INTERNATIONAL STANDARD

**ISO/IEC 15444-6**

First edition
2003-10-15

# Information technology — JPEG 2000 image coding system —

## Part 6: Compound image file format

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

*Partie 6: Format de fichier d'image de composant*



Reference number
ISO/IEC 15444-6:2003(E)

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

Published in Switzerland

 

# Contents

# Foreword

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

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

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

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

ISO/IEC 15444-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

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

— *Part 1: Core coding system*

— *Part 2: Extensions*

— *Part 3: Motion JPEG 2000*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Compound image file format*

— *Part 8: Secure JPEG 2000*

# Information technology — JPEG 2000 image coding system —

# Part 6:
# Compound image file format

## 1 Scope

This 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 ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I and uses boxes defined for both the JP2 file format and the JPX file format defined in ITU-T Rec T.801 | ISO/IEC 15444-2 Annex M. This standard is useful for applications storing multiple pages, images with mixed content, and/or images that need more structure than provided in JP2. Applications that implement this file format shall implement it as described in this International Standard.

This 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,
- specifies a mechanism for storing JPEG 2000 and other compressed image data formats,
- specifies a mechanism by which metadata can be included in files specified by this International Standard.

## 2 Normative references

### 2.1 Identical Recommendations | International Standards

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

- ISO 5807:1985, *Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*
- ISO 8601:2000, *Data elements and interchange formats - Information interchange - Representation of dates and times*
- ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange*
- ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*
- ISO/IEC 11578:1996, *Information technology - Open Systems Interconnection - Remote Procedure Call (RPC)*
- 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*

- 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.42, *Continuous-tone colour representation method for facsimile*
- ITU-T Recommendation T.44 | ISO/IEC 16485:2000, *Information technology - Mixed Raster Content (MRC)*
- ITU-T Recommendation T.44 Amendment 1, *Accommodation of new Annex B*
- ITU-T Recommendation T.45, *Run-length colour encoding*
- 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 11544:1993, *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 10918-3:1997, *Information technology - Digital compression and coding of continuous-tone still images: Extensions*
- ITU-T Recommendation T.84 Amendment 1 | ISO/IEC 10918-3:1997/Amd 1:1999, *Provisions to allow registration of new compression types and versions in the SPIFF header*
- ITU-T Recommendation T.86 | ISO/IEC 10918-4:1999, *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 14495-1:2000, *Information technology - Lossless and near-lossless compression of continuous-tone still images - Baseline*
- ITU-T Recommendation T.88 | ISO/IEC 14492:2001, *Information technology - Lossy/lossless coding of bi-level images*
- ITU-T Recommendation T.89, *Application profiles for Recommendation T.88 - Lossy/lossless coding of bi-level images (JBIG2) for facsimile*
- ITU-T Recommendation T.800 | ISO/IEC 15444-1:2000, *Information technology - JPEG 2000 image coding system - Part 1: Core coding system*
- ITU-T Recommendation T.801 | ISO/IEC 15444-2, *Information technology - JPEG 2000 image coding system - Part 2: Extensions*
- ITU-T Recommendation T.803 | ISO/IEC 15444-4, *Information technology - JPEG 2000 image coding system - Part 4: Conformance testing*

## 2.2 Additional References

- IEEE Std. 754-1985 R1990, *IEEE Standard for Binary Floating-Point Arithmetic*
- IETF RFC 1766, *Tags for the Identification of Languages*, March 1995
- IETF RFC 2279, UTF–8, *A transformation format of ISO 10646*, January 1998
- ICC.1:1998-09, *International Color Consortium, File Format for Color Profiles*
- IEC 61966-2-1:1999-10, *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB*

— IEC 61966-2-1/Amd.1: *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB*

— W3C, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation 6 October 2000, <http://www.w3.org/TR/REC-xml>

## 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ITU-T Rec T.800 | ISO/IEC 15444-1 Clause 3 and the following apply.

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

**3.2 BasePage**: The original state of the page before it is rendered with layout objects.

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

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

**3.5 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.

**3.6 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.

**3.7 fragment**: A portion of the codestream for an image. Clause 5.2.6 describes fragment usage.

**3.8 JP2 file**: The name of a file in the file format described in ITU-T Rec T.800 | ISO/IEC 15444-1. Structurally, a JP2 file is a contiguous sequence of boxes.

**3.9 JPM file**: The name of a file in the file format described in this 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.

**3.10 JPX file:** The name of a file in the file format described in ITU-T Rec T.801 | ISO/IEC 15444-2. Structurally, a JPX file is a contiguous sequence of boxes.

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

**3.12 main page collection**: The main page collection contains all pages and page collections in a file.

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

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

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

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

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

**3.18 page collection**: A collection of pages logically grouped together in a JPM file. Each page must be contained in at least one page collection.

**3.19 primary page collection**: A page collection which provides back and forward navigation in the main document associated with a page.

**3.20 PageImage**: The image created by rendering the BasePage with the layout objects. The $PageImage_k$ is the images created by rendering the BasePage with the first k layout objects.

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

**3.22 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 International Standard, the following abbreviations apply. The abbreviations defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Clause 4 also apply to this International Standard

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

**DPI**: Dots per inch

**IPR**: Intellectual Property Rights

**JP2**: JPEG 2000 File Format defined in ITU-T Rec T.800 | ISO/IEC 15444-1

**JPX**: JPEG 2000 File Format defined in ITU-T Rec T.801 | ISO/IEC 15444-2; JPEG 2000 File Format E tended

**JPM**: JPEG 2000 File Format defined in this International Standard; JPEG 2000 File Format - Multi-layer

**MRC**: Mixed Raster Content

**UUID**: Universal Unique Identifier

## 5 General Description

The purpose of this clause is to give an overview of this International Standard. Terms defined in previous clauses in this International Standard will also be introduced. (Terms defined in Clause 3 and 4 in ITU-T Rec T.800 | ISO/IEC 15444-1 continue to apply in this International Standard.) Throughout this International Standard, text formatted as a NOTE in the following form is informative only:

NOTE - Informative text appears here.

This International Standard defines a file format for storing compound images using the JPEG 2000 file format family architecture. A compound image file contains multiple images, both contiguous tone and bi-level, together with composition models describing how the individual images are combined to generate the compound image. This International Standard is based on the multi-layer Mixed Raster Content (MRC) imaging model, defined in ITU-T T.44 | ISO/IEC 16485.

This International Standard defines a member of the JPEG 2000 file format family that enables the 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 using multiple layers, 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) content usually associated with naturally occurring images; bi-level detail associated with text and line-art; and multi-level colours associated with the text and line-art. This International Standard makes provisions for processing, interchange, and archiving of these image types in multiple layers and defines composition models which regenerate the desired image.

### 5.1 Mixed Raster Content Model

A file that conforms with this International Standard contains one or more pages. The *PageImage* associated with a page is generated by combining the page's layout objects with the *BasePage*.

The *BasePage* is the initial *PageImage* before any layout objects have been rendered. The *BasePage* has the same width and height as the page and is either transparent or filled with a single colour. *BasePage* is defined in 5.2.3.1. The Layout Objects are applied sequentially, in an order defined by the Layout Object Identifier field

in the Layout Object Header boxes, to the *BasePage* to create the final *PageImage*. The Layout Object with Layout Object Identifier value of 0 is the page thumbnail and is not used in creating the *PageImage*.

Associated with each Layout Object is a mask *M* and an image *I*. The mask *M* is an opacity image and has only one component; *I* can be greyscale or colour, with one or more components. *M* and *I* are defined in Clause 5.2.4. Both *M* and *I* have the same width and height as the page.

The following equations show the model for combining the *BasePage* and a sequence of *n* layout objects with non-zero Layout Object Identifier values to create the final *PageImage*. We will use the notation *N[c][x,y]* to refer to the sample value at position *(x,y)* in component *c* of an image *N*.

$$\mathrm{PageImage}_0[c][x,y] = \mathrm{BasePage}[c][x,y] \tag{1}$$

$$\mathrm{PageImage}_m[c][x,y] = \frac{(s_m - M_m[0][x,y]) \times \mathrm{PageImage}_{m-1}[c][x,y] + M_m[0][x,y] \times I_m[c][x,y]}{S_m} \quad m = 1, \ldots, n \tag{2}$$

$$\mathrm{PageImage}[c][x,y] = \mathrm{PageImage}_n[c][x,y] \tag{3}$$

where $M_m[c][x,y]$ and $I_m[c][x,y]$ are the image sample values of component *c* at position *(x,y)* of the $m^{th}$ Layout Object's mask *M* and image *I* respectively, and $s_m = 2^{\text{bit depth of } M_m} - 1$. *M*, *I* and *BasePage* are defined in 5.2.4 and 5.2.3.1. *PageImage*[c][x,y] is the final page image, obtained after combining all *n* layout objects associated with the page.

NOTE - Figure 1 shows a simple example of a *PageImage* constructed from a *BasePage* with a single solid colour and two Layout Objects. Note that white in the masks $M_0$ and $M_1$ denotes a value of 0.

NOTE - In a JPM file, the mask and image objects in a layout object typically have different spatial resolutions. Therefore, they must be scaled to the same resolution and to the page resolution before they are combined to create a page image according to Equations 1-3. In a JPM file, the size of the mask, image and page are specified in page grid units, but their resolutions may not be specified. The scaling of the mask and image is specified by a scale factor. The method of scaling is not specified in this International Standard, although informative guidelines for scaling of the mask and image are provided. The page image would then be scaled to the resolution of the output device for rendering on a display or printer. As described here, there would be two separate scaling operations: in the first, the mask and image of successive layout objects are scaled to a common resolution and combined on the page; in the second, the resulting page image is scaled to the device resolution. In practice, these two scaling operations are often combined into a single, device-dependent and implementation-specific scaling operation.

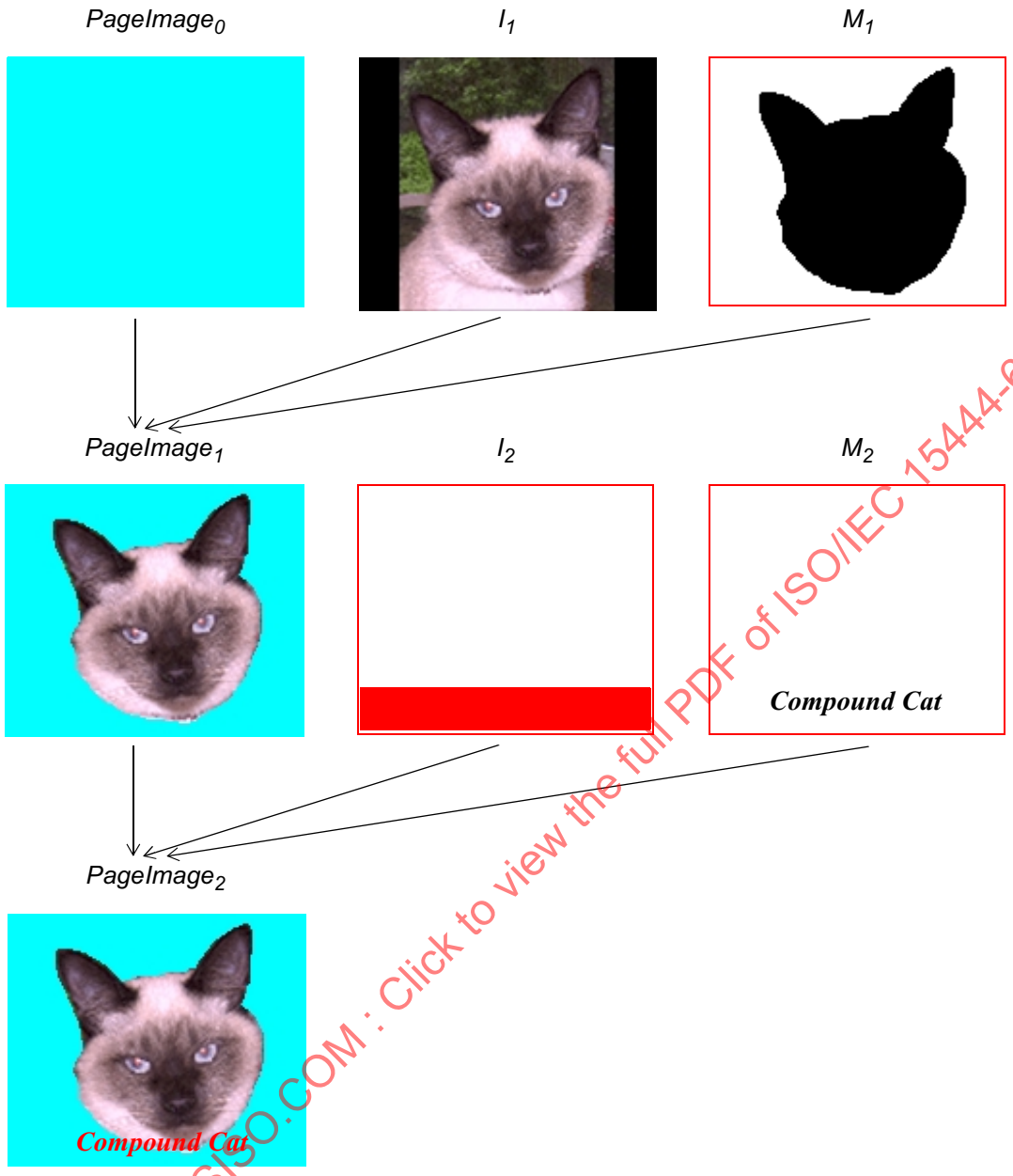


**Figure 1 — Example compound document (Informative)**



## 5.2 File Elements and Structure

The files that conform to the format defined in this 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 equations 1-3. The key elements or boxes of a JPM file are: page collections, page, layout object, and object. An object points to its image data directly via a Contiguous Codestream box or indirectly via a Fragment Table box. 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. A Compound Image Header Box, containing general information about the file, is then followed by Page Collection, Page, Fragment Table, Contiguous Codestream, Media Data and general metadata boxes. Refer to Annex B for constraints on the location of boxes in a JPM file.

This list illustrates the hierarchical relationship between the key elements in a JPM file. A particular order of these boxes is not implied. Full details of all the boxes may be found in Annex B.

JPEG 2000 Signature
File Type
Compound Image Header
Page Collection
Page
    Layout Object
        Mask Object
        Image Object
Fragment Table
Codestream Fragment
Contiguous Codestream

### 5.2.1 Page Collections

A JPM compatible file consists of a sequence of pages, each represented by a Page box which occurs at the top level of the file and each of which can be rendered independently of any other page. Page collections are used to logically group pages in a JPM file. Page collections can be logically nested, so that a page collection can itself consist of one or more page collections and/or one or more pages. Page collections referred to from other page collections are called subsidiary page collections. All pages in a JPM file must be pointed to by at least one page collection.

A page can be said to be contained in a page collection, but this does not mean that the Page box is located within the Page Collection box. It is not. Page boxes and Page Collection boxes both occur at the top level of the file and are not contained within other boxes.

A JPM file contains one page collection known as the main page collection which is used to locate all pages of the file. Any additional page collections in a JPM file are logically nested within the main page collection (see Clause 5.2.2.3). A Page Collection box contains an optional Label box, optional metadata (XML and/or UUID) boxes, and a Page Table box that contains the locations of the pages and page collections belonging to the page collection.

It is recommended that optimized files have the main page collection located near the beginning of the file. While Page boxes and Page Collection boxes occur at the top level of the file, they may be located in external files. This case may be viewed as equivalent to being at the top level of the file.

Each page in a JPM file has a primary page collection. The purpose of a primary page collection is to enable navigation in the primary document to which the page belongs using the sequential order within the primary page collection, thus providing support for previous page and next page commands.

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

By walking the tree of pages and logically nested page collections in a page's primary page collection, all pages in the page's primary document can be reached. Every page (with the exception of a self-contained JPM file containing only a single page) has a Primary Page Collection Locator or PPCLoc box. This box points to the primary page collection of the page and provides an index, PIx, into that page collection's page table where the page is referenced. Then the next page and previous page can be found by walking one page forward or backward from the current page.

NOTE - 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 point to only the first pages of sections or chapters. Any page collections of this sort must be auxiliary page collections, since they provide redundant pointers to pages and page collections and are sparse rather than comprehensive (see Clause 5.2.2.3).

NOTE - Figure 3 illustrates a logical grouping of page collections PC and pages P in a JPM file. PCa is the main page collection and the primary page collection for pages P0, P8 and P9 and page collections PCb, PCc and PCe. In a JPM file, the Page Table box of the Page Collection box for PCa would reference the Page boxes for P0, P8 and P9, and the Page Collection boxes for PCb, PCc and PCe. The Primary Page Collection Locator boxes in these Page and Page Collection boxes would reference the Page Collection box for PCa.

PCb is a subsidiary page collection of PCa and the primary page collection for pages P1, P6 and P7 and for page collection PCd. PCd is the primary page collection for pages P2, P3, P4 and P5. PCc is the primary page collection for pages P10 and P11.

PCe is an auxiliary page collection, which references page collection PCc and page P5.

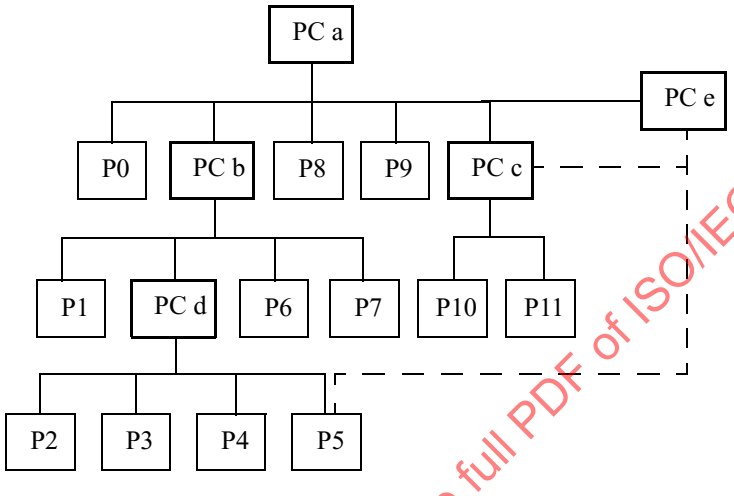


**Figure 2 — Example of page collections and pages (Informative)**



#### 5.2.2.1 Main Page Collection

Each JPM file has one main page collection. The purpose of the file's main page collection is to comprehensively list all pages (and subsidiary page collections) within a JPM file. This allows random seeking to any of the pages in the file.

The Page boxes for these pages occur at the top level of the file format, as do Media Data boxes containing fragments of codestreams. Since these Media Data boxes may be large and since they would likely be interspersed with Page boxes, the Page boxes might be widely separated in the file. In a JPM file containing a client copy of several browsed pages of a server's JPM file, each successively viewed page and some portion of its codestream fragments would be appended in turn to the bottom of the file. An update to the main page collection allows each of these new pages to be located.

The main page collection comprehensively references all pages and page collections by means of a hierarchical arrangement. Some pages may be referenced from a page collection beneath the main page collection, but all pages and page collections are part of the tree structure of the main page collection.

A JPM file optimized for browsing would have the main page collection near the front of the file. On the other hand, a client copy created during a browsing session would likely have the main page collection appended to the end of the file each time it is modified. The old main page collection's Page Collection box could then be left in place but have its box type changed from "pcol" to "free". A later garbage collection step could delete these Free boxes. Because of cases like this, the file format has a pointer to the main page collection included in the Compound Image Header box near the top of the file. This makes it easy to locate the main Page Collection box.

#### 5.2.2.2 Primary Page Collection

Each page or page collection has a primary page collection. By way of distinction, each JPM file has a main page collection. A primary page collection is a property of a page or page collection, not a property of a JPM file.

The primary page collection of a page is the page collection where a JPM reading application would find the “next page” and “previous page” for a current page.

A Primary Page Collection Locator or PPCLoc box must appear in all Page boxes and all Page Collection boxes, with the one exception detailed in the next paragraph. The PPCLoc box provides a pointer to the primary page collection of the page or page collection. This backward pointer enables a comprehensive tree walk to find all the pages and page collections in the file.

The PPCLoc box is optional only in the case of a single-page, self-contained JPM file (i.e. one which has no external references). In this case, it appears that the page has no primary page collection, but in fact the file’s main page collection functions as the page’s primary page collection.

The primary page collection of a page or page collection may not be in the local JPM file in which that Page box or Page Collection box is located. The local file may, for example, have three single pages, each of which has been copied from one of three different remote files. Keeping the original primary page collection pointers for these pages pointing to the original remote files allows a user to keep browsing the original source document via next page and previous page commands.

The main page collection for a file may be a copy of a subsidiary page collection on a remote server, for instance, in which case it would have the parent page collection on that remote server as its primary page collection. When the main page collection does not have a primary page collection, then the main Page Collection Box shall not contain a PPCLoc box.

#### 5.2.2.3 Auxiliary Page Collections

Auxiliary page collections are page collections which provide redundant pointers to pages already pointed to in the logical tree structure of the main page collection. Examples of auxiliary page collections could include a List of Figures or a List of Tables. Auxiliary page collections still appear in the logical tree structure of the main page collection so that they can be located, but a flag is used to mark them as auxiliary. This way, a decoding application knows they are not to be used to perform a comprehensive tree walk through all the pages referenced in the main page collection.

Auxiliary page collections appear in the main page collection to assist an application in locating them within the file, but they are not part of the logical tree that is walked to comprehensively locate all pages. They instead provide a redundant means of reaching selected pages and thus should be ignored when trying to determine the natural page order of the file. Auxiliary page collections can appear down in the hierarchy of the main page collection; they need not occur at the top level.

Auxiliary page collections should be labeled by means of a Label box in order to be useful to a receiving application. If they are labeled, then a decoding application could present the label to the end user and offer such options as “next page in List of Figures” and “previous page in List of Figures.”

As an example, if page 17 appears in the “List of Figures” page collection, the decoding application would return to that page collection to get the next or previous page containing a figure, but would return to the primary page collection for page 17 (by means of the PPCLoc box in page 17’s Page box) to find the next page or previous page.

### 5.2.3 Pages

A page in a JPM file is represented by a Page box, a superbox that consists of a Page Header box, containing general information about the page, a Page Collection Locator box, containing the location of the page’s primary page collection, an optional Base Colour box, which describes the base page colour, optional Metadata

boxes, and Layout Object boxes, one for each layout object on the page. Full details of these boxes may be found in Annex B.2.

Pages occur at the top level of the file format. This means incremental updates to the file may be accomplished by simply appending new pages to the end of the file. Page boxes and boxes containing codestreams or portions of codestreams may be intermixed in the file.

With the exception of document thumbnails, each image in a JPM files is logically associated with at least one page.

#### 5.2.3.1 Base Page

The *BasePage* is the initial *PageImage* before any layout objects have been rendered. Let *page_width* and *page_height* be the width and height of the page respectively, as signalled in the Page Header box.

Let *spc* be the colourspace in which the *I* images are to be combined to generate a *PageImage*. *BasePage* is an image with dimensions *page_width* and *page_height* and colourspace *spc*.

If the PColour field of Page Header box is 0 then the *BasePage* is transparent and contains the sample values of any underlying image, converted to the colourspace *spc*.

If the PColour field of the Page Header box is 1 then every *BasePage* sample contains the representation of *white* in the colourspace *spc*.

If the PColour field of the Page Header box is 2 then every *BasePage* sample contains the representation of *black* in the colourspace *spc*.

If the PColour field of the Page Header box is 255 then there is a mandatory Base Colour box within the Page box and every *BasePage* sample contains the *spc* representation of the colour indicated by the Base Colour box.

### 5.2.4 Layout Objects

Within a Page Box, there are as many Layout Object boxes as there are layout objects on the page. A Layout Object box is a superbox that consists of a Layout Object Header box, containing general information about the layout object, optional boxes containing metadata associated with the layout object, and one or two Object boxes - either an image Object box and/or mask Object box, or a combined image/mask Object box.

Each Object box contains an Object Header box identifying whether the object represents an image, a mask or a combined image/mask, specifying the location of any codestream associated with the object and containing positioning information for the object.

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, then it must have a defined image base colour. A mask object or image/mask object, if present, must have a codestream associated with it and have NoCodestream=0.

If an object has an associated codestream then the Object Header box is followed by an optional Object Scale box and a JP2 Header box containing boxes describing the object data: an Image Header, Colour Specification, optional Bits Per Component, Palette, Component Mapping and Channel Definition boxes.

Associated with each layout object is a single component opacity mask *M* and an image *I*, each with the same width and height as the containing page.

The *I* images for the layout objects to be combined to generate a *PageImage* must all use the same colourspace and bit-depth. The colourspace to be used for the *I* images of layout objects may be decided by the implementor and is not defined by this International Standard.

The general methods for generating the mask *M* and the image *I* associated with a layout object are described in 5.2.4.2 and 5.2.4.4.

5.2.4.1 describes the special case of generating a mask *M* and an image *I* from a JBIG 2 codestream with an associated ITU-T Rec. 45 encoding of colour tags.

#### 5.2.4.1 Colour Tagged JBIG 2 Layout Objects

If the Layout Object box contains an image Object box with a compression type of ITU-T Rec. T.45 (Run Length) coding then it must also contain a mask Object box with a compression type of ITU-T Rec. T.88 (JBIG 2), and both must have the NoCodestream field in the Object Header boxes set to 0. In addition, the following fields must be the same for the image and mask Objects:

- the OVoff and OHoff fields in the Object Header boxes;
- the VRN, VRD, HRN, VRD fields in the Scale boxes;
- the HEIGHT and WIDTH fields in the Image Header box in the JP2 Header boxes.

In 5.2.4.3.1, *m_orig* is the image obtained by decompressing the JBIG 2 codestream associated with the mask Object box.

In 5.2.4.4.1, *i_orig* is the image obtained by applying the colour tags associated with the image Object box to the symbol occurrences in the JBIG 2 mask codestream as described in ITU-T Rec. T.45.

#### 5.2.4.2 Mask *M* for a Layout Object

If the Layout Object box does not contain a mask Object box or a combined image/mask Object box, then the mask *M* for the layout object is defined to have a bit-depth of 1 and to have value 1 at all sample locations, i.e. *M[0][x,y] = 1*.

Sub-clauses 5.2.4.3.1 - 5.2.4.3.5 define *M* when a codestream is associated with a mask Object box or a combined image/mask Object box in the Layout Object box.

The first stage in generating *M*, described in 5.2.4.3.1, is to decode the mask object, converting to a bit-depth of 8 if the mask has a lower bit-depth, and ensuring that the samples use a “min is white” representation.

The second stage, described in 5.2.4.3.2, is to scale the mask.

The third stage, described in 5.2.4.3.3, is to clip the mask from the top and from the left.

The fourth stage, described in 5.2.4.3.4, is to position the mask on the page.

The fifth and final stage, described in 5.2.4.3.5, is to clip the mask to the layout window.

These stages are described individually for clarity and an efficient JPM decoder may be able to combine one or more of these stages.

Figure 3 illustrates the five stages for an example mask. Note that white in the images denotes a value of 0.

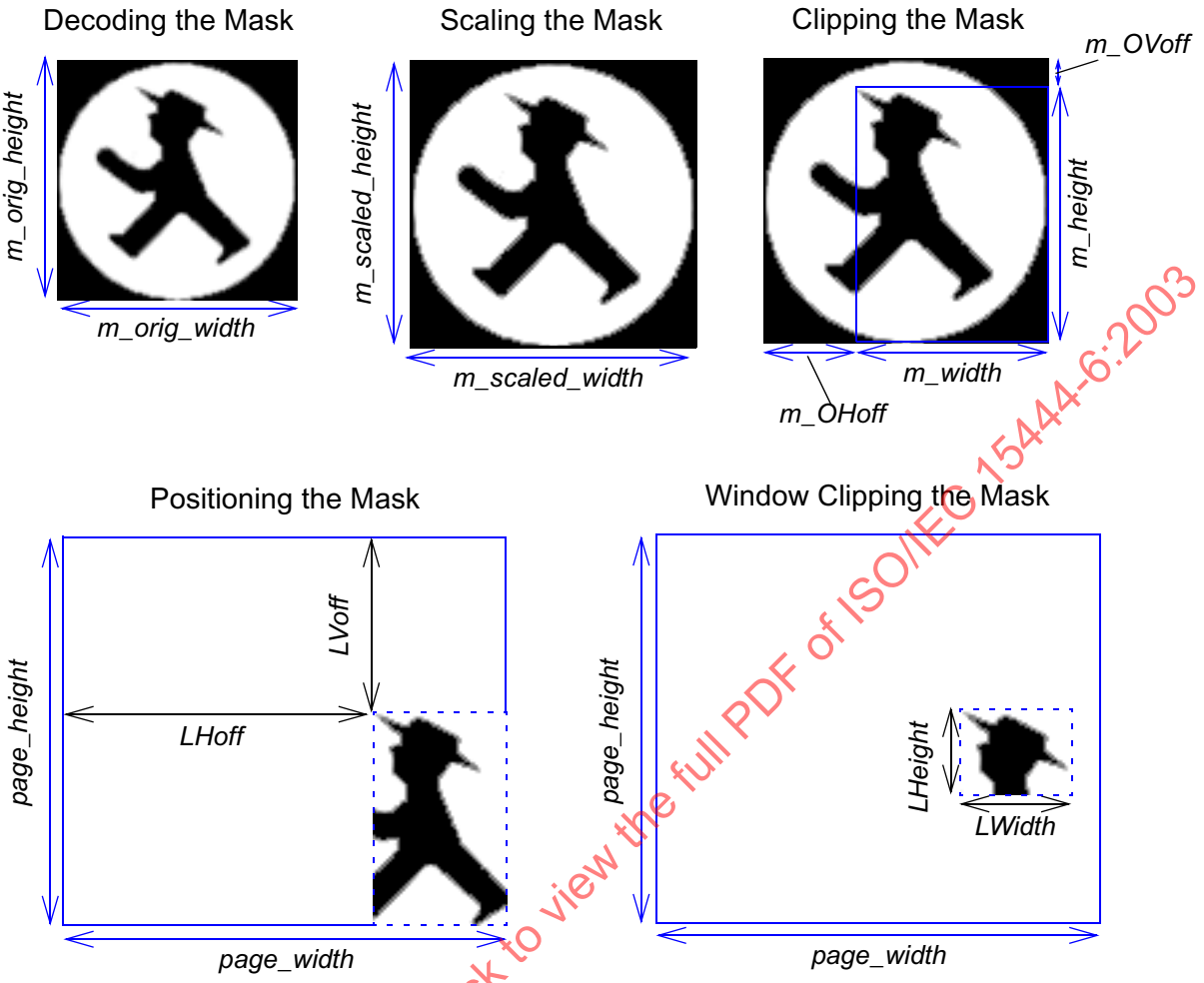


**Figure 3 — Example of generating M (Informative)**



#### 5.2.4.3.1 Decoding the Mask

If a mask Object box is used then let *m_orig* be the image obtained by decompressing the associated codestream, otherwise let *m_orig* be the image containing only the opacity component of the image obtained by decompressing the associated codestream. Let *m_orig_width* and *m_orig_height* be the width and height of *m_orig* and let *b* be the bit-depth of *m_orig*.

If *b* is 1 then, unless indicated otherwise by the codestream, samples of value 0 are assumed to represent white while samples of value 1 are assumed to represent black.

If *b* is greater than 1 then, unless indicated otherwise by the codestream, samples of value 0 are assumed to represent black while samples of value $2^b - 1$ are assumed to represent white.

Let *grey* be an image with bit-depth *g* = *max(8, b)* and the same width and height as *m_orig*. *grey*, defined as follows:

$$\text{grey}[0][x, y] = \begin{cases} (2^b - 1 - \text{m\_orig}[0][x,y]) \times \dfrac{2^g - 1}{2^b - 1} & \text{if 0 represents black in m\_orig} \\ \text{m\_orig}[0][x,y] \times \dfrac{2^g - 1}{2^b - 1} & \text{otherwise} \end{cases} \quad 4)$$

#### 5.2.4.3.2 Scaling the Mask

If an Object Scale box is contained in the mask Object box or combined image/mask Object box then let *m_VRN*, *m_VRD*, *m_HRN* and *m_HRD* be the scaling factors contained in this box, otherwise let *m_VRN*, *m_VRD*, *m_HRN* and *m_HRD* all be 1. Let *m_scaled* be the result of scaling *grey* vertically by the ratio $\frac{\text{m\_VRN}}{\text{m\_VRD}}$ and horizontally by the ratio $\frac{\text{m\_HRN}}{\text{m\_HRD}}$, generating an image with bit-depth *g* and dimensions:

$$\text{m\_scaled\_width} = \left\lfloor \frac{\text{m\_HRN}}{\text{m\_HRD}} \times \text{m\_orig\_width} \right\rfloor \text{ and } \text{m\_scaled\_height} = \left\lfloor \frac{\text{m\_VRN}}{\text{m\_VRD}} \times \text{m\_orig\_height} \right\rfloor . \quad (5)$$

NOTE - The method for scaling a binary or grey mask up or down is implementation specific. For example, the scaling may use nearest neighbour or bilinear interpolation.

#### 5.2.4.3.3 Clipping the Mask

Let *m_OVoff* and *m_OHoff* be the vertical and horizontal offsets in the mask Object box or combined image/mask Object box.

Let *m_clipped* be an image with bit-depth *g*, width *m_width = max(0, m_scaled_width - m_OHoff)* and *m_height = max(0, m_scaled_height - m_OVoff)*. *m_clipped* is defined as follows:

$$\text{m\_clipped}[0][x,y] = \text{m\_scaled}[0][x + \text{m\_OHoff}, y + \text{m\_OVoff}] \quad (6)$$

#### 5.2.4.3.4 Positioning the Mask

Let *page_width* and *page_height* be the width of the page containing the layout object, as signalled in the Page Header box. Let *LVoff* and *LHoff* be the layout vertical and horizontal offsets in the Layout Object Header box.

Let *m_pos* be an image with a width of *page_width* and height of *page_height*, defined as follows:

$$\text{m\_pos}[0][x, y] = \begin{cases} \text{m\_clipped}[0][x - \text{LHoff}, y - \text{LVoff}] & (0 \le (x - \text{LHoff}) < \text{m\_width}) \text{ and } (0 \le (y - \text{LVoff}) < \text{m\_height}) \\ 2^g - 1 & \text{otherwise} \end{cases} \quad (7)$$

#### 5.2.4.3.5 Window Clipping the Mask

Let *M* be an image with a width of *page_width* and height of *page height*. Let *LWidth* and *LHeight* be the layout object width and layout object height as specified in the Layout Object Header box. *M* is defined as follows:

$$M[0][x, y] = \begin{cases} \text{m\_pos}[0][x, y] & (0 \le (x - \text{LHoff}) < \text{LWidth}) \text{ and } (0 \le (y - \text{LVoff}) < \text{LHeight}) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

#### 5.2.4.4 Image *I* for a Layout Object

Let *spc* be the colourspace in which the *I* images are to be combined to generate a *PageImage*.

If the Layout Object box contains an image in either an image Object box or a combined image/mask Object box and also contains a Base Colour box, then let *base* be the representation in colourspace *spc* of the colour indicated by the Base Colour box; otherwise let *base* be the representation in colourspace *spc* of the colour *black*.

If the Layout Object box does not contain an image Object box or a combined image/mask Object box, then the image *I* for the layout object is defined to be an image in colourspace *spc* where every sample has the colour *base*.

If the Layout Object does contain an image Object box or a combined image/mask Object box, but with no associated codestream, then the image *I* for the layout object is also defined to be an image in colourspace *spc* where every sample has the colour *base*.

Sub-clauses 5.2.4.4.1 - 5.2.4.4.5 define *I* when a codestream is associated with a image Object box or a combined image/mask Object box in the Layout Object box.

The first stage in generating *I*, described in 5.2.4.4.1, is to decode the image object and convert the image to the *spc* colourspace.

The second stage, described in 5.2.4.4.2, is to scale the image.

The third stage, described in 5.2.4.4.3, is to clip the image from the top and from the left.

The fourth stage, described in 5.2.4.4.4, is to position the image on the page.

The fifth and final stage, described in 5.2.4.4.5, is to clip the image to the layout window.

These stages, as with those to generate the mask *M*, are described individually for clarity and an efficient JPM decoder may be able to combine one or more of these stages

##### 5.2.4.4.1 Decoding the Image

If an image Object box is used then let *i_orig* be the image obtained by decompressing the associated codestream, otherwise let *i_orig* be the image containing all but the last component of the image obtained by decompressing the associated codestream. Let *i_orig_width* and *i_orig_height* be the width and height of *i_orig*.

Let *conv* be the image obtained by converting *orig* to the colourspace *spc*.

##### 5.2.4.4.2 Scaling the Image

If an Object Scale box is contained in the image Object box or combined image/mask Object box then let *i_VRN*, *i_VRD*, *i_HRN* and *i_HRD* be the scaling factors contained in this box, otherwise let *i_VRN*, *i_VRD*, *i_HRN* and *i_HRD* all be 1. Let *i_scaled* be the result of scaling *conv* vertically by the ratio $\frac{\text{i\_VRN}}{\text{i\_VRD}}$ and horizontally by the ratio $\frac{\text{i\_HRN}}{\text{i\_HRD}}$, generating an image with dimensions:

$$\text{i\_scaled\_width} = \left\lfloor \frac{\text{i\_HRN}}{\text{i\_HRD}} \times \text{i\_orig\_width} \right\rfloor \text{ and } \text{i\_scaled\_height} = \left\lfloor \frac{\text{i\_VRN}}{\text{i\_VRD}} \times \text{i\_orig\_height} \right\rfloor . \quad (9)$$

NOTE - An image may be scaled up using bilinear interpolation or, if it's a palette image, using nearest neighbour

interpolation. An image may be scaled down using averaging followed by subsampling, or if it's a palette image, using subsampling. If the image is compressed using JPEG 2000, an implementation may also choose to use the progressive-by-resolution decompression feature to obtain a power-of-two reduced resolution image for scaling.

#### 5.2.4.4.3 Clipping the Image

Let *i_OVoff* and *i_OHoff* be the vertical and horizontal offsets in the image Object box or combined image/mask Object box.

Let *i_clipped* be an image with bit-depth *g*, width *i_width = max(0, i_scaled_width - i_OHoff)* and *i_height = max(0, i_scaled_height - i_OVoff)*. *i_clipped* is defined as follows:

$$i\_clipped[0][x,y] = i\_scaled[0][x + i\_OHoff, y + i\_OVoff] \tag{10}$$

#### 5.2.4.4.4 Positioning the Image

Let *page_width* and *page_height* be the width of the page containing the layout object, as signalled in the Page Header box. Let *LVoff* and *LHoff* be the layout vertical and horizontal offsets in the Layout Object Header box.

Let *i_pos* be an image with a width of *page_width* and height of *page_height*, defined as follows:

$$i\_pos[0][x, y] = \begin{cases} i\_clipped[0][x - LHoff, y - LVoff] & (0 \le (x - LHoff) < i\_width) \text{ AND } (0 \le (y - LVoff) < i\_height) \\ base[c] & \text{otherwise} \end{cases} \tag{11}$$

#### 5.2.4.4.5 Window Clipping the Image

Let *I* be an image with a width of *page_width* and height of *page height*. Let *LWidth* and *LHeight* be the layout object width and layout object height as specified in the Layout Object Header box. *I* is defined as follows:

$$I[c][x, y] = \begin{cases} i\_pos[c][x, y] & (0 \le (x - LHoff) < LWidth) \text{ and } (0 \le (y - LVoff) < LHeight) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

### 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 JPM file with a document thumbnail is created by adding a JP2 Header box for the thumbnail image immediately after the File Type box. The codestream for the thumbnail image must be contained in the first Contiguous Codestream box following the JP2 Header box.

NOTE - A JP2-compatible document thumbnail can be used in the construction of a JPM file that also conforms to the JP2 file format defined in ITU-T Rec T.800 | ISO/IEC 15444-1. A JP2 compatible document thumbnail must use JPEG 2000 as the coder and 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\040" to the compatibility string. A JP2 reader shall then look for a JP2 Header box and jump over any following boxes until it finds a Contiguous Codestream box. It would then access 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.

A page thumbnail is simply the layout object in a page with layout object identifier 0. A page thumbnail layout object must be 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 page thumbnail. Page thumbnails are optional.

NOTE - 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 allow the display of a more useful initial image onto a new page than simply the page's background colour while waiting for additional content to arrive.

### 5.2.6 Contiguous and Fragmented Codestreams

The codestream data for the objects of a page may either be a single contiguous codestream, or be composed of one or more codestream fragments. A codestream consisting of fragments is accessed via a Fragment Table box. A Fragment List box lists the locations in the file, or external pointers to another file, for the codestream fragments. Each codestream fragment is contained within a Media Data box. If the codestream is not fragmented then the object shall contain the location of a Contiguous Codestream box containing the codestream data. Full details of these boxes may be found in Annex B.5.

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 URI 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 be decoded in this way.

Fragment table entries always point to codestreams. All header information required to decode the codestream, such as width and height, compression type or bit depth is stored in fields in the Object box.

### 5.2.7 Shared Data

A file can contain multiple boxes or fragments containing identical data. File size can be reduced by sharing these boxes, including by reference a box or fragment that occurs in one part of a JPM file in a different part of the same file or in a different JPM file. The two mechanisms for sharing boxes in a JPM file are Shared-Data Reference box and Cross-Reference box. Wherever a Shared-Data Reference or a Cross-Reference box occurs in the file, the data it references is treated as if it exists at the point in the file where the reference is located. The reference can be by means of either an identifier, in the case of a Shared-Data Reference box (Annex B.1.8), or a data reference including an offset and length, in the case of a Cross-Reference box (Annex B.6.4).

A Shared-Data Reference box in a file uses an identifier that is defined in the same file; therefore this mechanism shall only be used to share data within a file. The Cross-Reference box uses a data reference and therefore may be used to share data across files. In the case of a reference to an offset from the beginning of the file and length, the offset applies before any substitution implied by the reference occurs. A shared data substitution would not be understood by a JP2-only reader, so that a JP2-conforming JPM file must include directly any box whose presence in the file is necessary for JP2 conformance.

### 5.2.8 Metadata

This International Standard allows metadata to be attached to any structural element of a JPM file, including the file or document itself, page collections, pages, layout objects and objects. This is indicated in the box

definitions in Annex B by the inclusion of a “metadata” box in the data field of the container box. Whenever there is a reference to a metadata in a box definition, it is understood that the box may contain an XML box or a UUID box with metadata. These “metadata” boxes are defined in Annex C.

## 5.3 JPM Use Scenarios (Informative)

### 5.3.1 Self-Contained Files and Files with External References

Clause 5.2 outlines the 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 available and local bandwidths are high.

The multiple layout object model is a method of representing and constructing document images, which allows access paradigms based on the multiple objects in addition to providing compression advantages due to applying the optimal compression method to each object. 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, whose contents are distributed across multiple files, some of which may be located remotely on networks such as the Internet. This sub-clause examines some of the file organisations enabled by these features.

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 URI pointing to an Internet file.

### 5.3.2 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.

### 5.3.3 Page Collection Example

The main page collection of a very large multi-volume set of books such as an encyclopedia should not 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 main page collection might serve as what amounts to 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, and so on. Each of these page collections could have an appropriate label applied by means of Label boxes in their Page Collection boxes.

A search of metadata for a term occurring on a few of the pages returns a new “search results” JPM file containing a single main “search results” page collection pointing to a handful of pages. The first entry in this page collection box would point directly to the first search result page. This would actually point to one of the pages in the encyclopedia over on the server 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 client software would go to the PPCLoc box found in that page's Page box. This entry 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 first find the current page’s entry in that page collection by means of the PIx portion of the PPCLoc box. The PIx + 1 entry in that page collection would be the next page. By that means, the previous and next pages can be found.

### 5.3.4 Page Collections in a Client / Server Context

The JPM file format places Page boxes and Page Collection boxes at the file level (not within other superboxes) to assist with incremental browsing of pages. New pages or page collections drawn over from a server may simply be appended to the end of the local (incomplete) copy of the file.

In a client/server environment such as the web, it would be common for a user of a client application to browse only certain pages of a multi-page JPM file on a server. An initial link could point the client application to a specific page in the server’s JPM file. The client application would create a new local JPM file and could choose to continuously update it as portions of the server file are retrieved. This would ensure that the browser cache always contains a valid JPM file when browsing ceases.

This local file would have a main page collection. The main page collection would initially contain just one page. As the user begins fetching additional pages, these would be appended to the local file and the main page collection’s page table updated to point to each new local copy of a server page. At the end of browsing, a stand-alone file having no clear connection to the server file could exist on the client.

A better way of performing this browsing step would be to copy the server’s main page collection to the client prior to downloading any pages. This page collection would be fixed up to point to the server copies of the pages by creating a data reference to the server URL. Then, as the first page is brought down to the client, the main page collection is updated to point to the local page, while continuing to point to the server for all other pages.

### 5.3.5 Example of a Client/Server Interaction

In a client/server system, a user might want to browse a multi-page JPM file on a server, and to begin on a page in the middle of that remote file. Perhaps a search engine found a hit on the JPM file metadata associated with that page. The URL returned by the search engine would point to the JPM file and could contain a construction such as http://webimaging.org/test.jpm?page=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, byte serving 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 substantial 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.

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 located 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, Shared Data Reference boxes permit 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 directly to the first byte of the codestream fragment, beyond the beginning of any containing Media Data box or other box.

### 5.3.6 Example of Scanner Capture

In addition to supporting a variety of streaming and remote browsing behaviors, 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.

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.

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.

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 colour or greyscale image suitable for use by a subsequent decomposition system. If the scanning system also is capable of producing a bitonal (depth 1 bit) image natively, it may wish to include both this image and the full colour 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 colour 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

# Annex A

(normative)

# Compound Image File Structure

This Annex describes the structure and organization of a JPM file. A JPM file uses the file format architecture specified in ITU-T Rec T.800 | ISO/IEC 15444-1. Therefore, a JPM file is a contiguous sequence of boxes. This Annex defines a box and describes the notation used for the box definitions in this International Standard. This Annex also lists the boxes that are used in a JPM file.

## A.1 File Identification

The brand shall be 'jpm\040' for files that are completely defined by this Part of this International Standard; brand is defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I. JPM files are files that conform to this Part of this 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 ITU-T Rec T.800 | ISO/IEC 15444-1. 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 Rec T.800 | ISO/IEC 15444-1 Annex I.4).

This International Standard defines boxes mandatory to 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 data needed to composite the page or pages in the file. A JPM file may also reference images and data in external files. References to the content of an external file are by means of the Data Reference box (Annex B.1.5) or Cross Reference boxes (Annex B.6.4).

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.

.

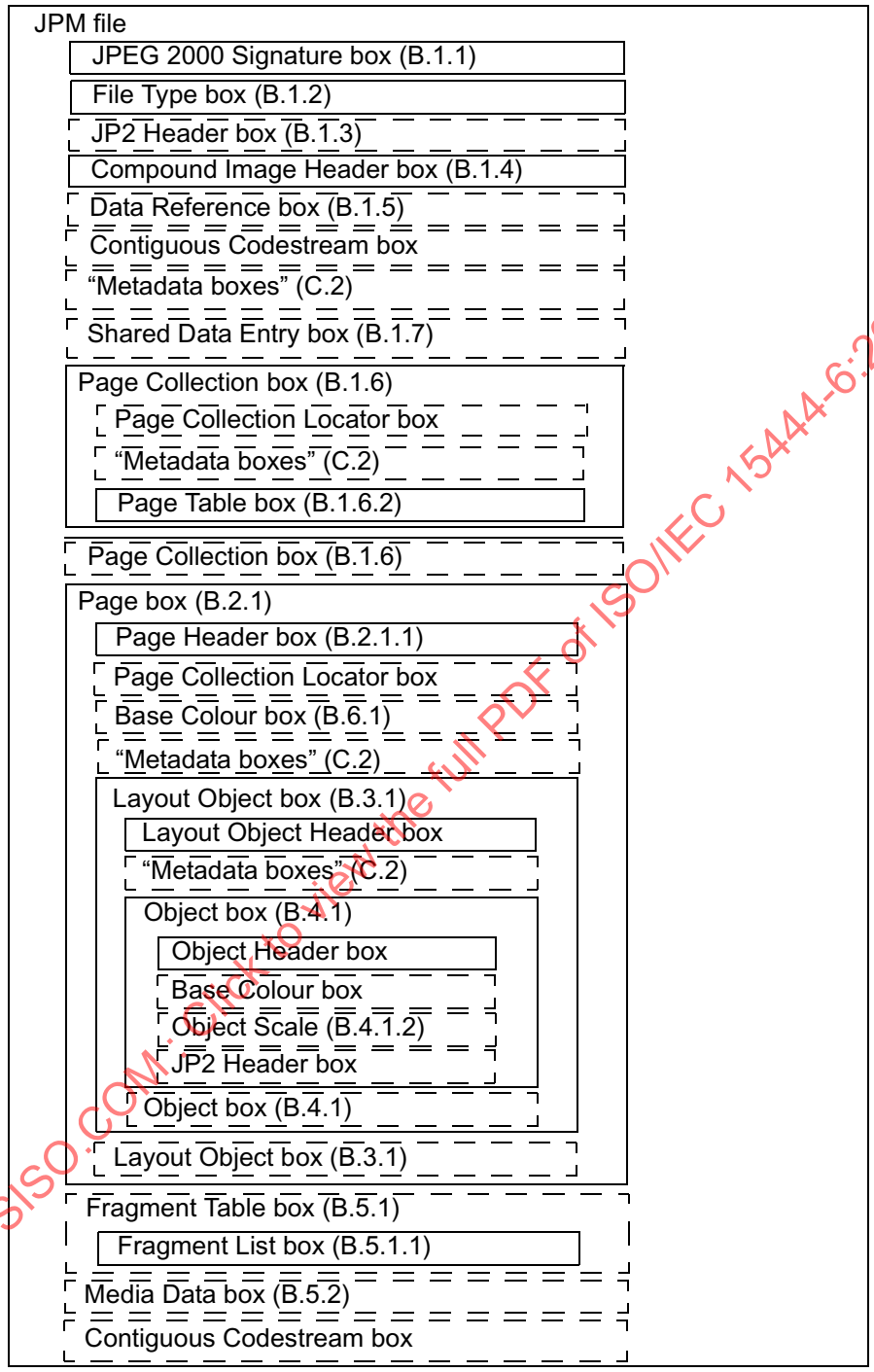




**Figure A-1 Conceptual structure of a JPM file**

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.

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 and a Compound Image Header box. 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.

## A.3 Box Definition

Physically, each element in a JPM file is encapsulated within a binary structure called a box. That binary structure is defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.4.

Annex B defines the boxes used in this Part of this International Standard. Each box is described using the notation defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.3.6.

The following additional information is provided 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. Container specifies the box that contains the box being defined. 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.

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

— Some boxes have values assigned to groups of bits within a field. In some cases there are bits, denoted by “x”, that are not assigned a value for any field within a parameter. The fields shall contain a value of zero for all such bits. The decoder shall ignore these bits.

## A.4 Boxes Used in a Compound Image File

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

**Table A-1 Boxes defined or referenced within this International Standard**

| Box name | Type | Superbox | Comments (Informative) |
|---|---|---|---|
| Base Colour | ‘bclr’ (0x6263 6C72) | Yes | This box contains all the information specifying the colour of a page or of an object for which no image data is available. |
| Base Colour Value | ‘bcvl’ (0x6263 766C) | No | This box specifies the component values for the colour of a page or of an object for which no image data is available. |
| Colour Specification | ‘colr’ (0x636F 6C72) | No | This box specifies the colourspace of an image. |
| Compound Image Header | ‘mhdr’ (0x6D68 6472) | No | This box contains general information about the compound image file, such as profile and version. |
| Contiguous Code-stream | ‘jp2c’ (0x6A70 3263). | No | This box contains a valid and complete JPEG 2000 codestream. |
| Cross-Reference | ‘cref’ (0x6372 6566) | 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. |
| Data Reference | ‘dtbl’ (0x6474 626C) | No | This box contains a set of pointers to other files or data streams not contained within the JPM file itself. |
| File Type | ‘ftyp’ (0x6674 7970) | No | This box specifies file type, version and compatibility information, including specifying if this file is a conforming JPM file. |
| Fragment List | ‘flst’ (0x666C 7374) | No | This box specifies a list of fragments that make up one particular codestream within the file. |
| Fragment Table | ‘ftbl’ (0x6674 626C) | Yes | This box describes how a codestream has been split up or fragmented and then stored within the file. |
| Free | ‘free’ (0x6672 6565) | No | This box contains data that is no longer used and may be overwritten when the file is updated. |
| Image Header | ‘ihdr’ (0x6968 6472) | No | This box specifies the size of the image and other related fields. |
| JP2 Header | ‘jp2h’ (0x6A70 3268) | Yes | This box contains a series of boxes that contain header-type information about a file containing a single image. |
| JPEG 2000 Signature | ‘jP\040\040’ (0x6A50 2020) | No | This box uniquely defines the file as being part of the JPEG 2000 family of files. |
| Label | ‘lbl\040’ (0x6C62 6C20) | No | This box specifies a textual label for a Page box or a Page Collection box. |
| Layout Object | ‘lobj’ (0x6C6F 626A) | Yes | This box contains the information and image data needed to composite a mask-image object pair. |

**Table A-1 Boxes defined or referenced within this International Standard** (continued)

| Box name | Type | Superbox | Comments (Informative) |
|---|---|---|---|
| Layout Object Header | 'lhdr' (0x6C68 6472) | No | This box describes the properties of the layout object and assigns each a unique identifier within the page. |
| Media Data | 'mdat' (0x6D64 6174). | No | This box contains generic media data, which is referenced through the Fragment List box. |
| Object | 'objc' (0x6F62 6A63) | Yes | This box contains all the image data and information for one object in a layout object. |
| Object Header | 'ohdr' (0x6F68 6472) | No | 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 | This box describes the scaling for an object before applying it to the page. |
| Page | 'page' (0x7061 6765) | Yes | This box contains all the information needed to image a page including references to code-stream data. |
| Page Collection | 'pcol' (0x70636F6C) | Yes | This box groups together the locations of a set of pages so that they are treated as related and associated with each other. |
| Page Header | 'phdr' (0x7068 6472) | No | This box describes the properties of a page and gives the number of layout objects in the page. |
| Page Table | 'pagt' (0x7061 6774) | No | This box gives the locations of the pages in a page collection and enables random access of pages in a file. |
| Primary Page Collection Locator | 'ppcl' (0x7070 636C) | No | This box gives the location of the primary page collection for the page collection. |
| Shared Data Entry | 'sdat' (0x7364 6174) | Yes | 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 | This box can be used to insert a box in the file by reference to a previous occurrence of the box in the same file. |
| UUID | 'uuid' (0x7575 6964) | No | This box contains vendor specific information. |
| UUID Info | 'uinf' (0x7569 6E66) | Yes | This box contains additional information associated with a UUID. |
| XML | 'xml\040' (0x786D 6C20) | No | This box contains vendor specific information in XML format. |

# Annex B

(normative)

# Box Definitions

This Annex defines the boxes that are used in a conforming JPM file and that 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' (0x6A502020)
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: First box in file

The format and structure of the JPEG 2000 Signature box is defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.1.

### B.1.2 File Type box

Box type: 'ftyp' (0x66747970)
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: Immediately after the JPEG 2000 Signature box

The format and structure of the File Type box is defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.1.

The BR field in the File Type box shall be 'jpm\040' for files that are completely defined by this International Standard. In addition, a file that conforms to this International Standard shall have at least one $CL^i$ field in the File Type box, and shall contain the value 'jpm\040' in one of the $CL^i$ fields in the File Type box.

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

Box type: ‘jp2h’ (0x6A703268)
Container: File
Mandatory: No
Quantity: At most one
Location: Anywhere after the File Type box

The format and structure of the JP2 Header box is defined in Annex B.6.2.

The first file level JP2 Header box 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 thumbnail codestream. If the JP2 Header box and the Contiguous Codestream box are JP2 compatible, then the JPM file is JP2 compatible and ‘jp2\040’ occurs in the compatibility list in the File Type box. See 5.2.5 for more details.

### B.1.4 Compound Image Header box

Box type: ‘mhdr’ (0x6D686472)
Container: File
Mandatory: Yes
Quantity: Exactly one
Location: Anywhere after the File Type box

The Compound Image Header box contains general information about the compound image. It is recommended that this box be placed immediately after the File Type box and any file level JP2 Header box.

The type of a Compound Image Header box shall be ‘mhdr’ (0x6D686472). This box contains the following fields.

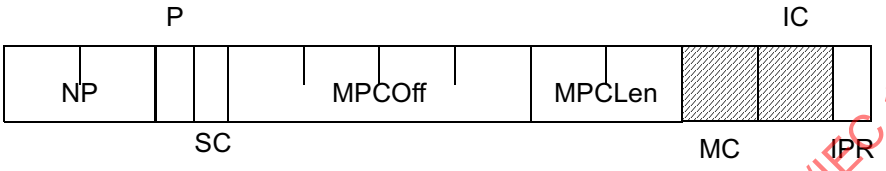


**Figure B-1 Organization of the contents of a Compound Image Header box**

**NP:** Number of pages. This parameter specifies the number of Page boxes in the compound image file and is stored as a 4-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-1. See Annex D for full details of the profiles.

**Table B-1 Legal P values**

| Value | Meaning |
|---|---|
| 0 | No profile specified |
| 1 | Web profile |
| other values | Reserved for ISO use. |

**SC:** Self-contained: This value is stored as a 1-byte unsigned integer. If it has value 1, then the file is self contained and has no references to external data; otherwise, it has value 0.

**MPCOff:** Main Page Collection Offset. This field specifies the offset from the start of the file to the first byte of the main Page Collection box. This field is encoded as an 8-byte unsigned big endian integer.

**MPCLen:** Main Page Collection Length. This field specifies the length of the main Page Collection box. This field is encoded as a 4-byte unsigned big endian integer.

**MC:** Mask coders; the coder or coders that may be 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-2.

**Table B-2 Mask Coders**

| Values (bits)<br>MSB LSB | Mask Coder Flags |
|---|---|
| xxxx xxx0<br>xxxx xxx1 | One dimensional ITU-T Rec. T.4 (MH) coding is not used<br>One dimensional ITU-T Rec. T.4 (MH) coding may be used |

**Table B-2 Mask Coders** (continued)

| Values (bits)<br>MSB LSB | Mask Coder Flags |
|---|---|
| xxxx xx0x<br>xxxx xx1x | Two dimensional ITU-T Rec. T.4 (MR) coding is not used<br>Two dimensional ITU-T Rec. T.4 (MR) coding may be used |
| xxxx x0xx<br>xxxx x1xx | ITU-T Rec. T.6 (MMR) coding is not used<br>ITU-T Rec. T.6 (MMR) coding may be used |
| xxxx 0xxx<br>xxxx 1xxx | ITU-T Rec. T.82 (JBIG) coding is not used<br>ITU-T Rec. T.82 (JBIG) coding may be used |
| xxx0 xxxx<br>xxx1 xxxx | ITU-T Rec. T.800 (JPEG 2000) coding is not used<br>ITU-T Rec. T.800 (JPEG 2000) coding may be used |
| xx0x xxxx<br>xx1x xxxx | ITU-T Rec. T.88 (JBIG2) coding applying ITU-T Rec. T.89 is not used<br>ITU-T Rec. T.88 (JBIG2) coding applying ITU-T Rec. T.89 may be used |
| x0xx xxxx<br>x1xx xxxx | Reserved for ISO use<br>Reserved for ISO use |
| 0xxx xxxx<br>1xxx xxxx | Last byte specifying the coders which may be used<br>Extend with another byte specifying coders which may be used |

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.

**IC:** Image coders; the coder or coders that may be 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-3.

**Table B-3 Image Coders**

| Values (bits)<br>MSB LSB | Image Coder Flags |
|---|---|
| xxxx xxx0<br>xxxx xxx1 | ITU-T Rec. T.81 (JPEG) coding is not used<br>ITU-T Rec. T.81 (JPEG) coding may be used |
| xxxx xx0x<br>xxxx xx1x | ITU-T Rec. T.82 (JBIG) coding applying ITU-T Rec. T.43 is not used<br>ITU-T Rec. T.82 (JBIG) coding applying ITU-T Rec. T.43 may be used |
| xxxx x0xx<br>xxxx x1xx | ITU-T Rec. T.45 (Run-Length Colour Encoding) coding is not used<br>ITU-T Rec. T.45 (Run-Length Colour Encoding) coding may be used |
| xxxx 0xxx<br>xxxx 1xxx | ITU-T Rec. T.87 (JPEG-LS) coding is not used<br>ITU-T Rec. T.87 (JPEG-LS) coding may be used |
| xxx0 xxxx<br>xxx1 xxxx | ITU-T Rec. T.800 (JPEG 2000) coding is not used<br>ITU-T Rec. T.800 (JPEG 2000) coding may be used |
| xx0x xxxx<br>xx1x xxxx | Reserved for ISO use<br>Reserved for ISO use |
| x0xx xxxx<br>x1xx xxxx | Reserved for ISO use<br>Reserved for ISO use |
| 0xxx xxxx<br>1xxx xxxx | Last byte specifying the coders which may be used<br>Extend with another byte specifying coders which may be used |

Bit 7, the extend bit, would be set when adding another byte to accommodate future support for 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 C.1. Other values are reserved for ISO use.

**Table B-4 Format of the contents of the Compound Image Header box**

| Field name | Size (bits) | Value |
|---|---|---|
| NP | 32 | 0 — ($2^{32}$-1) |
| P | 8 | see Table B-1 |
| SC | 8 | 0 or 1 |
| MPCOff | 64 | 12 — ($2^{64}$-1) |
| MPCLen | 32 | 0 — ($2^{32}$-1) |
| MC | variable | see Table B-2 |
| IC | variable | see Table B-3 |
| IPR | 8 | 0 — 1 |

### B.1.5 Data Reference box

Box type: ‘dtbl’ (0x6474626C)
Container: File
Mandatory: No
Quantity: At most one
Location: Anywhere after the Compound Image Header box

The Data Reference box contains an array of URLs that are referenced by this 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.

— In a Page Table box, a data reference and offset refer to an external Page box or Page Collection box.

— In an Object Header box, a data reference and offset refer to an external Fragment Table box.

— In a Fragment List box within a Fragment Table box, a data reference and offset refer to an external codestream fragment.

— In a Fragment List box within a Cross-Reference box, a data reference and offset refer to an external shared header or metadata fragment.

The type of a Data Reference box shall be ‘dtbl’ (0x6474626C). This box contains the following fields:

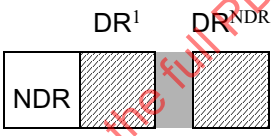


**Figure B-2 Organization of the contents of a Data Reference box**

**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 specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.7.3.2. However, in this context, the Location field in the box is not specific to UUID Info boxes. The meaning of the URL is specified in the context of the box that refers to the particular entry in the Data Reference box.
The indices of the elements in the array of DR$^i$ fields are 1 based; a data reference of 1 in a DR$^i$ 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-5 Format of the contents of the Data Reference box**

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

### B.1.6 Page Collection box (superbox)

Box type: ‘pcol’ (0x70636F6C)
Container: File
Mandatory: No
Quantity: At least one
Location: Anywhere at the file level after the File Type box

NOTE - For efficient access, an optimized file would have the first Page Collection box 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 a Primary Page collection box that is required except when the page collection is the main page collection without a primary page collection as described in 5.2.2.2, 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 shall have at least one page collection. Every page in a JPM file belongs to what is called its primary page collection. The role of the primary page collection is discussed in 5.2.1.

The type of a Page Collection box shall be ‘pcol’ (0x70636F6C). This box contains the following fields:

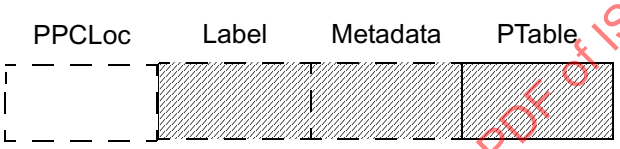


**Figure B-3 Organization of the contents of a Page Collection box**

**PPCLoc:** Primary Page Collection Locator box. This box contains a Page Collection Locator box as specified in Annex B.1.6.1.

**Label:** Label box. This optional field may contain a Label box as specified in Annex B.6.3. This box contains a label associated with the Page Collection; this box is optional.

**Metadata:** Optional metadata boxes

**PTable:** Page Table box. This field contains a Page Table box as specified in Annex B.1.6.2.

### B.1.6.1 Primary Page Collection Locator box

Box type: ‘ppcl’ (0x7070636C)
Container: Page Collection box or Page box
Mandatory: Yes, except in the case of a single page, self-contained JPM file, where it is optional in the Page box.
Quantity: At most one
Location: Anywhere

Each Page Collection box and Page box shall contain a Primary Page Collection Locator box. A Primary Page Collection Locator box contains a set of fields used to locate the “Primary Page Collection” of the page or page collection. The role of the primary page collection is discussed in 5.2.1.

The type of a Primary Page Collection Locator box shall be ‘ppcl’ (0x7070636C). This box contains the following fields:

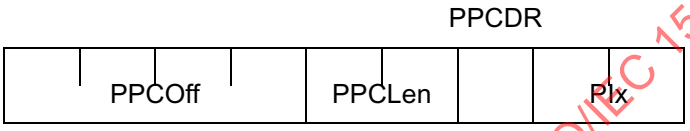


**Figure B-4 Organization of the contents of a Primary Page Collection Locator box**

**PPCOff:** Primary Page Collection Offset. This field specifies the offset from the start of the file to the first byte of the primary Page Collection box to which the page or page collection container belongs. This field is encoded as an 8-byte big endian unsigned integer.

**PPCLen:** Primary Page Collection Length. This field specifies the length of the primary Page Collection box for this page. This field is encoded as a 4-byte big endian unsigned integer.

**PPCDR:** Primary Page Collection Data Reference. This field specifies the data file or resource that contains the primary page collection to which the page or page collection container belongs. 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 2-byte big endian unsigned integer.

**PIx:** Primary Page Collection Page Table Index. This field specifies an index in the Page Table of the primary page collection to which the page or page collection container belongs. The entry at the specified index in the Page Table of the primary page collection shall reference the page or page collection contained in the primary page collection. This field is encoded as a 4-byte big endian unsigned integer.

**Table B-6 Format of the contents of the Primary Page Collection Locator box**

| Field name | Size (bits) | Value |
|---|---|---|
| PPCOff | 64 | 0 — ($2^{64}$–1) |
| PPCLen | 32 | 0 — ($2^{32}$–1) |
| PPCDR | 16 | 0 — 65 535 |
| PIx | 32 | 0 — ($2^{32}$–1) |

### B.1.6.2 Page Table box

Box type: ‘pagt’ (0x70616774)
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 box data contains 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 type of a Page Table box shall be ‘pagt’ (0x70616774). This box contains the following fields:

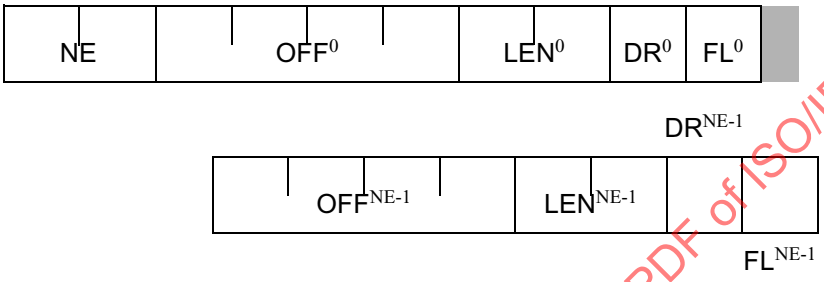


**Figure B-5 Organization of the contents of a Page Table box**

**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. This field is encoded as a 4-byte big endian unsigned integer.

**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 an 8-byte 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 4-byte 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 2-byte big endian unsigned integer.

**FL$^i$:** Flag. This field indicates the type of entry. This field is encoded as a 1-byte unsigned integer. Legal values of this field are as follows:

**Table B-7 Legal FL values**

| Value<br>MSB LSB | Meaning |
|---|---|
| xxxx x001 | Offset to Page box |
| xxxx x011 | Offset to Page box containing thumbnail. |
| xxxx x000 | Offset to non-auxiliary Page Collection box. |
| xxxx x100 | Offset to auxiliary Page Collection box. |
| xxxx 0xxx<br>xxxx 1xxx | Offset to Page or Page Collection box containing no metadata<br>Offset to Page or Page Collection box containing metadata |
| other values | Reserved for ISO use. |

**Table B-8 Format of the contents of the Page Table box**

| Parameter | Size (bits) | Value |
|---|---|---|
| NE | 32 | 1 — ($2^{32}$–1) |
| $OFF^i$ | 64 | 12 — ($2^{64}$–1) |
| $LEN^i$ | 32 | 0 — ($2^{32}$–1) |
| $DR^i$ | 16 | 0 — 65 535 |
| $FL^i$ | 8 | see Table B-7 |

### B.1.7 Shared Data Entry box

Box type: ‘sdat’ (0x73646174)
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere after the Compound Image Header box

The Shared Data Entry box contains an Identifier 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 Reference box shall be ‘sdat’ (0x73646174). This box contains the following fields:

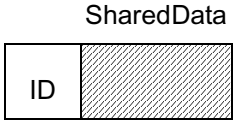


**Figure B-6 Organization of the contents of a Shared Data Entry box**

**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.

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

**Table B-9 Format of the contents of a Shared Data Entry box**

| Field name | Size (bits) | Value |
|---|---|---|
| ID | 16 | 0 — 65 535 |
| Shared Data | Variable | Variable |

### B.1.8 Shared Data Reference box

Box type: ‘sref’ (0x73726566)
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.

Any offsets in the file are understood to apply before any substitution implied by the Shared Data Reference.

The type of a Shared Data Reference box shall be ‘sref’ (0x73726566). This box contains the following field:


**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.

**Table B-10 Format of the contents of the Shared Data Reference box**

| Field name | Size (bits) | Value |
|---|---|---|
| ID | 16 | 0 — 65 535 |

## B.2 Page Level Boxes

### B.2.1 Page box (superbox)

Box type: ‘page’ (0x70616765)
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 a Page box shall be ‘page’ (0x70616765). This box contains the following fields:

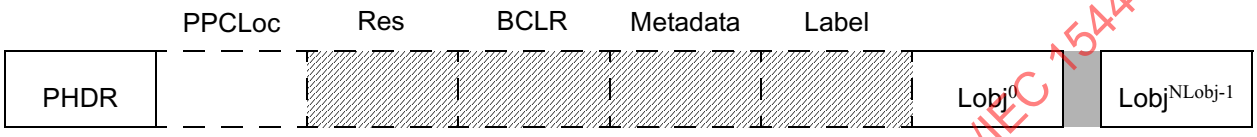


**Figure B-8 Organization of the contents of a Page box**

**PHDR:** Page Header box. This field contains a Layout Object Header box as specified in Annex B.2.1.1.

**PPCLoc:** Primary Page Collection Locator box. This box contains a Page Collection Locator box as specified in Annex B.1.6.1. This is mandatory except in the case of a single page, self-contained JPM file, where it is optional.

**Res:** Resolution box. This optional field may contain a Resolution box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.7. This box gives the resolution of a page grid unit. Together with the page height and width in the Page Header box, it gives the physical size of the page.

**BCLR:** Base Colour box. This optional field may contain a Base Colour box as specified in Annex B.6.1. The Base Colour box occurs in a Page box when the value of the PColour field in the Page Header box is 255. Within a Page Box it specifies the colour that is applied prior to rendering any objects onto the page.

**Metadata:** Optional metadata boxes.

**Label:** Label box. This field may contain a Label box as specified in Annex B.6.3. This box contains a label associated with the page; this box is optional.

**Lobj$^{i}$:** Layout Object box i, where i = 0, NLobj-1; the Layout Object box is specified in Annex B.3.1. NLobj is the number of layout objects on this page and is specified in the Page Header box contained within this box.

### B.2.1.1 Page Header box

Box type: ‘phdr’(0x70686472)
Container: Page box
Mandatory: Yes
Quantity: Exactly one
Location: First box in a Page box

A Page Header box specifies 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.

The type of a Page Header box shall be ‘phdr’ (0x70686472). This box contains the following fields:

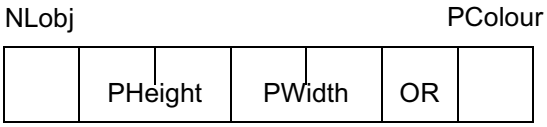


**Figure B-9 Organization of the contents of a Page Header box**

**NLobj:** Number of layout objects on this page. This value is stored as a 2-byte big endian unsigned integer.

**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 to be 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 to be 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-11 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 |
| | All other values reserved |

**PColour:** Page Colour. This field indicates whether the page is transparent, in which case the layout objects are imaged onto whatever exists within the boundaries of the page, or whether the page has a colour, in which case, the page colour 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-12 Legal Pcolour values**

| Value | Meaning |
|---|---|
| 0 | Page is transparent |
| 1 | Page is white |
| 2 | Page is black |
| 255 | Page colour is specified in Base Colour box (Annex B.6.1) |
| | All other values reserved |

**Table B-13 Format of the contents of a Page Header box**

| Field name | Size (bits) | Value |
|---|---|---|
| NLobj | 16 | 0 — 65 535 |
| PHeight | 32 | 1 — $(2^{32}-1)$ |
| PWidth | 32 | 1 — $(2^{32}-1)$ |
| OR | 16 | see Table B-11 |
| PColour | 16 | see Table B-12 |

## B.3 Layout Object Level Boxes

### B.3.1 Layout Object box (superbox)

Box type: 'lobj' (0x6C6F626A)
Container: Page box
Mandatory: Yes
Quantity: Any number
Location: Anywhere after Page Header box

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, and an Object Header box (for image data) and an optional Object Header box (for mask data).

The type of a Layout Object box shall be 'lobj' (0x6C6F626A). This box contains the following fields:

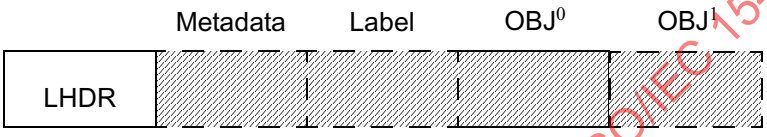


**Figure B-10 Organization of the contents of a Layout Object box**

**LHDR:** Layout Object Header box. This field contains a Layout Object Header box as specified in Annex B.3.1.1.

**Metadata:** Optional metadata boxes.

**Label:** Label box. This field may contain a Label box as specified in Annex B.6.3. This box contains a label associated with the page; this box is optional.

**OBJ$^0$:** Object box; the first Object box. This field contains an Object box as specified in Annex B.4.1.

**OBJ$^1$:** Object box; the second Object box. This field contains an Object box as specified in Annex B.4.1; this box is optional.

### B.3.1.1 Layout Object Header box

Box type: 'lhdr' (0x6C686472)
Container: Layout Object box
Mandatory: Yes
Quantity: Exactly one
Location: First box in Layout Object box

This box gives the layout object ID (unique within a page), height (in page grid units), width (in page grid units), offset with respect to the page and style of the layout object.

The type of a Layout Object Header box shall be 'lhdr' (0x6C686472). This box contains the following fields:

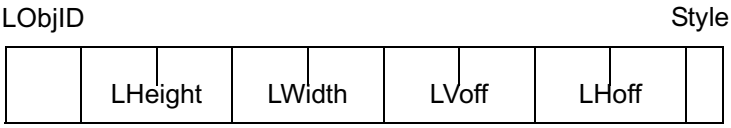


**Figure B-11 Organization of the contents of a Layout Object Header box**

**LObjID:** Layout Object Identifier. Each layout object in a given Page box has a unique LObjID value and Layout Object boxes within a Page box must occur in order of increasing LObjID value. If the page does not contain a page thumbnail then the LObjID values range from 1 to NLobj, with NLobj defined in the Page Header box within the Page box. If a page does contain a page thumbnail then the LObjID values for the contained Layout Object boxes range from 0 to NLobj - 1, with a LObjID value of 0 identifying the page thumbnail object (see Clause 5.2.5). This value is stored as a 2-byte big endian unsigned integer.

**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 consist 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 1-byte unsigned integer. Values defined in this specification are:

**Table B-14 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 | User specified layout object. |
| | All other values reserved |

**Table B-15 Format of the contents of the Layout Object Header box**

| Field name | Size (bits) | Value |
|---|---|---|
| LObjID | 16 | 0 — 65 535 |
| 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 | 8 | see Table B-14 |

## B.4 Object Level Boxes

### B.4.1 Object box (superbox)

Box type: ‘objc’ (0x6F626A63)
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 objects.

The type of an Object box shall be ‘objc’ (0x6F626A63). This box contains the following fields:

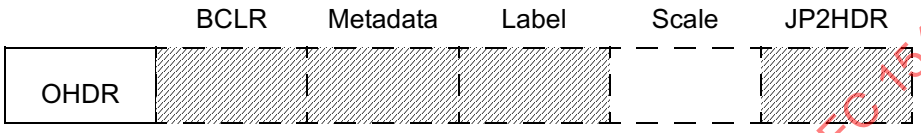


**Figure B-12 Organization of the contents of an Object box**

**OHDR:** Object Header box. This field contains an Object Header box as specified in Annex B.4.1.1.

**BCLR:** Optional Base Colour box. This optional field may contain a Base Colour box as specified in Annex B.6.1. The Base Colour 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.

**Metadata:** Optional metadata box.

**Label:** Optional Label box. This optional field may contain a Label box as specified in Annex B.6.3. This box contains a label associated with the page.

**Scale:** Optional Object Scale box. This optional field may contain an Object Scale box as specified in Annex B.4.1.2. It 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. This optional field may contain a JP2 Header box as specified in Annex B.4.1.3.

### B.4.1.1 Object Header box

Box type: ‘ohdr’ (0x6F686472)
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 coloured by a unique colour specified by the Base Colour 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 type of a Object Header box shall be ‘ohdr’ (0x6F686472). This box contains the following fields:

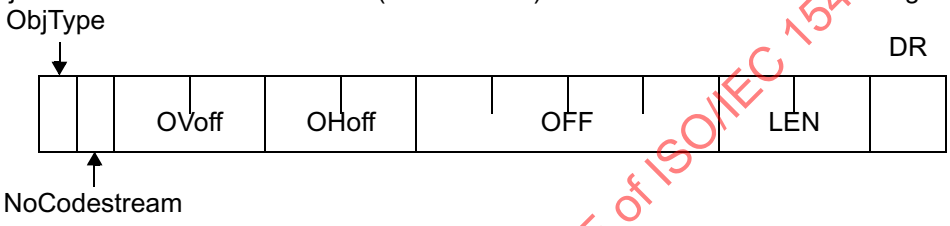


**Figure B-13 Organization of the contents of an Object Header box**

**ObjType:** Object Type; Identifies whether the object is a mask object or an image object in the Layout Object. The value is a 1-byte big endian unsigned integer. Values defined in this specification are:

**Table B-16 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 |
| | All other values reserved |

**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 colour value as specified in the Base Colour box is used as the object. The value is a 1-byte big endian unsigned integer. Values defined in this specification are:

**Table B-17 Legal NoCodestream values**

| Value | Meaning |
|---|---|
| 0 | The object contains a codestream |
| 1 | The object does not contain a codestream |
| | All other values reserved |

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

**OHoff:** Horizontal offset in the scaled object; the horizontal offset into the scaled object in page grid units. This field is a 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 an 8-byte big endian unsigned integer. If the NoCodestream field has a value of 1, then this field is ignored.

**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-18 Format of the contents of the Object Header box**

| Field name | Size (bits) | Value |
|---|---|---|
| ObjType | 8 | see Table B-16 |
| NoCodestream | 8 | see Table B-17 |
| 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 — 65 535 |

### B.4.1.2 Object Scale box

Box type: ‘scal’ (0x7363616C)
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 vertical and horizontal scaling ratios are assumed to be 1, i.e. no scaling. This box describes the scaling required to transform from object units to 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 $\frac{HRN}{HRD}$. See 5.2.4 for a full description of the scaling operation.

The type of a Object Scale box shall be ‘scal’ (0x7363616C). This box contains the following fields:

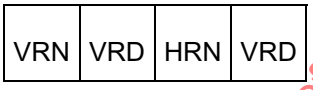


**Figure B-14 Organization of the contents of an Object Header box**

**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-19 Format of the contents of the Object Scale box**

| Field name | Size (bits) | Value |
|---|---|---|
| VRN | 16 | 1 — 65 535 |
| VRD | 16 | 1 — 65 535 |
| HRN | 16 | 1 — 65 535 |
| HRD | 16 | 1 — 65 535 |

#### B.4.1.3 JP2 Header box (superbox) in Object box

Box type: ‘jp2h’ (0x6A70 3268')
Container: Object box
Mandatory: No
Quantity: At most one; always exists if the NoCodestream field in the Object Header box is 0
Location: Immediately after the Object Scale box if present, anywhere after the Object Header box otherwise

The format and structure of the JP2 Header box is defined in Annex B.6.2.

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. Within an Object box of a JPM file, there shall be one and only one JP2 Header box.

## B.5 Codestream Element Boxes

### B.5.1 Fragment Table box (superbox)

Box type: ‘ftbl’ (0x6672626C)
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere after File Type box

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 a Fragment Table box shall be ‘ftbl’ (0x6672626C). This box contains the following field:

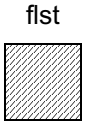


**Figure B-15 Organization of the contents of a Fragment Table box**

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

### B.5.1.1 Fragment List box

Box type: ‘flst’ (0x666C7374)
Container: Cross-Reference box and 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 contents of the referenced box; it shall not point to the header of the referenced box. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox.

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

The type of a Fragment List box shall be ‘flst’ (0x666C7374). This box contains the following fields:

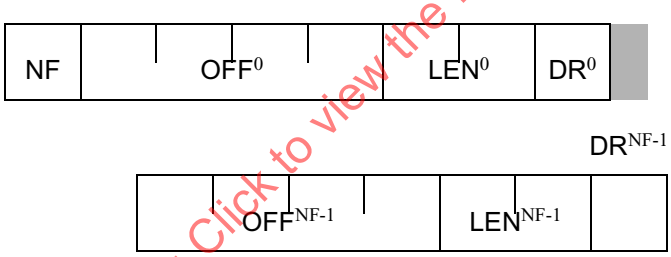


**Figure B-16 Organization of the contents of a Fragment List box**

**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 an 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-20 Format of the contents of the Fragment List box**

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

### B.5.2 Media Data box

Box type: ‘mdat’ (0x6D646174)
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere after the File Type box

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

The type of a Media Data box shall be ‘mdat’ (0x6D646174). The contents of a Media Data box in general are not defined by this International Standard.

### B.5.3 Contiguous Codestream box

Box type: ‘jp2c’ (0x6A703263)
Container: File
Mandatory: No
Quantity: Any number
Location: Anywhere after the File Type box

The format and structure of the Contiguous Codestream box is defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.4.

## B.6 General / Common Boxes

### B.6.1 Base Colour box (superbox)

Box type: ‘bclr’(0x62636C72)
Container: Page box or Object box
Mandatory: No
Quantity: At most one
Location: Anywhere in the Page box after the Page Header box or in the Object box after the Object Header box

A Base Colour box is a superbox that specifies a base colour for an image Object or a Page.

The type of a Base Colour box shall be ‘bclr’ (0x62636C72). The data field of the box is:

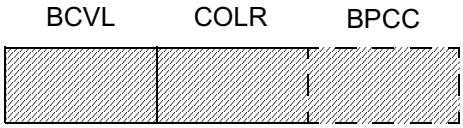


**Figure B-17 Organization of the contents of a Base Colour box**

**BCVL:** Base Colour Value box. This box specifies the individual component values for the base colour together with some additional information.

**COLR:** Colour Specification box. This box specifies the colourspace of the base colour. Its structure is specified in Annex B.6.2.2.

**BPCC:** Bits Per Component box. This box specifies the bit depth of each component in the image. This field contains a Bits Per Component box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2 of this International Standard. If the bit depth of all components in the image is the same (in both sign and precision), then this box shall not be found. Otherwise, this box specifies the bit depth of each individual component.

### B.6.1.1 Base Colour Value box

Box type: ‘bcvl’(0x6263766C)
Container: Base Colour box
Mandatory: Yes
Quantity: Exactly one
Location: Anywhere

A Base Colour Value box specifies the individual component values for the base colour together with information regarding the bit depth for the components.

The type of a Base Colour box shall be ‘bcvl’ (0x6263766C). The box contains the following fields:

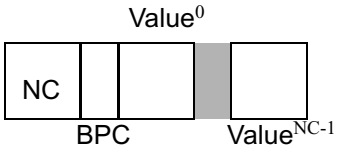


**Figure B-18 Organization of the contents of a Base Colour Value box**

**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 Base Colour box shall also contain a Bits Per Component box defining the bit depth of each component (as defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2).
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. Legal values of this field are given in Table B-22.:

**Table B-21 Legal BPC values**

| Value | Meaning |
|---|---|
| x000 0000 – x000 1111 | Component bit depth = value + 1. From 1 to 16 bits (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 |

**Value$^i$:** The value of the individual Base Colour component. Each value is represented by a 2-byte big endian signed integer.

**Table B-22 Format of the contents of the Base Colour Value box**

| Field name | Size (bits) | Value |
|---|---|---|
| NC | 16 | 1 — 16 384 |
| BPC | 8 | see Table B-21 |
| Value[i] | 16 | 0 — 65 535 |

### B.6.2 JP2 Header box (superbox)

Box type: ‘jp2h’ (0x6A703268)
Location: File or Object box

The format and structure of the JP2 Header box is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.

A JP2 Header box in a JPM file may be found immediately after the File Type box, Annex B.1.2, or within an Object box, Annex B.4.1.

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 International Standard are as follows:

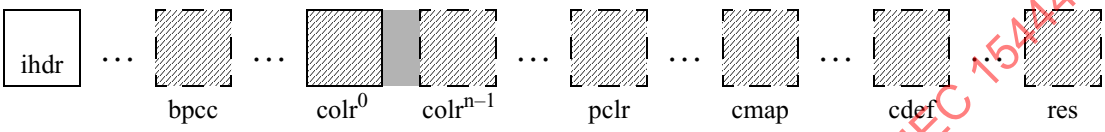


**Figure B-19 Organization of the contents of a JP2 Header box**

**ihdr:** Image Header box. This box specifies information about the object, such as its height and width. Its structure is specified in Annex B.6.2.1. 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. This field contains a Bits Per Component box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2. If the bit depth of all components in the image is the same (in both sign and precision), then this box shall not be found. Otherwise, this box specifies the bit depth of each individual component. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

**colr$^{i}$:** Colour Specification box. This box specifies the colourspace of the decompressed image. Its structure is specified in Annex B.6.2.2. This box may be found anywhere in the JP2 Header box provided that it comes after the Image Header box.

**pclr:** Palette box. This box defines the palette to use to create multiple components from a single component. This field contains a Palette box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.4. 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 field contains a Component Mapping box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.5. This box may be found any where in the JP2 Header box provided that it comes after the Image Header box.

**cdef:** Channel Definition box. This field contains a Channel Definition box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.6. 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 is optional and may be used to specify the capture and/or default display grid resolutions of the object. This field may contain a Resolution box as specified in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.7. 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 and a Colour Specification box, and may contain a Bits Per Component box, a Palette box, a Component Mapping box, a Channel Definition box and a Resolution box.

### B.6.2.1 Image Header box

Box type: ‘ihdr’ (0x69686472)
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’ (0x69686472) and the contents of the box shall have the following format:

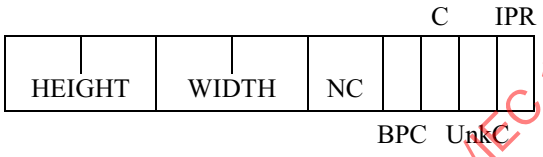


**Figure B-20 Organization of the contents of an Image Header box**

**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. This field is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2.

**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. This field is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2.

**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. This field is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2.

**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 ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2).
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. Legal values of this field are given in Table B-21.

**C:** Compression type. This parameter specifies the compression algorithm used to compress the image data. It is encoded as a 1-byte unsigned integer. See Table B-24 for legal values of C.

**UnkC:** Colourspace Unknown. This field specifies if the actual colourspace of 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. This field is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2.

**IPR:** Intellectual Property. This parameter indicates whether this 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 an IPR box as defined in Annex C.2.1. Other values are reserved for ISO use. This field is identical to that defined in ITU-T Rec T.800 | ISO/IEC 15444-1 Annex I.5.3.2.

**Table B-23 Format of the contents of the Image Header box**

| Field name | Size (bits) | Value |
|---|---|---|
| HEIGHT | 32 | 1 — ($2^{32}$-1) |
| WIDTH | 32 | 1 — ($2^{32}$-1) |
| NC | 16 | 1 — 16 384 |
| BPC | 8 | See Table B-21 |
| C | 8 | See Table B-24 |
| UnkC | 8 | 0 — 1 |
| IPR | 8 | 0 — 1 |