attribute ref="jpm:height"/>
    <xsd:attribute name="orientation" default="unspecified">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="unspecified"/>
          <xsd:enumeration value="rotate0deg"/>
          <xsd:enumeration value="rotate90deg"/>
          <xsd:enumeration value="rotate180deg"/>
          <xsd:enumeration value="rotate270deg"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="page-color" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="transparent"/>
          <xsd:enumeration value="white"/>
```

```
          <xsd:enumeration value="black"/>
          <xsd:enumeration value="use-base-color"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <!-- layout-box contains one or more object-boxes and metadata -->
  <xsd:complexType name="layout-box-type">
    <xsd:sequence>
      <xsd:element name="object-box" type="jpm:object-box-type"
                   maxOccurs="2"/>
    </xsd:sequence>
    <xsd:attribute ref="jpm:id" use="required"/>
    <xsd:attribute ref="jpm:width"/>
    <xsd:attribute ref="jpm:height"/>
    <xsd:attribute name="style" default="separate-objects">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="separate-objects"/>
          <xsd:enumeration value="same-object"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <!-- object-box has metadata about an image in a layout-box -->
  <xsd:complexType name="object-box-type">
    <xsd:sequence>
      <xsd:element ref="jpm:base-color" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute ref="jpm:id" use="required"/>
    <xsd:attribute name="type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="mask"/>
          <xsd:enumeration value="image"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="no-image" type="boolean" default="0"/>
    <xsd:attribute name="h-offset" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="v-offset" type="xsd:nonNegativeInteger"/>
    <xsd:attribute ref="jpm:width"/>
    <xsd:attribute ref="jpm:height"/>
    <xsd:attribute name="res-ratio" type="xsd:nonNegativeInteger"/>
    <xsd:attribute ref="jpm:offset"/>
    <xsd:attribute ref="jpm:length"/>
    <xsd:attribute ref="jpm:data-ref"/>
  </xsd:complexType>
</xsd:schema>
```

Annex G
Guidelines for Constructing URLs for JPM Files (informative)

(This Annex forms an informative part of this Recommendation | International Standard.)

This informative Annex describes how to construct URLs that reference sub-elements of a JPM file. These sub-elements may be pages, images or metadata boxes. These conventions may be used when creating links within a JPM file or when linking to an interior element of a JPM file from a web page. When a decoding application receives such a request, it should render only the requested portion of a JPM file.

These URLs are constructed by appending a question mark ("?") character to the path and file name of the JPM file and then appending descriptors, separated by space characters ("%20"), that identify the sub-element for which access is desired.

G.1 Pages and Layout Objects

This example shows how to reference a specific page in a JPM file via its PageID. The PageID occurs in the Page Header box of the desired page. It also shows how to reference a single layout object on a given page via its LObjID.

foo.jpm?page=3%20obj=32

page=3 is a page reference
obj=32 is a layout object reference to a layout object on page 3

If the "page=" string and/or the "obj=" string are omitted, the first parameter is presumed to be the PageID and the second is presumed to be the LObjID.

foo.jpm?17%2031
foo.jpm?17

17 refers to page having a PageID of 17
31 refers to layout object on that page having a LObjID of 31

G.2 Metadata boxes

Metadata boxes may be referred to in a URL containing a JPM file. In such a case, the XML metadata from that level of the JPM file (whether page level, page collection level, or layout object level) is returned to the requester of the URL rather than any image data.

foo.jpm?page=43%20type=meta

In cases where more than one metadata box occur at the same level of a JPM file, an index parameter is used to select which metadata box is to be used. "index=1" refers to the first such metadata box. If no "index=" parameter is used, yet multiple metadata boxes exist at that level, the first metadata box is used.

foo.jpm?page=77%20obj=19%20type=meta%20index=1

If the metadata box is an XML box, the metadata is of MIME type XML. If the metadata box is a UUID box, the type may be or may not be XML. The "mtype=" parameter can be used to identify the type of the data using standard MIME type identification strings. If no "mtype=" parameter is used, the mtype is presumed to be XML.

foo.jpm?page=91%20type=meta%20mtype=text

G.3 Labels

Label boxes may be used in a JPM file to attach labels to other boxes in the file by means of an Association box. Label boxes may be used to label metadata boxes. In such a case, the metadata from that box is returned to the requester of the URL rather than any image data. XHTML pages may occur inside an XML box, since they represent valid XML data. If nothing but metadata occurs inside the Association box with the Label box, then the "type=meta" parameter may be omitted.

foo.jpm?label=album.html
foo.jpm?label=IPR-info%20type=meta%20index=2

G.4 Page Collections

Page Collection boxes may contain an optional Label box. This label may be used to refer to a page collection in a URL. Every JPM file has a primary page collection. Even if it contains no label, the primary page collection may be referred to by the reserved label "primary". If the "coll=" parameter is omitted, then any page referred to is presumed to be locatable in the primary page collection. The behavior of the decoder here is undefined. It may wish to provide a textual list of page labels of the pages in the collection, or it may wish to show the page thumbnails of the pages in the collection, or both.

foo.jpm?coll=toc
foo.jpm?coll=primary
foo.jpm?coll=list-of-figs
foo.jpm?coll=list-of-tables
foo.jpm?coll=toc%20page=31
foo.jpm?coll=primary%20page=7
foo.jpm?page=7
foo.jpm?

G.5 Page Thumbnails

Each page may have an optional thumbnail. The "type=thumb" parameter value is used to specify the thumbnail for a page rather than the page itself. The "type=imag" parameter value is used to specify the page itself is to be rendered, rather than the thumbnail.

foo.jpm?page=3%20type=img
foo.jpm?page=3%20type=thumb

G.6 Document Thumbnail

Every JPM file may have at most one optional document thumbnail. The reserved page label "thumb" gets and displays the document thumbnail even if no Label box is used to label the document thumbnail image.

foo.jpm?page=thumb
foo.jpm?thumb

G.7 Byte Ranges

A byte range within a JPM file may be specified for processing by the decoder. The behavior of the decoder then depends on what box types are located within the byte range. The "off=" parameter is used to specify an offset in bytes from the front of the file. The "len=" parameter is used to specify a number of bytes to be returned. This is not a very reliable method of getting to sub-elements of a JPM file, since they may be moved and re-arranged as the file grows or changes. Better methods are provided above for this purpose. If, however, the decoder has already received enough information from the file in question to know where a sub-element of interest is located, then this is a useful method. Ths method may also be useful to retrieve a small amount of data from the front of the file in order for the decoder to better understand the organization of the file.

foo.jpm?off=12384%20len=1024