International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.805

(01/2012)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Still-image compression – JPEG 2000

Information technology – JPEG 2000 image coding system: Compound image file format

Recommendation ITU-T T.805

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

Facsimile – Framework	T.0–T.19
Still-image compression – Test charts	T.20–T.29
Facsimile – Group 3 protocols	T.30–T.39
Colour representation	T.40–T.49
Character coding	T.50–T.59
Facsimile – Group 4 protocols	T.60–T.69
Telematic services – Framework	T.70–T.79
Still-image compression – JPEG-1, Bi-level and JBIG	T.80–T.89
Telematic services – ISDN Terminals and protocols	T.90–T.99
Videotext – Framework	T.100–T.109
Data protocols for multimedia conferencing	T.120–T.149
Telewriting	T.150–T.159
Multimedia and hypermedia framework	T.170–T.189
Cooperative document handling	T.190–T.199
Telematic services – Interworking	T.300–T.399
Open document architecture	T.400–T.429
Document transfer and manipulation	T.430–T.449
Document application profile	T.500–T.509
Communication application profile	T.510–T.559
Telematic services – Equipment characteristics	T.560–T.649
Still-image compression – JPEG 2000	**T.800–T.829**
Still-image compression – JPEG XR	T.830–T.849
Still-image compression – JPEG-1 extensions	T.850–T.899

For further details, please refer to the list of ITU-T Recommendations.

INTERNATIONAL STANDARD ISO/IEC 15444-6
RECOMMENDATION ITU-T T.805

Information technology – JPEG 2000 image coding system: Compound image file format

Summary

Recommendation ITU-T T.805| International Standard ISO/IEC 15444-6 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 Recommendation ITU-T 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 Recommendation ITU-T T.801 | ISO/IEC 15444-2 Annex M. This Recommendation | International 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 Recommendation | International Standard. This Recommendation | International Standard:

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

– specifies a mechanism by which multiple images can be combined into a single compound image, based on the mixed raster content (MRC) 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 Recommendation | International Standard.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T T.805	2012-01-13	16

CONTENTS

Page

1 Scope 1

2 Normative references 1

2.1 Identical Recommendations | International Standards 1

2.2 ITU, IEC and ISO references 2

2.3 Additional references 2

3 Definitions 3

4 Abbreviations 4

5 General arrangement 4

5.1 Mixed raster content model 5

5.2 File elements and structure 6

5.3 Hidden text metadata 14

5.4 JPM use scenarios 15

Annex A – Compound image file structure 18

A.1 File identification 18

A.2 File organization 18

A.3 Box definition 20

A.4 Boxes used in a compound image file 20

Annex B – Box definitions 23

B.1 File level boxes 23

B.2 Page level boxes 29

B.3 Layout object level boxes 31

B.4 Object level boxes 33

B.5 JP2 codestream element boxes 36

B.6 General/common boxes 37

Annex C – Metadata 49

C.1 Adding intellectual property rights information in JPM 49

C.2 Adding vendor specific information to the JPM file format 49

Annex D – Profiles 50

D.1 JPM profiles 50

D.2 Decompression profiles 51

Annex E 52

Annex F – Hidden text and annotations storage 53

F.1 Storage of HTX in JPM 53

F.2 Compression of HTX 53

Annex G – Hidden text and annotations types and elements 54

G.1 Overview 54

G.2 Types 55

G.3 Common attributes 56

G.4 Elements 58

Annex H – Hidden text and annotations schema 69

H.1 XML schema 69

Annex I – Hidden text and annotations examples 70

I.1 Example 1 70

I.2 Example 2 72

I.3 Example 3 72

Page

Annex J – Guidelines for constructing URLs for JPM files 90
J.1 Pages and layout objects 90
J.2 Metadata boxes 90
J.3 Labels 90
J.4 Page collections 91
J.5 Page thumbnails 91
J.6 Document thumbnail 91
J.7 Byte ranges 91

Introduction

Recommendation ITU-T T.805 | ISO/IEC 15444-6 was prepared by ITU-T Study Group 16 and Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 29, Coding of audio, picture, multimedia and hypermedia information.

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

– Part 1: Core coding system

– Part 2: Extensions

– Part 3: Motion JPEG 2000

– Part 4: Conformance testing

– Part 5: Reference software

– Part 6: Compound image file format

– Part 8: Secure JPEG 2000

– Part 9: Interactivity tools, APIs and protocols

– Part 10: Extensions for three-dimensional data

– Part 11: Wireless

– Part 12: ISO base media file format

– Part 13: An entry level JPEG 2000 encoder

– Part 14: XML structural representation and reference

INTERNATIONAL STANDARD

RECOMMENDATION ITU-T

Information technology – JPEG 2000 image coding system: Compound image file format

1 Scope

This Recommendation | International Standard defines a normative but optional file format for storing compound images using the JPEG 2000 file format family architecture. This format is an extension of the JP2 file format defined in Rec. ITU-T 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 Rec. ITU-T T.801 | ISO/IEC 15444-2 Annex M. This Recommendation | International 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 Recommendation | International Standard. This Recommendation | International Standard:

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

– specifies a mechanism by which multiple images can be combined into a single compound image, based on the mixed raster content (MRC) 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 Recommendation | International Standard.

2 Normative references

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

2.1 Identical Recommendations | International Standards

– Recommendation ITU-T T.44 (1999) | ISO/IEC 16485/Amd 1:2000, *Information technology – Mixed Raster Content (MRC).*

– Recommendation ITU-T T.44 Amd.1 (1999) | ISO/IEC 16485:2000 Amd 1, *Accommodation of new Annex B.*

– Recommendation ITU-T T.45 (2000), *Run-length Colour Encoding*.

– Recommendation ITU-T T.50 (1992) | ISO/IEC 646:1991, *International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5) – Information technology – 7-bit coded character set for information interchange*.

– Recommendation ITU-T T.81 (1992) | ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*.

– Recommendation ITU-T T.82 (1993) | ISO/IEC 11544:1993, *Information technology – Coded representation of picture and audio information – Progressive bi-level image compression.*

– Recommendation ITU-T T.83 (1994) | ISO/IEC 10918-2:1995, *Information technology – Digital compression and coding of continuous-tone still images: Compliance testing*.

– Recommendation ITU-T T.84 (1996) | ISO/IEC 10918-3:1997, *Information technology – Digital compression and coding of continuous-tone still images: Extensions*.

– Recommendation ITU-T T.84 Amendment 1 (1999) | ISO/IEC 10918-3:1997/Amd 1:1999, *Provisions to allow registration of new compression types and versions in the SPIFF header*.
– Recommendation ITU-T T.86 (1998) | 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)*.
– Recommendation ITU-T T.87 (1998) | ISO/IEC 14495-1:2000, *Information technology – Lossless and near-lossless compression of continuous-tone still images – Baseline*.
– Recommendation ITU-T T.88 (2000) | ISO/IEC 14492:2001, *Information technology – Lossy/lossless coding of bi-level images*.
– Recommendation ITU-T T.800 (2002) | ISO/IEC 15444-1:2004, *Information technology – JPEG 2000 image coding system: Core coding system*.
– Recommendation ITU-T T.801 (2002) | ISO/IEC 15444-2:2004, *Information technology – JPEG 2000 image coding system: Extensions*.
– Recommendation ITU-T T.803 (2002) | ISO/IEC 15444-4:2004, *Information technology – JPEG 2000 image coding system: Conformance testing*.

2.2 ITU, IEC and ISO references

– Recommendation ITU-T T.4 (2003), *Standardization of Group 3 facsimile terminals for document transmission*.
– Recommendation ITU-T T.6 (1988), *Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus*.
– Recommendation ITU-T T.42 (2003), *Continuous-tone colour representation method for facsimile*.
– Recommendation ITU-T T.89 (2001), *Application profiles for Recommendation T.88 – Lossy/lossless coding of bi-level images (JBIG2) for facsimile*.
– 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.1:2003, *Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB.*
– 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*.
– 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 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)*.

2.3 Additional references

– IEEE Std. 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*.
– IETF RFC 1766 (1995), *Tags for the Identification of Languages*.
– IETF RFC 1950 (1996), *ZLIB Compressed Data Format Specification version 3.3*.
– IETF RFC 1951 (1996), *DEFLATE Compressed Data Format Specification version 1.3*.
– IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.
– IETF RFC 2279 (1998), *UTF-8, a transformation format of ISO 10646*.
– IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.
– W3C (2008), *Cascading Style Sheets, level 1 (CSS1) Specification.*<http://www.w3.org/TR/REC-CSS1/>

– W3C (2011), *Cascading Style Sheets, level 2, revision 1 (CSS2) Specification*. <http://www.w3.org/TR/REC-CSS2>
– W3C (2008), *Extensible Markup Language (XML) 1.0, Fifth Edition*. <http://www.w3.org/TR/REC-xml>
– W3C (1999), *HTML 4.01 Specification*. <http://www.w3.org/TR/html401>
– W3C (2002), *XHTML™ 1.0 Extensible HyperText Markup Language, Second Edition*. <http://www.w3.org/TR/xhtml1>
– W3C (2004), *XML Schema Part 0: Primer, Second Edition*. <http://www.w3.org/TR/xmlschema-0>
– W3C (2004), *XML Schema Part 1: Structures, Second Edition*. <http://www.w3.org/TR/xmlschema-1>
– W3C (2004), *XML Schema Part 2: Datatypes, Second Edition*. <http://www.w3.org/TR/xmlschema-2>
– ICC.1:1998-09, *International Color Consortium, File Format for Color Profiles*.

3 Definitions

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

3.1 **annotation**: Particular region of a page in a JPM document that has associated a URL reference, a note or a highlight.

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

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

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

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

3.6 **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.

3.7 **compressed hidden text XML**: Hidden text XML data compressed using the mechanisms defined in clause F.2.

3.8 **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.

3.9 **fragment**: A portion of the codestream for an image. Clause 5.2.5 describes fragment usage.

3.10 **hidden text**: Symbolic representation for the characters and words found in an image.

3.11 **hidden text UUID box**: UUID box containing compressed hidden text XML.

3.12 **hidden text XML**: XML data which describe hidden text and annotations for a single page in a JPM file and which conform to the schema in Annex H.

3.13 **hidden text XML schema**: XML schema for hidden text XML, as defined in clause H.1.

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

3.15 **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.

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

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

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

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

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

3.21 **MRC**: Mixed raster content; a multi-layer imaging model described in Rec. ITU-T T.44 | ISO/IEC 16485.

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

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

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

3.25 **PageImage**: The image created by rendering the BasePage with the layout objects. The PageImagek are the images created by rendering the BasePage with the first k layout objects.

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

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

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

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply. The abbreviations defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 4 also apply.

DPI	Dots per inch
HTX	Hidden Text XML
IPR	Intellectual Property Rights
JP2	JPEG 2000 File Format defined in Rec. ITU-T T.800 | ISO/IEC 15444-1
JPX	JPEG 2000 File Format defined in Rec. ITU-T T.801 | ISO/IEC 15444-2; JPEG 2000 File Format Extended
JPM	JPEG 2000 File Format defined in this International Standard; JPEG 2000 File Format – Multilayer
MRC	Mixed Raster Content
PPCLoc	Primary Page Collection Locator
UUID	Universal Unique Identifier

5 General arrangement

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 clauses 3 and 4 of Rec. ITU-T 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 Rec. 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 clause 5.2.2.1. 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 a 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 5.2.3. 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.

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

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

$$\mathrm{PageImage}[c][x,y] = \mathrm{PageImage}_0[c][x,y] \tag{3}$$

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 mth layout object's mask M and image I respectively, and $s_m = 2^{\text{bit depth of Mm}} - 1$. M, I and BasePage are defined in clauses 5.2.2.1 and 5.2.3. 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.

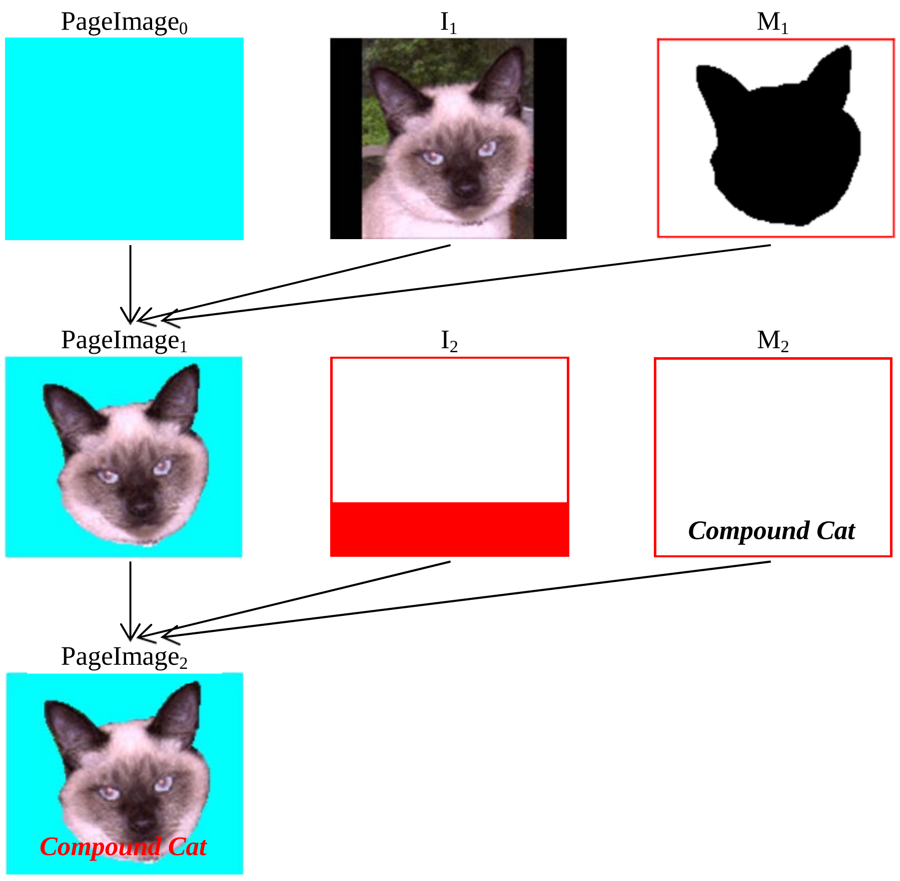


Figure 1 – Example compound document

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 5.2.1.3). 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 5.2.1.3).

NOTE – Figure 2 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.

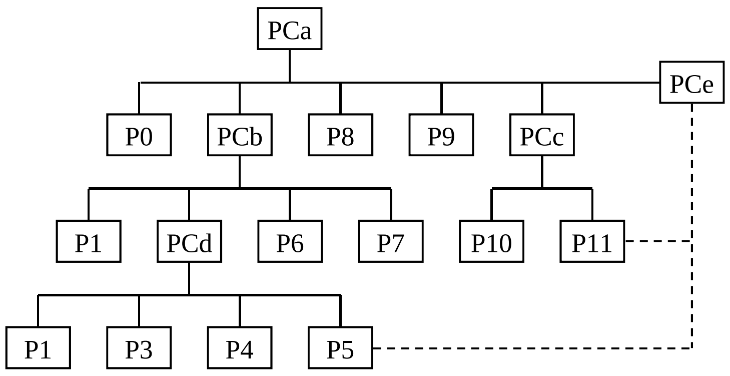


Figure 2 – Example of page collections and pages

5.2.1.1 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.

5.2.1.2 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.

5.2.1.3 Auxiliary page collection

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 labelled by means of a label box in order to be useful to a receiving application. If they are labelled, 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.

5.2.2 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 clause 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.

5.2.2.1 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 the 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.

5.2.3 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 implementer 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 clauses 5.2.3.2 and 5.2.3.4.

Clause 5.2.3.1 describes the special case of generating a mask M and an image I from a JBIG 2 codestream with an associated Recommendation ITU-T 45 encoding of colour tags.

5.2.3.1 Colour tagged JBIG 2 layout objects

If the layout object box contains an image object box with a compression type of Recommendation ITU-T T.45 (Run Length) coding then it must also contain a mask object box with a compression type of Recommendation ITU-T 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 clause 5.2.3.3.1, m_orig is the image obtained by decompressing the JBIG 2 codestream associated with the mask object box.

In clause 5.2.3.4.1, 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 Recommendation ITU-T T.45.

5.2.3.2 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.

Clauses 5.2.3.3.1 – 5.2.3.3.5 define M when a codestream is associated with a mask object box or a combined image/mask object box in the layout object box.

1. The first stage in generating M, described in clause 5.2.3.3.1, 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.
2. The second stage, described in clause 5.2.3.3.2, is to scale the mask.
3. The third stage, described in clause 5.2.3.3.3, is to clip the mask from the top and from the left.
4. The fourth stage, described in clause 5.2.3.3.4, is to position the mask on the page.
5. The fifth and final stage, described in clause 5.2.3.3.5, 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.

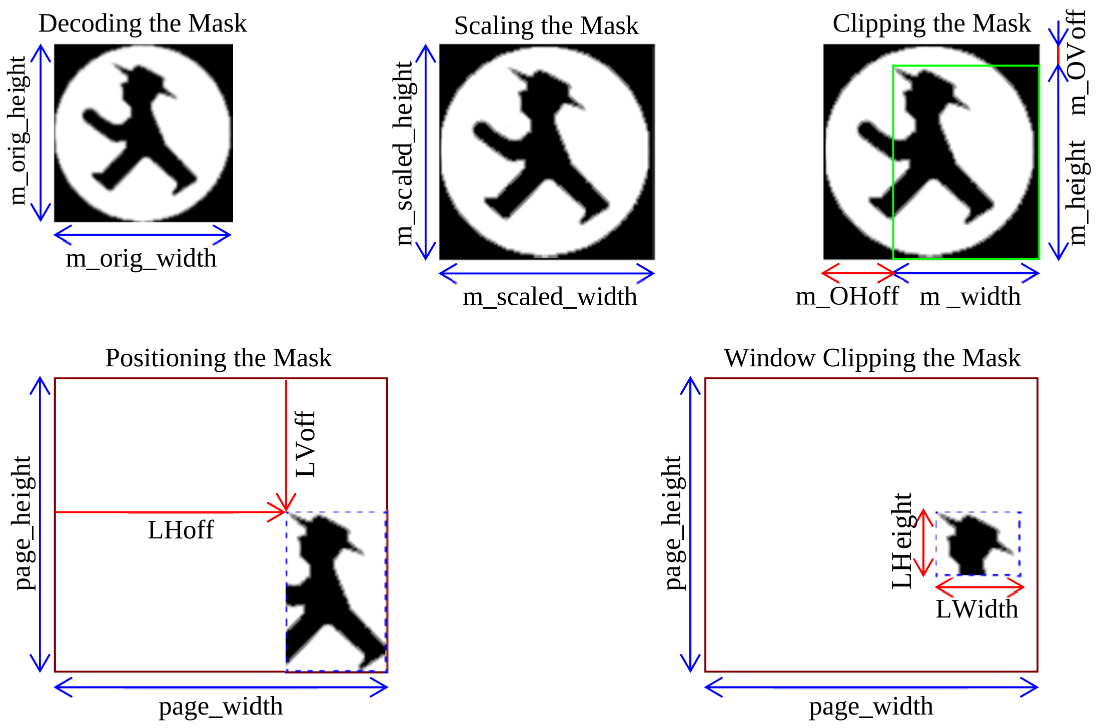


Figure 3 – Example of generating M

5.2.3.2.1 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:

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

5.2.3.2.2 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:

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

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.

5.2.3.2.3 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:

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

5.2.3.2.4 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:

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

5.2.3.2.5 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:

$$\text{M}[0][x,y] = \begin{cases} \text{m\_pos}[0][x,y] & 0 \le x - \text{LHoff} < \text{m\_width and } 0 \le y - \text{LVoff} < \text{m\_height} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

5.2.3.3 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.

Clauses 5.2.3.4.1 – 5.2.3.4.5 define I when a codestream is associated with an image object box or a combined image/mask object box in the layout object box.

1. The first stage in generating I, described in clause 5.2.3.4.1, is to decode the image object and convert the image to the spc colourspace.
2. The second stage, described in clause 5.2.3.4.2, is to scale the image.
3. The third stage, described in clause 5.2.3.4.3, is to clip the image from the top and from the left.
4. The fourth stage, described in clause 5.2.3.4.4, is to position the image on the page.
5. The fifth and final stage, described in clause 5.2.3.4.5, is to clip the image to the layout window.

These stages, as with those used 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.

5.2.3.3.1 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.

5.2.3.3.2 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-byresolution decompression feature to obtain a power-of-two reduced resolution image for scaling.

5.2.3.3.3 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:

$$\text{i\_clipped}[0][x,y] = \text{i\_scaled}[0][x + \text{i\_OHoff}, y + \text{i\_OVoff}] \quad (10)$$

5.2.3.3.4 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:

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

5.2.3.3.5 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:

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

5.2.4 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 Rec. ITU-T 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.

5.2.5 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 that is required to decode the codestream, such as width and height, compression type or bit depth is stored in fields in the object box.

5.2.6 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 a shared-data reference box and a 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 (clause B.1.8), or a data reference including an offset and length, in the case of a cross-reference box (clause 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.

5.2.7 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 Hidden text metadata

Hidden text metadata is data representing the text, text elements and text flow associated with an image. In the context of this standard, hidden text is associated with a particular region of a page in a JPM document.

Common uses for hidden text include text searching and highlighting, cut-and-paste, and text-to-speech processing. Hidden text describes the flow of the text on a page as well as the text elements.

JPM allows a rich, multiple content-type representation of a document. Each region of a page may be encoded with a compression technique best suited to its characteristics. In regions containing text, high fidelity reproduction of the source image is retained by not replacing the text regions with a character-based rendition through OCR, but rather by using advanced coding methods such as JBIG2. Even OCR results with a 99 percent accuracy contain substantial numbers of errors per page which require expensive human labour to correct. The searchable nature of a character-based rendition can be obtained instead by associating hidden "dirty OCR" results with the corresponding text image. This standard defines a format for hidden text metadata.

A key issue with hidden text is capturing the ambiguities seen by the OCR engine in a way that allows properly-constructed search engines to find whether and where a given word might be present in a text image. Properly captured, this information provides nearly as much searching precision as an approach using human-corrected "clean OCR" data, but at much lower cost. Search results are most useful where there are fewer false positives to weed through. Intelligent search engines can take account of such data as confidence and alternate characters or alternate words to appropriately alter the ranking of search hits on less certain characters.

In many cases, true ambiguity exists in the image and it would confuse a human observer as well. In these cases, saving confidence values for characters and their alternatives or describing several alternative parsings of a string of characters into words can amount to saving the state of the OCR process to allow the problem to be revisited in a later stage, perhaps by a different engine or by access to first a general dictionary and then a set of more specialized dictionaries.

As a last step, when a person is presented with the search results, they can dismiss a given search hit by comparison to the actual image data for a character or word. For this purpose (and to allow later-stage OCR processes to resume analysis on the image), bounding box rectangles can be defined for all the elements of the hidden text such as characters, words, lines, paragraphs and regions. By indicating a container relationship among these items, intelligent navigation and text selection can occur at character, word, line, paragraph boundaries. A reading order through these rectangles can be defined for what was in the image just a random placement of unrelated glyphs.

While it is primarily designed for use by machines such as search engines, the hidden text can also serve as a crude (if "dirty") or adequate (if "clean") alternate representation for an image region to allow it to display on character-based devices (such as mobile phones) or small-area graphics devices (such as PDAs).

Annotations are added to the document typically with a WYSIWYG editor to indicate URL references, notes, and to highlight key sections of the document text. Each annotation is associated with a particular region of a page in a JPM document.

XML is used for hidden text and annotations because it is a format widely used to store structured information, and can be machine processed.

5.4 JPM use scenarios

5.4.1 Self-contained files and files with external references

This clause is an informative section. 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 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.

5.4.2 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.

5.4.3 Page collection example

The main page collection of a very large multi-volume set of books such as an encyclopaedia 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 encyclopaedia over on the server via an external reference. By navigating the "search results" page collection, the application can let the user go to the 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 encyclopaedia 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.

5.4.4 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.

5.4.5 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 URI 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 as being 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.

5.4.6 Example of scanner capture

In addition to supporting a variety of streaming and remote browsing behaviours, the file format can support data streams 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 is also 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

Compound image file structure

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

This annex describes the structure and organization of a JPM file. A JPM file uses the file format architecture specified in Rec. ITU-T 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; the brand is defined in Rec. ITU-T 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 Rec. ITU-T 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 (Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 (clause B.1.5) or cross-reference boxes (clause 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.

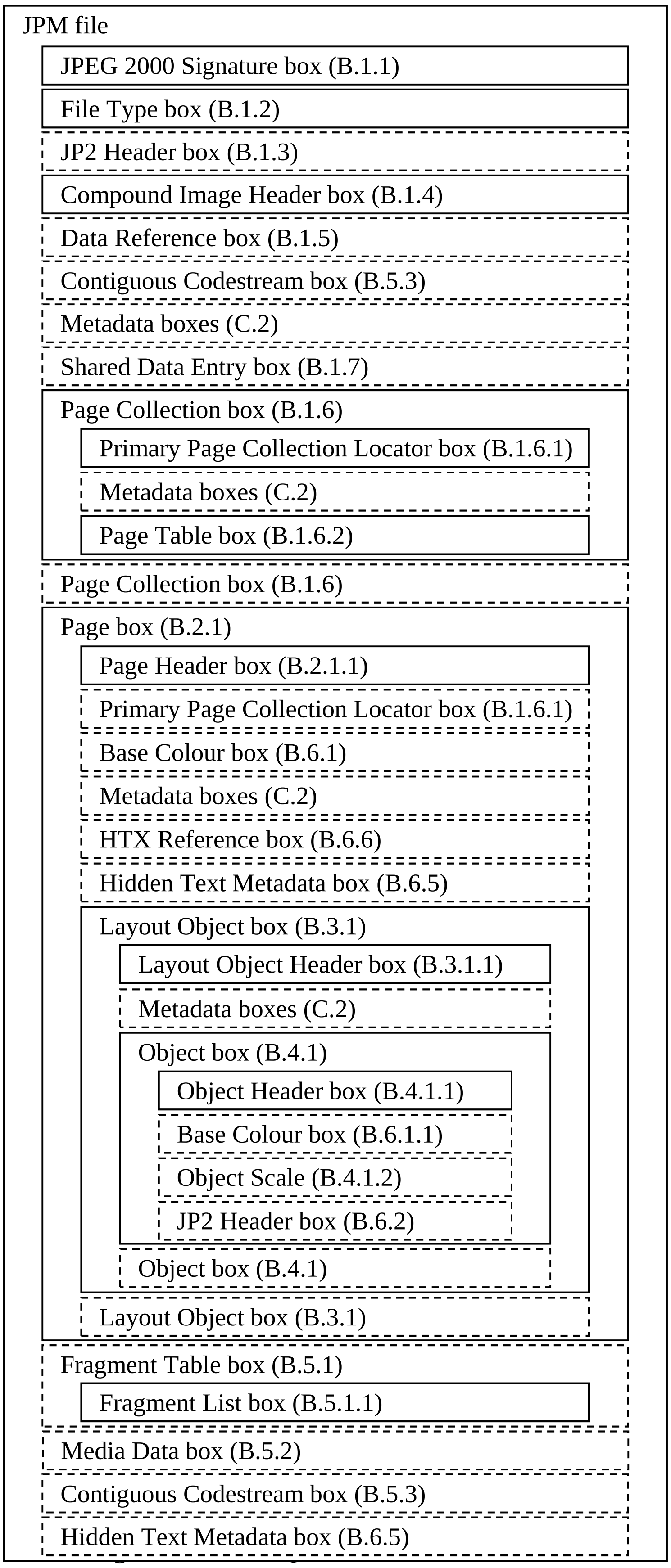

JPM file
JPEG 2000 Signature box (B.1.1)
File Type box (B.1.2)
JP2 Header box (B.1.3)
Compound Image Header box (B.1.4)
Data Reference box (B.1.5)
Contiguous Codestream box (B.5.3)
Metadata boxes (C.2)
Shared Data Entry box (B.1.7)
Page Collection box (B.1.6)
Primary Page Collection Locator box (B.1.6.1)
Metadata boxes (C.2)
Page Table box (B.1.6.2)
Page Collection box (B.1.6)
Page box (B.2.1)
Page Header box (B.2.1.1)
Primary Page Collection Locator box (B.1.6.1)
Base Colour box (B.6.1)
Metadata boxes (C.2)
HTX Reference box (B.6.6)
Hidden Text Metadata box (B.6.5)
Layout Object box (B.3.1)
Layout Object Header box (B.3.1.1)
Metadata boxes (C.2)
Object box (B.4.1)
Object Header box (B.4.1.1)
Base Colour box (B.6.1.1)
Object Scale (B.4.1.2)
JP2 Header box (B.6.2)
Object box (B.4.1)
Layout Object box (B.3.1)
Fragment Table box (B.5.1)
Fragment List box (B.5.1.1)
Media Data box (B.5.2)
Contiguous Codestream box (B.5.3)
Hidden Text Metadata box (B.6.5)

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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause I.4.

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

The following additional information is provided in the box definitions.

Box type: four character code and hexadecimal code for the box type

Container: parent box name or file for the box

Mandatory: Yes or no

Quantity: number of boxes in the container

Location: location of the box

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 names	Types	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.

Table A.1 – Boxes defined or referenced within this International Standard

Box names	Types	Superbox	Comments (informative)
Compound image header	'mhdr' (0x6D68 6472)	No	This box contains general information about the compound image file, such as profile and version.
Contiguous codestream	'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.
Hidden text metadata	'htxb' (0x68747862)	Yes	This optional box contains hidden text and annotations.
HTX reference box	'phtx' (0x70687478)	No	This optional box can be used to point to hidden text metadata box contents at top file level.
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.
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 codestream data.
Page collection	'pcol' (0x7063 6F6C)	No	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.

Table A.1 – Boxes defined or referenced within this International Standard

Box names	Types	Superbox	Comments (informative)
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

Box definitions

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

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 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 CLi field in the file type box, and shall contain the value 'jpm\040' in one of the CLi 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 clause 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 clause 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.

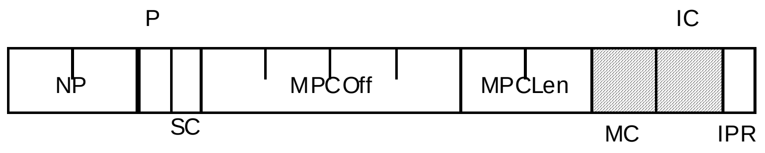


Figure B.1 – Organization of the contents of a compound image header box

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

Table B.2 – Mask coders

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

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. 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.

Table B.3 – Image coders

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

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 clause 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:

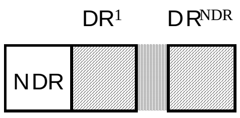


Figure B.2 – Organization of the contents of a data reference box

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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 DRi fields are 1 based; a data reference of 1 in a DRi field within a fragment list box or page table box specifies the first data reference URL contained within the data reference box. A data reference value of 0 is a special case that indicates that the reference is to data contained within this file itself.

Table B.5 – Format of the contents of the data reference box

Field name	Size (bits)	Value
NDR	16	0 – 65 535
DRi	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 clause 5.2.1.2, 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 clause 5.2.1.

The type of a page collection box shall be 'pcol' (0x70636F6C). This box contains the following fields:

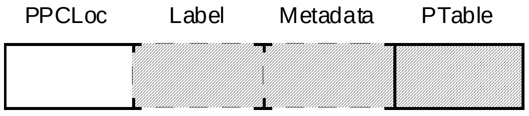


Figure B.3 – Organization of the contents of a page collection box

PPCLoc:	Primary page collection locator box. This box contains a page collection locator box as specified in clause B.1.6.1.
Label:	Label box. This optional field may contain a label box as specified in clause 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 clause 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 clause 5.2.1.

The type of a primary page collection locator box shall be 'ppcl' (0x7070636C). This box contains the following fields:

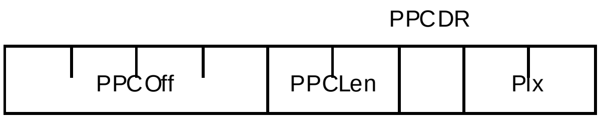


Figure B.4 – Organization of the contents of a primary page collection locator box

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:

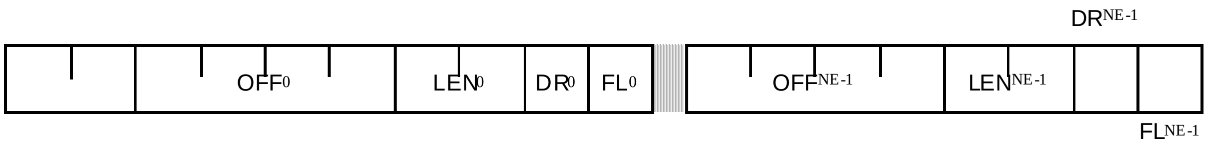


Figure B.5 – Organization of the contents of a page table box

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.
OFFi:	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.
LENi:	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.
DRi:	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.
FLi:	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

Values (bits) **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	Offset to page or page collection box containing no metadata.
xxxx 1xxx	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	$1 – (2^{32}–1)$
OFFi	64	$12 – (2^{64}–1)$
LENi	32	$0 – (2^{32}–1)$
DRi	16	0 – 65 535
FLi	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:

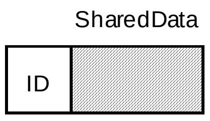


Figure B.6 – Organization of the contents of a shared-data entry box

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:

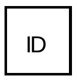


Figure B.7 – Organization of the contents of a shared-data reference box

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:

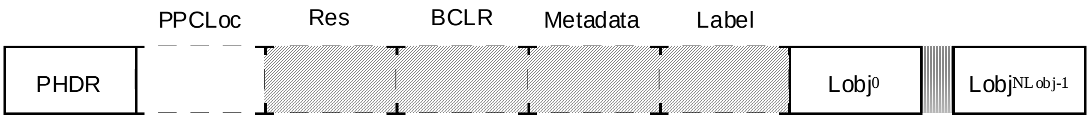


Figure B.8 – Organization of the contents of a page box

PHDR:	Page header box. This field contains a layout object header box as specified in clause B.2.1.1.
PPCLoc:	Primary page collection locator box. This box contains a page collection locator box as specified in clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 clause 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 clause 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 clause 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:

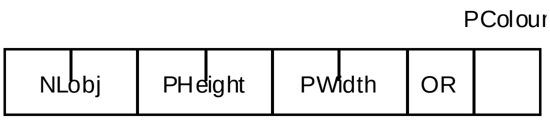


Figure B.9 – Organization of the contents of a page header box

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
other values	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 (clause B.6.1)
other values	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:

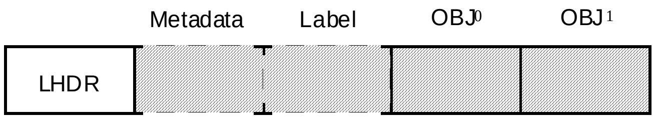


Figure B.10 – Organization of the contents of a layout object box

LHDR: Layout object header box. This field contains a layout object header box as specified in clause B.3.1.1.

Metadata:	Optional metadata boxes.
Label:	Label box. This field may contain a label box as specified in clause 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 clause B.4.1.
OBJ^1:	Object box; the second object box. This field contains an object box as specified in clause 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:

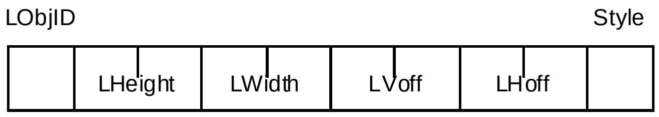


Figure B.11 – Organization of the contents of a layout object header box

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 5.2.4). 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.
other values	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:

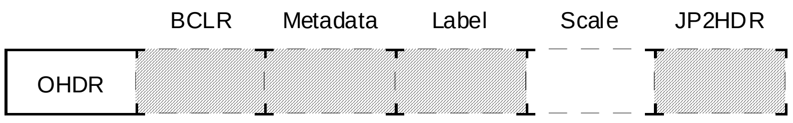


Figure B.12 – Organization of the contents of an object box

OHDR:	Object header box. This field contains an object header box as specified in clause B.4.1.1.
BCLR:	Optional base colour box. This optional field may contain a base colour box as specified in clause 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 clause 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 clause 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 clause 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 the 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 an object header box shall be 'ohdr' (0x6F686472). This box contains the following fields:

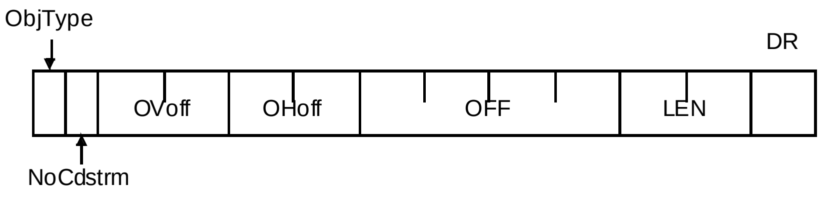


Figure B.13 – Organization of the contents of an object header box

ObjType: Object type. This 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
other values	All other values reserved

NoCdstrm: 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 NoCdstrm values

Value	Meaning
0	The object contains a codestream
1	The object does not contain a codestream
other values	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 the 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
NoCdstrm	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 $\frac{\text{VRN}}{\text{VRD}}$, and the horizontal scaling is by the ratio $\frac{\text{HRN}}{\text{HRD}}$. See clause 5.2.3 for a full description of the scaling operation.

The type of an object scale box shall be 'scal' (0x7363616C). This box contains the following fields:

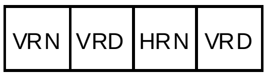


Figure B.14 – Organization of the contents of an object header box

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 clause 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 JP2 codestream element boxes

B.5.1 Fragment table box (superbox)

Box type: 'ftbl' (0x6674626C)

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' (0x6674626C). This box contains the following field:

flst:	Fragment List. This field contains a fragment list box as specified in clause B.5.2.

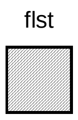


Figure B.15 – Organization of the contents of a fragment table box

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 which 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:

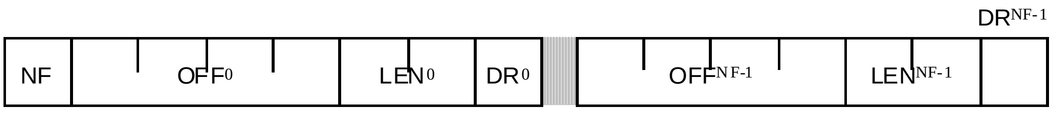


Figure B.16 – Organization of the contents of a fragment list box

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 – 65535

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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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:

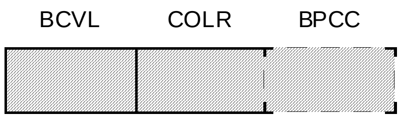


Figure B.17 – Organization of the contents of a base colour box

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 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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:

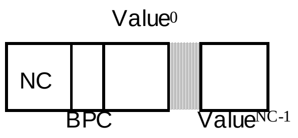


Figure B.18 – Organization of the contents of a base colour value box

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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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

Values (bits) MSB LSB	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.
other values	Reserved for ISO use

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
Valuei	16	0 – 65 535
DRi	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 Rec. ITU-T T.800 | ISO/IEC 15444- 1 clause I.5.3.

A JP2 header box in a JPM file may be found immediately after the file type box, clause B.1.2, or within an object box, clause 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:

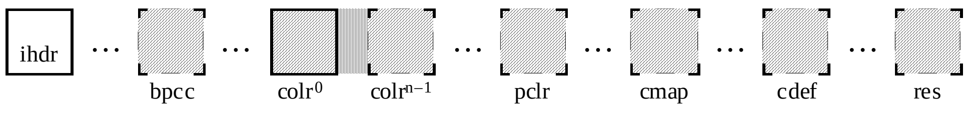


Figure B.19 – Organization of the contents of a JP2 header box

ihdr: Image header box. This box specifies information about the object, such as its height and width. Its structure is specified in clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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.

colri: Colour specification box. This box specifies the colourspace of the decompressed image. Its structure is specified in clause 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 be used to create multiple components from a single component. This field contains a palette box as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause I.5.3.5. This box may be found anywhere in the JP2 header box provided that it comes after the image header box.
cdef:	Channel definition box. This field contains a channel definition box as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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:

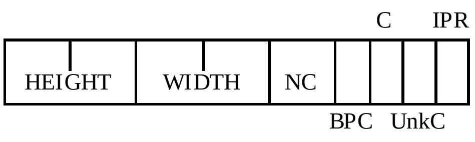


Figure B.20 – Organization of the contents of an image header box

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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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 clause C.2.1. Other values are reserved for ISO use. This field is identical to that defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 clause 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

Table B.24 – Legal C values for the image header box

Value	**Meaning**
0	Uncompressed. Picture data is stored in component interleaved format, encoded at the bit depth as specified by the BPC field. This value is only permitted for codestreams where all components are encoded at the same bit depth. When the bit depth of each component is not 8, sample values shall be packed into bytes so that no bits are unused between samples. However, each sample shall begin on a byte boundary and padding bits having value zero shall be inserted after the last sample of a scan line as necessary to fill out the last byte of the scan line. Simple values appear in component-interleaved order. When multiple sample values are packed into a byte, the first sample shall appear in the most significant bits of the byte. When a sample is larger than a byte, its most significant bit shall appear in earlier bytes.
1	Rec. ITU-T T.4, the basic algorithm known as MH (Modified Huffman). This value is only permitted for bi-level images.
2	Rec. ITU-T T.4, commonly known as MR (Modified READ). This value is only permitted for bi-level images.
3	Rec. ITU-T T.6, commonly known as MMR (Modified Modified READ). This value is only permitted for bi-level images.
4	Rec. ITU-T T.82 | ISO/IEC 11544. Commonly known as JBIG. This value is only permitted for bi-level images.
5	Rec. ITU-T T.81 | ISO/IEC 10918-1 or Rec. ITU-T T.84 | ISO/IEC 10918-3. Commonly known as JPEG. This compressed image stream shall conform to the syntax of interchange format for compressed image data as specified in the aforementioned standards. This value is only permitted for continuous tone, greyscale or colour images.
6	Rec. ITU-T T.87 (JPEG-LS).
7	ISO/IEC 15444-1 (JPEG 2000 Part 1).
8	Rec. ITU-T T.88 (JBIG2) coding.
9	Rec. ITU-T T.82 | ISO/IEC 11544. Commonly known as JBIG. This value is permitted for any image permitted by the JBIG standard.
other values	Reserved for ISO use.

B.6.2.2 Colour specification box

Box type: 'colr' (0x636F6C72)

Mandatory: No

Location: Anywhere after image header box in a JP2 header box

A colour specification box defines a method by which an application can interpret the colourspace of the decompressed image data. This colour specification is to be applied to the image data after it has been decompressed and after any reverse decorrelating component transform has been applied to the data.

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

The type of a colour specification box shall be 'colr' (0x636F6C72). This box contains the following fields:

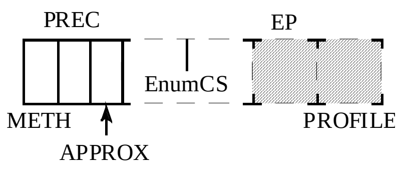


Figure B.21 – Organization of the contents of a colour specification box in JPM

METH:	Specification method. This field specifies the method used by this colour specification box to define the colourspace of the image. This field is encoded as a 1-byte unsigned integer. The value for this field in a JPM file shall be 1 or 2, as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 clause I.5.3.3.
PREC:	Precedence. This field is reserved for ISO use and the value shall be set to zero; however conforming readers shall ignore the value of the field. This field is specified as a signed 1-byte integer.
APPROX:	Colourspace approximation. This field specifies the extent to which this colour specification method approximates the "correct" definition of the colourspace. The value of this field shall be set to zero; however, conforming readers shall ignore the value of this field. Other values are reserved for other ISO use. This field is specified as a 1-byte unsigned integer.
EnumCS:	Enumerated colourspace. This field specifies the colourspace of the image using integer codes. To correctly interpret the colour of an image using an enumerated colourspace, the application must know the definition of that colourspace internally. This field contains a 4-byte big endian unsigned integer value indicating the colourspace of the image. Valid EnumCS values are defined in Table B.26. A mask object must use the greyscale or bi-level enumerated colourspace.
EP:	Enumerated parameters; this field contains a series of parameters that augment the generic colourspace definition specified by EnumCS. Together, the EnumCS and EP fields describe the colourspace and how that colourspace has been encoded in the JPM file. If a value of EP is not defined for a particular value of EnumCS, then the length of the EP field for that EnumCS value shall be zero, indicating that the EnumCS value alone describes the colourspace or that default values are used as defined by the referenced colourspace. This field shall not exist if the value of the METH field is 2.
PROFILE:	ICC profile. This field contains a valid ICC profile, as specified by the ICC profile format specification, which specifies the transformation of the decompressed image data into the PCS. This field shall not exist if the value of the METH field is 1. If the value of the METH field is 2, then the ICC profile shall conform to the monochrome input profile class or the three-component matrix-based input profile class as defined in ICC.1:1998-09.

Table B.25 – Legal METH values

Value	Meaning
1	**Enumerated colourspace**. This colour specification box contains the enumerated value of the colour-space of this image. The enumerated value is found in the EnumCS field in this box. If the value of the METH field is 1, then the EnumCS shall exist in this box immediately following the APPROX field.
2	**Restricted ICC profile**. This colour specification box contains an ICC profile in the PROFILE field. This profile shall specify the transformation needed to convert the decompressed image data into the PCSXYZ space, and shall conform to either the monochrome input or three-component matrix-based input profile class, and contain all the required tags specified therein, as defined in ICC.1:1998-09. As such, the value of the profile connection space field in the profile header in the embedded profile shall be 'XYZ\040' (0x5859 5A20) indicating that the output colourspace of the profile is in the XYZ colourspace. Any private tags in the ICC profile shall not change the visual appearance of an image processed using this ICC profile. The components from the codestream may have a range greater than the input range of the tone reproduction curve (TRC) of the ICC profile. Any decoded values should be clipped to the limits of the TRC before processing the image through the ICC profile. For example, negative sample values of signed components may be clipped to zero before processing the image data through the profile. See Rec. ITU-T T.800 | ISO/IEC 15444-1 clause J.9 for a more detailed description of the legal colourspace transforms, how those transforms are stored in the file, and how to process an image using that transform without using an ICC colour management engine. If the value of METH is 2, then the PROFILE field shall immediately follow the APPROX field and the PRO- FILE field shall be the last field in the box.
other values	Reserved for ISO use. If the value of METH is not 1 or 2, there may be fields in this box following the APPROX field, and a conforming JPM reader shall ignore the entire colour specification box.

Table B.26 – Legal EnumCS values

Value	Meaning
0	**Bi-level**: This value is used to indicate bi-level images. Each image sample is one bit: 0 = white, 1 = black.
3	**YCbCr(2)**: This is the most commonly used format for image data that was originally captured in RGB (uncalibrated format). The colourspace is based on Recommendation ITU-R BT.601-5. The valid ranges of the YCbCr components in this space are [0,255] for Y, and [–128,127] for Cb and Cr (stored with an offset of 128 to convert range to [0,255]). These ranges are different from the ones defined in Recommendation ITU-R BT.601-5. Recommendation ITU-R BT.601-5 specifies a 3x3 matrix transform that can be used to convert these samples into RGB.
14	**CIELab**: The CIE 1976 (L*a*b*) colourspace. A colourspace defined by the CIE (Commission Internationale de l' Eclairage), having approximately equal visually perceptible differences between equally spaced points throughout the space. The three components are L*, or lightness, and a* and b* in chrominance. For this colourspace, additional enumerated parameters are specified in the EP field as specified in clause B.6.2.2.1
16	**sRGB** as defined in IEC 61966-2
17	**Greyscale**: A greyscale space where image luminance is related to code values using the sRGB non-linearity given in Equations (2) through (4) of IEC 61966-2-1 (sRGB) specification: $Y' = Y_{8bit}/255$ (B.1) $\text{for}(Y' \le 0.04045),\ Y_{lin} = Y'/12.92$ (B.2) $\text{for}(Y' > 0.04045),\ Y_{lin} = \left(\frac{Y'+0.055}{1.055}\right)^{2.4}$ (B.3) where Y_{lin} is the linear image luminance value in the range 0.0 to 1.0. The image luminance values should be interpreted relative to the reference conditions in Section 2 of IEC 61966-2-1.
18	**sRGB YCC** as defined by IEC 61966-2-1 Amd. 1. It is not recommend to use ICT or RCT specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 Annex G with sYCC image data. See Rec. ITU-T T.800 | ISO/IEC 15444-1 clause J.15 for guidelines on handling YCC codestreams.

Table B.27 – Format of the contents of the colour specification box

Field name	Size (bits)	Value
METH	8	1 -2
PREC	8	0
APPROX	8	0
EnumCS	32 if METH = 1 0 if METH = 2	See Table B.26 no value
PROFILE	variable	variable
EP	variable	variable

B.6.2.2.1 EP field format for the CIELab colourspace

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

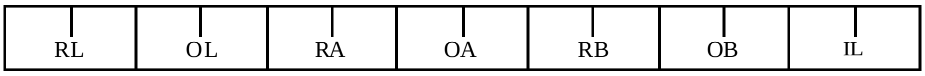


Figure B.22 – Organization of the contents of the EP field for CIELab (EnumCS = 14)

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

$$N_L = \frac{2^{n_L}}{RL} \times L* + OL \qquad (B.4)$$

$$N_a = \frac{2^{n_a}}{RA} \times a* + OA \qquad (B.5)$$

$$N_b = \frac{2^{n_b}}{RB} \times b* + OB \qquad (B.6)$$

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

RL:	Range for L*. This field specifies the RL value from Equation B.4. It is encoded as a 4-byte big endian unsigned integer.
OL:	Offset for L*. This field specifies the OL value from Equation B.4. It is encoded as a 4-byte big endian unsigned integer.
RA:	Range for a*. This field specifies the RA value from Equation B.5. It is encoded as a 4-byte big endian unsigned integer.
OA:	Offset for a*. This field specifies the OA value from Equation B.5. It is encoded as a 4-byte big endian unsigned integer.
RB:	Range for b*. This field specifies the RB value from Equation B.6. It is encoded as a 4-byte big endian unsigned integer.
OB:	Offset for b*. This field specifies the OB value from Equation B.6. It is encoded as a 4-byte big endian unsigned integer.
IL:	Illuminant. This field specifies the illuminant data used in calculating the CIELab values. Rather than specify the XYZ values of the normalizing illuminant, which are used in calculating CIELab, the specification of the illuminant data follows Recommendation ITU-T T.4 Annex E. The illuminant data consists of 4 bytes, identifying the illuminant. In the case of a standard illuminant, the 4 bytes are one of the following in Table B.28.

Table B.28 – Standard illuminant values for CIELab

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

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

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

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

Table B.29 – Format of the contents of the EP field for CIELab (EnumCS = 14)

Field name	Size (bits)	Value
RL	32	$0 – (2^{32}–1)$
OL	32	$0 – (2^{32}–1)$
RA	32	$0 – (2^{32}–1)$
OA	32	$0 – (2^{32}–1)$
RB	32	$0 – (2^{32}–1)$
OB	32	$0 – (2^{32}–1)$
IL	32	See Table B.28

B.6.3 Label box

Box type: 'lbl\040'(0x6C626C20)

Container: Layout Object, Object, Page or Page Collection box

Mandatory: No

Quantity: Any number

Location: Anywhere

The type of a label box shall be 'lbl\040' (0x6C626C20). The contents of the label box are as follows:

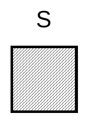


Figure B.23 – Organization of the contents of a label box

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

B.6.4 Cross-reference box

Box type: 'cref' (0x63726566)

Container: See details below

Mandatory: No

Quantity: Any number

Location: Anywhere

If a JPM file contains multiple codestreams or layout objects, it may be useful to share header and metadata information to minimize file size. One mechanism to share such data is to use a cross-reference to the actual header or metadata box in place of the actual data. This is done using a cross-reference box. A JPM file may contain zero or more cross-reference boxes and a cross-reference box shall not point to another cross- reference box.

The type of the cross-reference box shall be 'cref' (0x63726566) and it shall have the following contents:

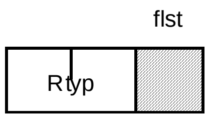


Figure B.24 – Organization of the contents of a cross-reference box

Rtyp: Referenced box type. This field specifies the actual type (as would be found in the TBox field in an actual box header) of the box referenced by this cross-reference box. However, a reader shall not attempt to locate a physically stored box header for the box represented by this cross-reference box, as it is legal to use a cross-reference box to create a new box that is not contiguously contained in other locations within this or other files, and thus the box header may not exist. This field is encoded as a 4-byte value.

flst: Fragment list box. This box specifies the actual locations of the fragments of the referenced box. When those fragments are concatenated, in order, as specified by the fragment list box definition, the resulting byte-stream shall be the entire referenced box and shall not include the box header fields. However, if the referenced box is a superbox, then the offset of the first fragment does point to the box header of the first box contained within the superbox. The format of the fragment list box is specified in clause B.5.1.1.

Table B.30 – Format of the contents of the cross-reference box

Field name	Size (bits)	Value
Rtyp	32	$0 – (2^{32}–1)$
flst	variable	variable

B.6.5 Hidden text metadata box (superbox)

Box type: 'htxb' (0x68747862)

Container: Page box or file

Mandatory: No

Quantity: At most one if the container is the page box, any number if the container is the file

Location: Anywhere in the page box after the page header box if the container is the page box, or anywhere after the file type box if the container is the file.

The hidden text metadata box ('htxb') serves as a container for hidden text data. It is a superbox that may contain an optional label box and must contain one of two box types. It may either contain one XML box containing hidden text metadata, or it may contain one UUID box containing hidden text metadata as specified in clause F.2.

The type of a hidden text metadata box shall be htxb' (0x68747862). The contents of a hidden text metadata box shall be as in Figure B.25:

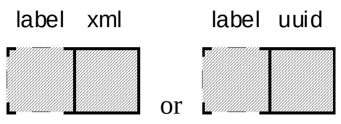


Figure B.25 – Organization of the contents of a hidden text metadata box

B.6.6 HTX reference box

Box type: 'phtx' (0x70687478)

Container: Page box

Mandatory: No

Quantity: At most one

Location: Anywhere in the page box after the page header box

If the hidden text for a page is contained in a hidden text metadata box within the corresponding page box, this box must not appear. If the hidden text for a page is contained in a series of one or more hidden text metadata boxes at the file level, one HTX reference box has to be included in the corresponding page box.

The type of an HTX reference box shall be 'phtx' (0x70687478). The contents of an HTX reference box shall be as in Figure B.26:

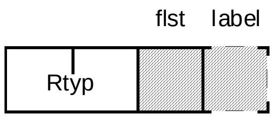


Figure B.26 – Organization of the contents of an HTX reference box

Rtyp:	Referenced box type. This field specifies the actual type (as would be found in the TBox field in an actual box header) of the box referenced by this HTX reference box. However, a reader shall not attempt to locate a physically stored box header for the box represented by this HTX reference box, as it is legal to use an HTX reference box to create a new box that is not contiguously contained in other locations within this or other files, and thus the box header will not exist.
flst:	Fragment list box. This box specifies the actual locations of the fragments of the referenced HTX element. When those fragments are concatenated, in order, as specified by the fragment list box definition, the resulting byte-stream shall be the contents of the referenced HTX element, which contains hidden text data, and shall not include the box header fields. The format of the fragment list box is specified in clause B.5.1.1. If Rtyp is 'uuid' and the UUID signals deflate compression as defined in clause F.2, the number of fragments of the fragment list box must be one.
label:	Label box. This optional box may contain a label box which specifies a label or name for the hidden text of the corresponding page. The structure of a label box is specified in clause B.6.3.

Table B.31 – HTX reference box contents data structure values

Parameter	Size (bits)	Value
Rtyp	32	See Table B.32
flst	Variable	Variable
label	Variable	Variable

Table B.32 – Legal Rtyp values

Value	Meaning
xml\40	The referenced HTX data shall be contained in an XML box as described in Annex F. The XML box is defined in clause I.7.1 of Rec. ITU-T T.800 (2002) | ISO/IEC 15444-1:2004.
uuid	The referenced HTX data shall be contained in a UUID box as described in Annex F. The UUID box is defined in clause I.7.2 of Rec. ITU-T T.800 (2002) | ISO/IEC 15444-1:2004.
other values	All other values reserved

B.6.7 Free box

Box type: 'free' (0x66726565)

Container: Anywhere

Container: Any Mandatory: No

Quantity: Any number

The free box specifies a section of the JPM file that is not currently used and may be overwritten when editing the file. Readers shall ignore all free boxes.

The type of a free box shall be 'free' (0x66726565). The contents of a free box in general are not defined by this International Standard.

Annex C

Metadata

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

A JPM file may contain metadata boxes with intellectual property rights information or vendor specific information, as defined in this annex. Metadata boxes are optional in a JPM file and may be ignored by conforming readers. In addition, a JPM file may contain other metadata boxes not defined in this annex.

C.1 Adding intellectual property rights information in JPM

ISO 15444-1/ITU-T 800 specifies a box type for a box that is devoted to carrying intellectual property rights information. Specifying the box type allows applications to recognize the existence of IPR information in a JPM file. Use and interpretation of this information is beyond the scope of this International Standard. Inclusion of this information in a JPM file is optional for conforming files. The definition of the format of the contents of this box is reserved for ISO. The type of the intellectual property Box shall be 'jp2i' (0x6A70 3269).

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

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

C.2.1 XML boxes

An XML box contains vendor specific information (in XML format) other than the information contained within boxes defined by this International Standard. There may be multiple XML boxes within the file, and those boxes may be found in an object box, layout object box, page box, page collection box, or anywhere at the file level except before the file type box. The XML box is defined in clause I.7.1 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

C.2.2 UUID boxes

A UUID box contains vendor specific information other than the information contained within boxes defined within this International Standard. There may be multiple UUID boxes within the file, and those boxes may be found in an object box, layout object box, page box, page collection box, or anywhere at the file level except before the file type box. The UUID box is defined in clause I.7.2 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

C.2.3 UUID Info boxes (superbox)

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

These boxes, if present, may not provide a complete index for the UUID's in the file, may reference UUID's not used in the file, and possibly may provide multiple references for the same UUID. The UUID Info box is defined in clause I.7.3 of Rec. ITU-T T.800 | ISO/IEC 15444-1.

Annex D

Profiles

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

This annex defines the profiles to which a JPM file can conform. In particular, this annex defines the values of the P field in the compound image header box. A profile defines legal values of various fields in a JPM file.

D.1 JPM profiles

The P field of the compound image header box, defined in clause B.1.4, shows the profile to which the JPM file conforms. A profile defines legal values of various fields in a JPM file. This annex gives the legal values for the P field values specified in the compound image header box.

D.1.1 Web profile

The profile field value for the web profile is 1. The following table gives values for boxes in a JPM file when the profile is in use.

Table D.1 – Web profile values

Box	Field	Value	Comment
Compound image header	MC	xxxx x1xx xx1x xxxx xxx1 xxxx	Rec. ITU-T T.6 (MMR) Rec. ITU-T T.88 (JBIG2 profile) Rec. ITU-T T.800 (JPEG 2000 Part 1)
	IC	xxxx xxx1 xxx1 xxxx xxxx x1xx	Rec. ITU-T T.81 (JPEG) Rec. ITU-T T.800 (JPEG 2000 Part 1) Rec. ITU-T T.45
Image header (in mask object box)	C	3 7 8	Rec. ITU-T T.6 (MMR) Rec. ITU-T T.88 (JBIG2 profile Rec. ITU-T T.800 (JPEG 2000 Part 1)
Image header (in image object box)	C	5 7 10	Rec. ITU-T T.81 (JPEG) Rec. ITU-T T.800 (JPEG 2000 Part 1 Rec. ITU-T T.45
	BPC	–	All image components shall have the same bits per component value
Colour specification (in image object box)	METH	1 2	Enumerated colourspace Restricted ICC profile
	EnumCS	16 17 14 3 18	sRGB Greyscale CIELab YCbCr(2) sRGB sYCC

The mask objects in a JPM file conforming to the web profile can use MMR, JBIG2 or JPEG 2000 compression; only JPEG 2000 can be used when the mask has more than one bit per component. The image objects in a conforming JPM file can use JPEG or JPEG 2000.

In this profile, all components of an image object shall have the same number of bits per component. All image objects within a page box shall have the same colour specification; image objects included by reference within a page box may or may not have the same colour specification as the image objects directly located within the page box.

D.2 Decompression profiles

There are also profiles that determine the use of the compression methods allowed in a JPM file. Table D.2 shows the decompression profiles that are legal in a JPM file.

Table D.2 – Decompression profile values

Compression method	Profile
JPEG 2000	JPEG 2000 Profile 1 (Rec. ITU-T T.800 | ISO/IEC 15444-1), Cclass1 (Rec. ITU-T T.803 | ISO/IEC 15444-4), with maximum image height or width of 16,384.
JPEG	Baseline JPEG (ISO 10918-1)
JBIG2	ITU-T T.88 profiles F3, F4, F5, F6 and F7, or ITU-T T.89, all profiles. When using T.88 profiles F6 or F7, or T.89 profile s 0x0000101 or 0x00000103, no mixing of types within a stripe may occur; there must be 2 or more stripes per page

Annex E

This annex is intentionally left blank.

Annex F

Hidden text and annotations storage

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

F.1 Storage of HTX in JPM

A hidden text XML element is restricted to represent text for a single page. It is stored in a hidden text metadata box as defined in clause B.6.5. The hidden text metadata box either appears within the corresponding page box or is placed at the top level of the file. If placed on the top level, an HTX reference box as defined in clause B.6.6 must be placed in the corresponding page box to point to the hidden text metadata boxes that composes the hidden text of the page.

When a hidden text metadata box is small in size, it is reasonable to place it directly in the page box. In keeping with the usual JPM approach, large objects are generally placed at the top file level. In this case, the much smaller HTX reference box is placed in the page box and points to the actual data. Also in this case a single HTX reference box can point to multiple file level hidden text metadata boxes. This can be used to compose the HTX for many pages from combinations of fixed page content (such as page headers and footers) and variable page content unique to each page.

XML data representing hidden text and annotations is defined using XML 1.0, and conforms to the schemas in Annex H. It shall be referred to as "Hidden Text XML" or HTX.

HTX shall be stored in a hidden text metadata box as defined in clause B.6.5.

The storage of uncompressed HTX may increase file size considerably. In order to minimise the increase in file size, HTX may be compressed using the mechanisms defined in clause F.2.

F.2 Compression of HTX

HTX may be compressed using the zlib format defined in IETF RFC 1950 with DEFLATE compression defined in IETF RFC 1951.

UUID boxes shall be used for the storage of compressed HTX in the JPM file format.

Compressed HTX shall be stored in a UUID box, as defined in clause I.7.3 of Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, with the following contents:

ID This field shall contain the following 16 hexadecimal bytes:

```
c2 f3 66 a4 27 ec 40 c4 a0 9a 7e 65 2f 36 eb 59
```

DATA This field will contain hidden text XML compressed to the DEFLATE format, as specified in clause F.1.

A UUID box with the above content shall be referred to as a ***hidden text UUID box***.

The following URL may be used in a UUID data entry box, as defined in clause I.7.3.2 of Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, to describe the format of the data contained in hidden text UUID boxes:

http://www.jpeg.org/hiddentext/htx.html

Annex G

Hidden text and annotations types and elements

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

G.1 Overview

This section describes each of the HTX types and elements, and how they are to be used and interpreted.

Annex H formally describes the schemas that the hidden text XML must conform to. Here the text is a description of each of the elements, what they are for, how they relate to each other, how often they can occur, how they are to be interpreted.

Hidden text can be encoded using subelements at different levels of detail as described in this section. This can be used to structure the hidden text and give it a text flow in regions, paragraphs, lines, words, etc. Whenever this kind of structured information is not available, the hidden text can be directly put into the appropriate elements, omitting specific positioning of lines inside paragraphs, words inside lines, etc. The following picture gives an overview of the various elements that can be used to store the hidden text of a page:

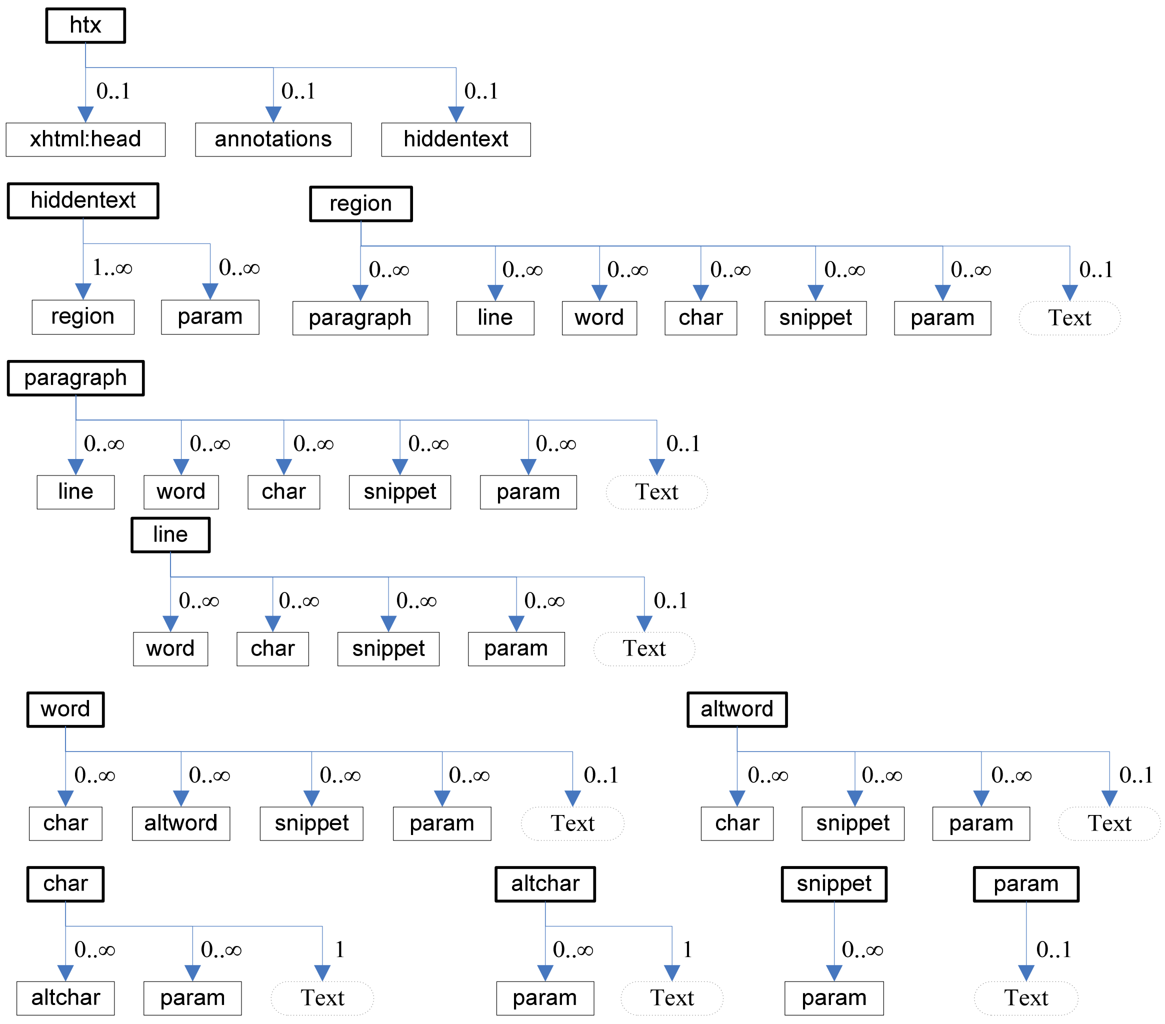


Figure G.1 – Structure of HTX

The hidden text XML schema (see H.1) uses some types and elements defined in the XHTML 1.0 XML schema. See the XHTML 1.0 reference for full details of these types and elements.

The following additional types and elements are defined:

G.2 Types

G.2.1 Shape

The **Shape** type is used to describe the shape of a region in the document and is defined by the following XML schema declaration:

```
<xs:simpleType name="Shape">
  <xs:annotation>
    <xs:documentation>Enumeration of shapes.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="rect"/>
    <xs:enumeration value="poly"/>
  </xs:restriction>
</xs:simpleType>
```

G.2.2 Coordinates

The **Coords** type is used to store a comma separated sequence of non-negative integer values. This type is similar to the XHTML 1.0 **Coords** type but excludes negative and percentage values. The attribute specifies the position and shape of the area. The number and order of values depends on the value of the shape attribute. Possible combinations:

- **rect**: left-x, top-y, right-x, bottom-y.
- **poly**: x1, y1, x2, y2, ..., xN, yN.
  If the first and last x and y coordinate pairs are not the same, user agents must infer an additional coordinate pair to close the polygon.

The **Coords** element is defined by the following XML schema declaration:

```
<xs:simpleType name="Coords">
  <xs:annotation>
    <xs:documentation>
Comma separated list of integer values.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="\d+,\s*\d+(,\s*\d+,\s*\d+)+"/>
  </xs:restriction>
</xs:simpleType>
```

G.2.3 Percentage

A simple type **Percentage** is defined to store a string that holds a percent value indicating the confidence of a hidden text word or character match. **Percentage** is defined as follows:

```
<xs:simpleType name="Percentage">
  <xs:annotation>
    <xs:documentation>Percentage value.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="\d+(\.\d+)?%?"/>
  </xs:restriction>
</xs:simpleType>
```

G.2.4 Angle

A simple type **Angle** is defined to store a string that indicates an angle for use in hidden text. The **Angle** type is defined as follows:

```
<xs:simpleType name="Angle">
  <xs:annotation>
    <xs:documentation>nn for radian measure or nn° for
degree</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[\-\+]?\d+(\.\d+)?°?"/>
  </xs:restriction>
</xs:simpleType>
```

G.2.5 Resolution

A simple type **Resolution** is defined to store a string that indicates a resolution for use with coordinates in hidden text and annotations. The **Resolution** type is defined as follows:

```
<xs:simpleType name="Resolution">
  <xs:annotation>
    <xs:documentation>
Resolution value in dots per inch (dpi). A single number stands for horizontal
and vertical resolution having the same values. Two numbers can be used to
define different resolutions for horizontal (first number) and vertical
(second number).
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="\d+(\.\d+)?(,\*s\d+(\.\d+)?)?"/>
  </xs:restriction>
</xs:simpleType>
```

G.3 Common attributes

G.3.1 Core attributes

Coreattrs, a set of core attributes that are common to most elements, is defined as follows:

```
<xs:attributeGroup name="coreattrs">
  <xs:annotation>
    <xs:documentation>
core attributes common to most elements
  id          document-wide unique id
  class       space separated list of classes
  lang        language code (backwards compatible)
  xml:lang    language code (as per XML 1.0 spec)
  dir         direction for weak/neutral text
  iref        URI of the image corresponding to the region
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="id"    type="xs:ID"       />
  <xs:attribute name="class" type="xs:NMTOKENS" />
  <xs:attribute name="iref"  type="xs:anyURI" />
  <xs:attributeGroup ref="xhtml:i18n"/>
</xs:attributeGroup>
```

The following attributes are members of the **Coreattrs** group:

- **lang** (optional)

An optional attribute of type **LanguageCode** to indicate the default language for text in the hidden text XML. Refer to the XHTML 1.0 specification for further details.

- **xml:lang** (optional)

  An optional attribute of type **xml:lang** to indicate the default language for text in the hidden text XML. Refer to the XHTML 1.0 specification for further details.

- **dir** (optional)

  An optional attribute containing the string **rtl** or **ltr**, indicating the default direction for text in the hidden text XML. Refer to the XHTML 1.0 specification for further details.

- **id** (optional)

  An optional attribute of type xs:ID. Contains an id that is unique in the scope of this document. This attribute can be used for referencing a certain element (e.g., in a style sheet). See the XML schema specification for further details.

- **class** (optional)

  This attribute can contain a space separated list of classes. Useful for convenient style sheet usage.

- **iref** (optional)
- URI which points to an image file corresponding to the region.

  (ex.1 iref=”http://jpeg.org/image.jp2”, ex.2 iref=”jpip://jpeg.org/image.jp2?fsize=32,32&rsiz=32,32”)

G.3.2 Position attributes

Posattrs, a set of position attributes that are common to most visual elements, is defined as follows:

```
<xs:attributeGroup name="posattrs">
  <xs:annotation>
    <xs:documentation>
positioning attributes common to most elements
shape        shape of an element
coords       coordinates of an element
angle        angle of text direction
              0 is horizontal to the right, positive values
              mean counter-clockwise rotation
baseline    angle of the characters in a line of text
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="shape"       type="Shape" default="rect"   />
  <xs:attribute name="coords"      type="Coords"                 />
  <xs:attribute name="angle"       type="Angle" default="0"      />
  <xs:attribute name="baseline"    type="Angle" default="0"      />
</xs:attributeGroup>
```

The following attributes are members of the **Posattrs** group:

- **shape** (optional)

  An optional attribute of type **Shape** containing the shape of the region bounding the element. Possible values are 'rect' for a rectangle and 'poly' for a polygon. The default value for this attribute is **rect.** If this attribute is missing then the bounding shape for this element is the bounding shape of the parent element (which is the whole page in the case of hiddentext).

- **coords** (optional)

  The logical coordinates of the shape bounding the hidden text for this page. The unit is pixels. A resolution can be defined as an attribute on the htx element. If this attribute is missing then the bounding shape for this element is the bounding shape of the parent element (which is the whole page in the case of hiddentext). How the value of cords is to be interpreted depends on the shape attribute. The coord values unit is pixel; there is no percentage or any length unit like inch or centimetre.

  The origin (coordinates '0, 0') is the upper left corner of the page.

- **angle** (optional)

  An attribute of type **Angle** that indicates the angle of orientation of the element, relative to the direction of the element's parent.

Can either be in degree (value followed by a ° sign) or radian measure (value without unit). A value of 0 means the same direction as the element's parent, positive values mean rotating counter-clockwise relative to that direction. Default value is '0'.

- **baseline** (optional)

  An attribute of type **Angle** that indicates the relative orientation of the sub elements and direct content contained in the element with respect to the direction given by the angle attribute.

  Can either be in degree (value followed by a ° sign) or radian measure (value without unit). A value of 0 means the same direction as the element, positive values mean rotating counter-clockwise relative to that direction. Default value is '0'.

The values of shape and cords attribute should be interpreted as described in HTML 4.01 clause 13.6.1 section "AREA attribute definitions".

G.4 Elements

G.4.1 HTX

The **htx** element, the global container and root elements for hidden text and annotations, is declared as follows:

```
<xs:element name="htx">
  <xs:annotation>
    <xs:documentation>
Global   container   for   hidden   text   and   annotations.   Contains   language
attributes, an optional xhtml head and a mandatory body.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="xhtml:head"    minOccurs="0"/>
      <xs:element ref="annotations"   minOccurs="0"/>
      <xs:element ref="hiddentext"    minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="res"       type="xs:Resolution" />
    <xs:attribute name="width"           type="xs:integer"  />
    <xs:attribute name="height"    type="xs:integer"  />
    <xs:attributeGroup ref="coreattrs"/>
  </xs:complexType>
</xs:element>
```

This root element contains an optional **xhtml:head** element, an annotations element and a **hiddentext** element.

Attributes:

Core attributes apply.

- **res** (optional)

  An optional attribute of type Resolution indicating the resolution for any coordinates in dots per inch (dpi). A single number stands for horizontal and vertical resolution having the same values. Two numbers can be used to define different resolutions for horizontal (first number) and vertical (second number).

- **width** (optional)

  The width of the page in pixels.

- **height** (optional)

  The height of the page in pixels.

Elements

- **xhtml:head** (at most one)

  An optional element containing general header data for the hidden text XML elements, including any required cascading style sheet data. Refer to the XHTML 1.0 specification for further details.

- **annotations** (at most one)

  An optional annotations element may be used to attach notes and to describe clickable and highlighted regions on a page.

- **hiddentext** (at most one)

  An optional hiddentext containing the hidden text XML data for a page.

Direct content

- **none**

G.4.2 Parameter

The **param** element is declared as follows:

```
<xs:element name="param">
  <xs:annotation>
    <xs:documentation>
User defined properties for a hidden text and annotations object.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:attribute name="name"  type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

A **param** element of an HTX contains user-defined properties for the associated element, e.g., to specify the OCR engine used to generate the hidden text.

Attributes:

- **name** (mandatory)

  A mandatory string attribute containing the name of this parameter.

Elements

- **none**

Direct content

- **value**

An optional string attribute containing the value for this parameter.

G.4.3 Hidden Text

The **hiddentext** element is used to represent the hidden text for a page and is declared as follows:

```
<xs:element name="hiddentext">
  <xs:annotation>
    <xs:documentation>
A hiddentext element contains hidden text for a page.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="region"                 maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

A **hiddentext** element of an HTX contains a sequence of **region** elements.

Attributes:

Core and position attributes apply.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for hidden text.
- **region** (one or more)

  Sequence of elements containing hidden text for regions in the page.

Direct content

- **none**

G.4.4 Region

The **region** element is declared as follows:

```
<xs:element name="region">
  <xs:annotation>
    <xs:documentation>
A region element contains hidden text for a page region.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"    minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="paragraph"              minOccurs="0"
                                 maxOccurs="unbounded" />
      <xs:element ref="word"     minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="char"     minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="snippet"  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

A **region** element of an HTX contains an optional sequence of **paragraph, line, word, char** and **snippet** elements and optional hidden text for a region.

Attributes:

Core and position attributes apply.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the region.
- **paragraph** (zero or more)

  Optional sequence of elements containing hidden text for paragraphs in the region.
- **line** (zero or more)

  Optional sequence of elements containing hidden text for lines in the region.
- **word** (zero or more)

  Optional sequence of elements containing hidden text for words in the region.
- **char** (zero or more)

  Optional sequence of elements containing hidden text for characters in the region.
- **snippet** (zero or more)

  Optional sequence of elements each identifying an unrecognised portion of the raster.

Direct content

- Region level hidden text

G.4.5 Paragraph

The **paragraph** element is declared as follows:

```
<xs:element name="paragraph">
  <xs:annotation>
    <xs:documentation>
A paragraph element contains hidden text for a paragraph.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="line"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="word"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="char"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="snippet"minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

A **paragraph** element of an HTX contains an optional sequence of **line, word, char** and **snippet** elements and optional hidden text for the paragraph.

Attributes:

Core and position attributes apply.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the paragraph.
- **line** (zero or more)

  Optional sequence of elements containing hidden text for lines in the paragraph.
- **word** (zero or more)

  Optional sequence of elements containing hidden text for words in the paragraph.
- **char** (zero or more)

  Optional sequence of elements containing hidden text for characters in the paragraph.
- **snippet** (zero or more)

  Optional sequence of elements each identifying an unrecognised portion of the raster.

Direct content

- Paragraph level hidden text

G.4.6 Line

The **line** element is declared as follows:

```
<xs:element name="line">
  <xs:annotation>
    <xs:documentation>
A line element contains hidden text for a line.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="word"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="char"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="snippet"minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

A **line** element of an HTX contains an optional sequence of **word, char** and **snippet** elements and optional hidden text for a line.

Attributes:

Core and position attributes apply.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the line.
- **word** (zero or more)

  Optional sequence of elements containing hidden text for words in the line.
- **char** (zero or more)

  Optional sequence of elements containing hidden text for characters in the line.
- **snippet** (zero or more)

  Optional sequence of elements each identifying an unrecognised portion of the raster.

Direct content

- Line level hidden text

G.4.7 Word

The **word** element is declared as follows:

```
<xs:element name="word">
  <xs:annotation>
    <xs:documentation>
A word element contains hidden text for a word.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"    minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="char"     minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="snippet" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="altword" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="conf"type="Percentage" />
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs"  />
  </xs:complexType>
</xs:element>
```

A **word** element of an HTX contains an optional sequence of **char, altword** and **snippet** elements and optional hidden text for a word.

Attributes:

Core and position attributes apply.

- **conf** (optional)

  An optional attribute indicating the confidence of the accuracy of the given word as a percentage in the range of 0% to 100%.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the word.
- **char** (zero or more)

  Optional sequence of elements containing hidden text for characters in the word.
- **snippet** (zero or more)

  Optional sequence of elements each identifying an unrecognised portion of the raster.
- **altword** (zero or more)

  Optional sequence of elements containing hidden text for alternate words. It defines a possible alternative interpretation of the word in the raster image when uncertainties exist.

Direct content

- Word level hidden text

G.4.8 Alternative word

The **altword** element contains alternative hidden text for a word and is declared as follows:

```
<xs:element name="altword">
  <xs:annotation>
    <xs:documentation>
A altword element contains hidden text for a alternative word.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="char"   minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="snippet" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="conf"type="Percentage" />
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

An **altword** element of an HTX contains an optional sequence of **char** and **snippet** elements, and optional hidden text for an alternative word.

Attributes:

Core and position attributes apply.

- **conf** (optional)

  An optional attribute indicating the confidence of the accuracy of the given alternate word as a percentage in the range of 0% to 100%.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the alternate word.

- **char** (zero or more)

  Optional sequence of elements containing hidden text for characters in the alternate word.

- **snippet** (zero or more)

  Optional sequence of elements each identifying an unrecognised portion of the raster.

Direct content

- Word level hidden text specifying some alternative text.

G.4.9 Character

The **char** element contains a hidden text character and is declared as follows:

```
<xs:element name="char">
  <xs:annotation>
    <xs:documentation>
A char element contains a hidden text character.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="altchar" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="conf"type="Percentage" />
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

A **char** element of an HTX contains an optional sequence of **altchar** elements and an optional hidden text character.

Attributes:

Core and position attributes apply.

- **conf** (optional)

  An optional attribute indicating the confidence of the accuracy of the given character as a percentage in the range of 0% to 100%.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the character.

- **altchar** (zero or more)

  Optional sequence of elements containing hidden text for alternate characters in the alternate word. This element may only occur directly after a char or altchar element. It defines a possible alternative interpretation of the character in the raster image when uncertainties exist.

Direct content

- Character level hidden text

G.4.10 Alternative character

The **altchar** element is declared as follows:

```
<xs:element name="altchar">
  <xs:annotation>
    <xs:documentation>
A altchar element contains a hidden text alternative character.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="param"  minOccurs="0" maxOccurs="unbounded"     />
    </xs:sequence>
    <xs:attribute name="conf"type="Percentage" />
    <xs:attributeGroup ref="coreattrs" />
    <xs:attributeGroup ref="posattrs" />
  </xs:complexType>
</xs:element>
```

An **altchar** element of an HTX contains an optional sequence of **snippet** elements and optional hidden text for an alternative character.

Attributes:

Core and position attributes apply.

- **conf** (optional)

  An optional attribute indicating the confidence of the accuracy of the given alternate character as a percentage in the range of 0% to 100%.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the alternate character.

Direct content

- Character level hidden text specifying an alternative character.

G.4.11 Snippet

The **snippet** element is declared as follows:

```
<xs:element name="snippet">
  <xs:annotation>
    <xs:documentation>
A snippet element identifies an unrecognised portion of the image.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:element ref="param" />
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs"   />
    <xs:attributeGroup ref="posattrs"    />
  </xs:complexType>
</xs:element>
```

A **snippet** element of an HTX identifies an unrecognised portion of the image.

Attributes:

Core and position attributes apply.

Elements

- **param** (zero or more)

  Optional sequence of parameter elements may be used to specify user-defined properties for the snippet.

Direct content

- **none**

G.4.12 Annotations

The **annotations** element is declared as follows:

```
<xs:element name="annotations">
  <xs:annotation>
    <xs:documentation>
The annotations element represents annotated, clickable and highlighted areas
on a page.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded">
      <xs:element ref="area"/>
    </xs:sequence>
    <xs:attributeGroup ref="coreattrs" />
  </xs:complexType>
</xs:element>
```

An annotation element of an HTX contains a sequence of area elements defining clickable and/or highlighted areas as well as notes for a page.

Attributes:

Core attributes apply.

Elements

- **area** (at least one)

  Sequence of elements defining clickable and highlighted areas on a page.

Direct content

- **none**

G.4.13 Area

The **area** element is declared as follows:

```
<xs:element name="area">
  <xs:annotation>
    <xs:documentation>
An area element represents a clickable and/or highlighted area on a page.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="href"  type="xhtml:URI" />
    <xs:attribute name="alt"   type="xhtml:Text" />
    <xs:attribute name="target"type="xhtml:URI" default="_self" />
    <xs:attribute name="shape" type="Shape" default="rect" />
    <xs:attribute name="coords"type="Coords" />
    <xs:attributeGroup ref="coreattrs" />
  </xs:complexType>
</xs:element>
```

An **area** element of an HTX defines a clickable and/or highlighted area on a page. To define a highlighted, non-clickable area the **href** attribute is to be omitted.

Attributes:

Core and position attributes apply.

- **href** (optional)

  URI of the hyperlink to use when the area is clicked.

- **alt** (optional)

  An optional text descriptor for the hyperlink, possibly for use as a tooltip. Must not be used without the **href** attribute defined.

- **target** (optional)

  The frame that the document the hyperlink points to should be displayed in (if the viewing program supports any kind of framing). Default is "_self". See HTML 4.01 definition for further details. Must not be used without the **href** attribute defined.

Elements

- **none**

Direct content

- Notes for this area.

Annex H

Hidden text and annotations schema

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

H.1 XML schema

The following URL references the XML schema for HTX:

http://www.jpeg.org/hiddentext/htx.xsd

This XML schema shall be referred to as the ***Hidden Text XML Schema***.

H.1.1 Version and encoding

The HTX schema uses XML 1.0 and UTF-8 encoding:

```
<?xml version="1.0" encoding="utf-8" ?>
```

H.1.2 Schema

The schema wrapper for HTX is declared as follows:

```
<xs:schema
     id = "htx"
     xmlns = "http://www.jpeg.org/hiddentext/htx"
     xmlns:xs = "http://www.w3.org/2001/XMLSchema"
     xmlns:xhtml = "http://www.w3.org/1999/xhtml"
     targetNamespace = "http://www.jpeg.org/hiddentext/htx"
     elementFormDefault = "qualified">
```

The schema uses several types and elements from the XHTML 1.0 XML schema, and imports this schema as follows:

```
<xs:import
     namespace = "http://www.w3.org/1999/xhtml"
     schemaLocation = "http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd">
</xs:import>
```

The following tag is used to close the HTX schema:

```
</xs:schema>
```

Annex I

Hidden text and annotations examples

(This annex does not form an integral part of this Recommendation | International Standard.)

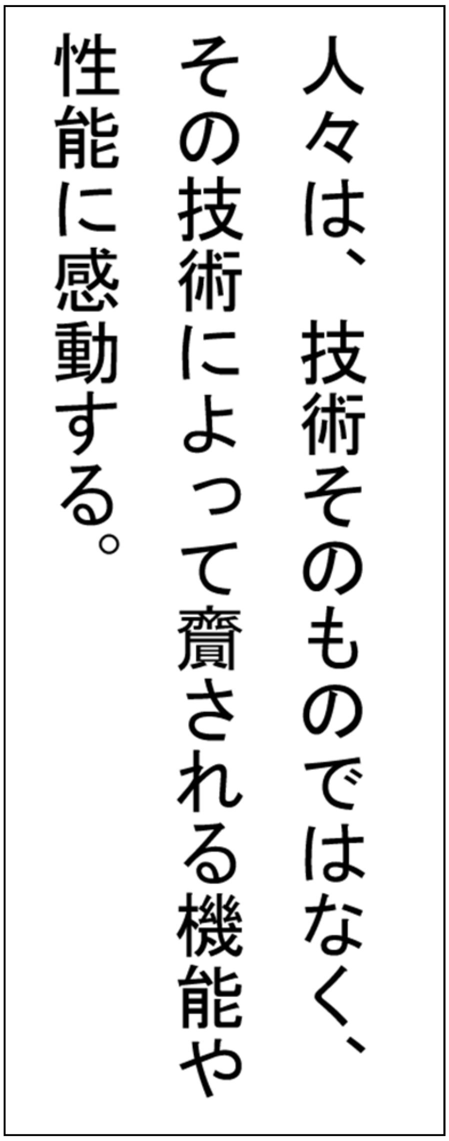
I.1 Example 1

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="#style" type="text/css"?>
<htx  xmlns                 = "http://www.jpeg.org/hiddentext/htx"
      xmlns:xhtml           = "http://www.w3.org/1999/xhtml"
      xmlns:xsi             = "http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation    = "http://www.jpeg.org/hiddentext/htx htx.xsd"
      lang                  = "en"
      dir                   = "ltr">

      <!-- ************************************************** -->
      <!-- Hidden text and annotations optional xhtml header  -->
      <!-- ************************************************** -->

    <xhtml:head lang="de" dir="ltr">
            <xhtml:title>Test Document Title</xhtml:title>
            <xhtml:style   type="text/css" media="Screen"
                           title="Style Sheet Name One" lang="de" dir="ltr" id="style">
          style                                         { display: none; }
          body, paragraph, line                         { display: block; }
          word, altword, char, altchar, snippet  { display: inline; }
          body                                          { font-family: Arial,sans-serif; }
          altword, altchar                              { font-style: italic; color: grey; }
          [class~="red"]                                { background-color: red; }
          [class~="blue"]                               { color: blue; }
          [class~="large"]                              { font-size: 150%; }
            </xhtml:style>
    </xhtml:head>

    <!-- ********************************************** -->
    <!-- Optional annotations for page                    -->
    <!-- ********************************************** -->

    <annotations>
            <area   shape="rect" coords="0, 10, 50, 200" alt="link somewhere"
                    href ="http://some.where.com/there" class="red" />
            <area   shape="poly" coords="23, 45, 0, 100, 90, 100"
           alt="link to somewhere alse" href ="http://somewhere.else.org/" />
    </annotations>

      <!-- ********************************************** -->
      <!-- Object for hidden text in the page             -->
      <!-- ********************************************** -->

      <hiddentext shape="rect" coords="0, 0, 100, 200">

            <!-- Optional, user defined page level parameters -->
            <param name="param1">abcd</param>
            <!-- First, mandatory region -->

            <region shape="rect" coords="0, 0, 100, 100">
                    Here is some region level hidden text.

                    <!-- Optional region parameters -->
                    <param name="param2">efgh</param>
                    <paragraph shape="poly"
```

```
                      coords="0, 0, 100, 0, 100, 50, 30, 50, 30, 60, 0, 60">
                              Here is some paragraph level hidden text.

                              <!-- Optional paragraph parameters -->
                              <param name="param3">ijkl</param>
                              <param name="param4">mnop</param>

                              <line shape="rect" coords="0, 0, 100, 20" class="large" >
                                      <word conf="70%">
                                              Word
                                              <char    conf="90%">
                                                  c
                                                  <altchar conf="10%">o</altchar>
                                              </char>
                                              <char   conf="80%">
                                                  h
                                              <altchar conf="12%">n</altchar>
                                              <altchar conf= "8%">k</altchar>
                                              </char>
                                      <altword conf="15%">
                                              Vordok
                                      </altword>
                                      <altword conf= "5%">
                                              Wordoh
                                      </altword>
                                      </word>
                                      <word>
                                              Next
                                      </word>

                                      Some line level hidden text.
                              </line>

                      </paragraph>

                      <line class="large blue">
                              And some line level hidden text.
                      </line>

              </region>

              <!-- Second, optional region -->
              <region>
                  <!-- ... -->
              </region>

      </hiddentext>
</htx>
```

I.2 Example 2

This example shows three lines in Japanese language formatted within HTX.

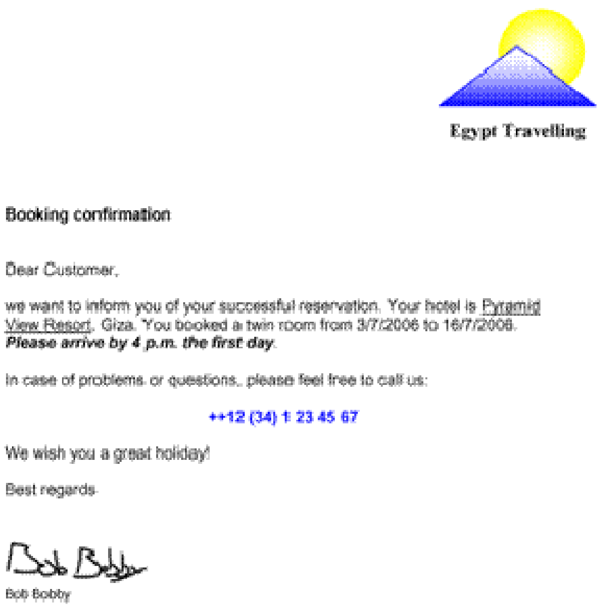


人々は、技術そのものではなく、
その技術によって齎される機能や
性能に感動する。

```
<paragraph angle="-90" baseline="90">
  <line shape="rect" coords="236, 20, 296, 854">
    <char conf="95%">人々は</char>

    <char conf="70%">、

      <altchar conf="40%">,</altchar>
    </char>

    <char conf="80%">技術</char>

    <char conf="90%">そのものではなく</char>

    <char conf="70">、

      <altchar conf="40%">,</altchar>
    </char>
  </line>

  <line shape="rect" coords="136, 20, 196, 888">
    <char conf="90%">その技術によって</char>

    <char conf="70%">齎

        <altchar conf="50%">斎</altchar>
    </char>

    <char conf="90%">される</char>

    <char conf="80%">機能</char>

    <char conf="90%">や</char>
  </line>

  </line shape="rect" coords="36, 20, 96, 452">
    <word conf="80%">

      <char>性能</char>
    </word>

    <char conf="90%">に感動する</char>

    <char conf="70%">。

      <altchar conf="50%">.</altchar>
    </char>
  </line>
</paragraph>
```

I.3 Example 3

This example shows an HTX that has been automatically generated from an OCR process' XML output by use of an extensible stylesheet language transformation (XSLT). It contains text formatting information and character confidences as provided by the OCR process. The included reference to HTX2HTML.xsl enables an automatic conversion of this HTX to HTML within a web browser that supports XSLT.



Egypt Travelling

Booking confirmation

Dear Customer,

we want to inform you of your successful reservation. Your hotel is Pyramid View Resort, Giza. You booked a twin room from 3/7/2006 to 16/7/2006. ***Please arrive by 4 p.m. the first day***.

In case of problems or questions, please feel free to call us:

++12 (34) 1 23 45 67

We wish you a great holiday!

Best regards

[illegible]

Bob Bobby

Figure I.1 – Example of a scanned document page

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="HTX2HTML.xsl" type="text/xsl"?>
<htx xmlns="http://www.jpeg.org/hiddentext/htx" res="300"
     xmlns:xhtml=http://www.w3.org/1999/xhtml
     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
     xsi:schemaLocation="http://www.jpeg.org/hiddentext/htx htx.xsd">
    <xhtml:head lang="de" dir="ltr">
        <xhtml:title/>
        <xhtml:style type="text/css" media="Screen" title="" lang="de" dir="ltr" id="style">
                style, param { display: none; }
                body, paragraph, line { display: block; }
                word, altword, char, altchar, snippet { display: inline; }
                *.format-bold { font-weight:bold; }
                *.format-italic { font-style:italic; }
                *.format-subscript { vertical-align:sub; }
                *.format-superscript { vertical-align:super; }
                *.format-smallcaps { font-variant:small-caps; }
                *.format-underline { text-decoration:underline; }
                *.format-strikeout { text-decoration:line-through; }
                *.format-ffTimes-New-Roman { font-family:Times New Roman; }
                *.format-fs14 { font-size:14pt; }
                *.format-ffArial { font-family:Arial; }
                *.format-fs12 { font-size:12pt; }
                *.format-color14942208 { color:rgb(0, 0, 228); }
                *.format-fs32 { font-size:32pt; }
                *.format-fs10 { font-size:10pt; }
                </xhtml:style>
    </xhtml:head>
    <hiddentext>
        <region shape="rect" coords="1744, 566, 2196, 634">
            <paragraph>
                <line shape="rect" coords="1759, 578, 2179, 630">
                    <word class="format-ffTimes-New-Roman format-fs14 ">
                        <char conf="100%" shape="rect" coords="1759, 579, 1793, 618">E</char>
                        <char conf="100%" shape="rect" coords="1798, 591, 1824, 630">g</char>
                        <char conf="100%" shape="rect" coords="1825, 592, 1854, 630">y</char>
                        <char conf="100%" shape="rect" coords="1855, 591, 1884, 630">p</char>
                        <char conf="100%" shape="rect" coords="1888, 582, 1905, 618">t</char>
                        <char conf="100%" shape="rect" coords="1905, 579, 1923, 618"> </char>
                    </word>
                    <word class="format-ffTimes-New-Roman format-fs14 ">
                        <char conf="100%" shape="rect" coords="1923, 579, 1958, 618">T</char>
                        <char conf="77%"  shape="rect" coords="1962, 591, 1985, 618">r</char>
                        <char conf="100%" shape="rect" coords="1987, 591, 2013, 619">a</char>
                        <char conf="100%" shape="rect" coords="2016, 592, 2044, 619">v</char>
                        <char conf="100%" shape="rect" coords="2046, 591, 2068, 619">e</char>
                        <char conf="49%"  shape="rect" coords="2071, 579, 2085, 618">l</char>
                        <char conf="49%"  shape="rect" coords="2087, 579, 2101, 618">l</char>
                        <char conf="100%" shape="rect" coords="2103, 578, 2117, 618">i</char>
                        <char conf="100%" shape="rect" coords="2121, 591, 2150, 618">n</char>
                        <char conf="100%" shape="rect" coords="2153, 591, 2179, 630">g</char>
                    </word>
                </line>
            </paragraph>
        </region>
        <region shape="rect" coords="360, 832, 2052, 1758">
            <paragraph>
                <line shape="rect" coords="379, 843, 888, 891">
                    <word class="format-ffArial format-fs12 format-bold ">
                        <char conf="100%" shape="rect" coords="379, 844, 409, 880">B</char>
                        <char conf="100%" shape="rect" coords="413, 853, 440, 881">o</char>
                        <char conf="100%" shape="rect" coords="444, 853, 471, 881">o</char>
                        <char conf="100%" shape="rect" coords="475, 844, 499, 880">k</char>
                        <char conf="100%" shape="rect" coords="503, 844, 510, 880">i</char>
                        <char conf="100%" shape="rect" coords="517, 853, 541, 880">n</char>
                        <char conf="100%" shape="rect" coords="546, 853, 571, 891">g</char>
                        <char conf="100%" shape="rect" coords="571, 853, 591, 881"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold ">
                        <char conf="100%" shape="rect" coords="591, 853, 615, 881">c</char>
                        <char conf="100%" shape="rect" coords="619, 853, 646, 881">o</char>
                        <char conf="100%" shape="rect" coords="650, 853, 674, 880">n</char>
                        <char conf="100%" shape="rect" coords="678, 843, 696, 880">f</char>
                        <char conf="100%" shape="rect" coords="698, 844, 705, 880">i</char>
                        <char conf="100%" shape="rect" coords="711, 853, 728, 880">r</char>
                        <char conf="100%" shape="rect" coords="731, 853, 770, 880">m</char>
                        <char conf="100%" shape="rect" coords="774, 853, 798, 881">a</char>
                        <char conf="100%" shape="rect" coords="801, 845, 816, 881">t</char>
                        <char conf="100%" shape="rect" coords="820, 844, 827, 880">i</char>
                        <char conf="100%" shape="rect" coords="833, 853, 860, 881">o</char>
                        <char conf="100%" shape="rect" coords="864, 853, 888, 880">n</char>
                    </word>
                </line>
            </paragraph>
            <paragraph>
```

```
                <line shape="rect" coords="379, 1015, 723, 1059">
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="379, 1016, 408, 1052">D</char>
                        <char conf="84%" shape="rect" coords="413, 1025, 437, 1053">e</char>
                        <char conf="100%" shape="rect" coords="441, 1025, 464, 1053">a</char>
                        <char conf="100%" shape="rect" coords="470, 1025, 484, 1052">r</char>
                        <char conf="100%" shape="rect" coords="484, 1015, 500, 1052"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="85%" shape="rect" coords="500, 1015, 531, 1053">C</char>
                        <char conf="100%" shape="rect" coords="537, 1026, 557, 1053">u</char>
                        <char conf="100%" shape="rect" coords="562, 1025, 584, 1053">s</char>
                        <char conf="100%" shape="rect" coords="587, 1017, 599, 1053">t</char>
                        <char conf="100%" shape="rect" coords="602, 1025, 626, 1053">o</char>
                        <char conf="100%" shape="rect" coords="631, 1025, 665, 1052">m</char>
                        <char conf="84%" shape="rect" coords="672, 1025, 696, 1053">e</char>
                        <char conf="100%" shape="rect" coords="701, 1025, 715, 1052">r</char>
                        <char conf="100%" shape="rect" coords="719, 1047, 723, 1059">,</char>
                    </word>
                </line>
            </paragraph>
            <paragraph>
                <line shape="rect" coords="375, 1130, 2036, 1178">
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="375, 1141, 411, 1167">w</char>
                        <char conf="84%" shape="rect" coords="413, 1140, 437, 1168">e</char>
                        <char conf="84%" shape="rect" coords="437, 1140, 453, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="453, 1141, 489, 1167">w</char>
                        <char conf="100%" shape="rect" coords="491, 1140, 514, 1168">a</char>
                        <char conf="100%" shape="rect" coords="521, 1140, 541, 1167">n</char>
                        <char conf="100%" shape="rect" coords="546, 1132, 558, 1168">t</char>
                        <char conf="100%" shape="rect" coords="558, 1132, 573, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="573, 1132, 585, 1168">t</char>
                        <char conf="100%" shape="rect" coords="588, 1140, 612, 1168">o</char>
                        <char conf="100%" shape="rect" coords="612, 1131, 632, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="632, 1131, 636, 1167">i</char>
                        <char conf="100%" shape="rect" coords="643, 1140, 663, 1167">n</char>
                        <char conf="100%" shape="rect" coords="668, 1130, 683, 1167">f</char>
                        <char conf="100%" shape="rect" coords="683, 1140, 707, 1168">o</char>
                        <char conf="100%" shape="rect" coords="712, 1140, 726, 1167">r</char>
                        <char conf="100%" shape="rect" coords="728, 1140, 762, 1167">m</char>
                        <char conf="100%" shape="rect" coords="762, 1140, 782, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="782, 1141, 806, 1178">y</char>
                        <char conf="100%" shape="rect" coords="808, 1140, 832, 1168">o</char>
                        <char conf="100%" shape="rect" coords="838, 1141, 858, 1168">u</char>
                        <char conf="100%" shape="rect" coords="858, 1140, 877, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="877, 1140, 901, 1168">o</char>
                        <char conf="100%" shape="rect" coords="904, 1130, 919, 1167">f</char>
                        <char conf="100%" shape="rect" coords="919, 1130, 932, 1168"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="932, 1141, 956, 1178">y</char>
                        <char conf="100%" shape="rect" coords="958, 1140, 982, 1168">o</char>
                    <char conf="100%" shape="rect" coords="988, 1141, 1008, 1168">u</char>
                    <char conf="100%" shape="rect" coords="1015, 1140, 1029, 1167">r</char>
                    <char conf="100%" shape="rect" coords="1029, 1140, 1043, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="1043, 1140, 1065, 1168">s</char>
                    <char conf="100%" shape="rect" coords="1071, 1141, 1091, 1168">u</char>
                    <char conf="100%" shape="rect" coords="1097, 1140, 1119, 1168">c</char>
                    <char conf="100%" shape="rect" coords="1122, 1140, 1144, 1168">c</char>
                    <char conf="82%" shape="rect" coords="1147, 1140, 1171, 1168">e</char>
                    <char conf="100%" shape="rect" coords="1173, 1140, 1195, 1168">s</char>
                    <char conf="100%" shape="rect" coords="1198, 1140, 1220, 1168">s</char>
                    <char conf="100%" shape="rect" coords="1223, 1130, 1238, 1167">f</char>
                    <char conf="100%" shape="rect" coords="1240, 1141, 1260, 1168">u</char>
                    <char conf="100%" shape="rect" coords="1268, 1131, 1272, 1167">l</char>
                    <char conf="100%" shape="rect" coords="1272, 1131, 1292, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="1292, 1140, 1306, 1167">r</char>
                    <char conf="84%" shape="rect" coords="1308, 1140, 1332, 1168">e</char>
                    <char conf="100%" shape="rect" coords="1335, 1140, 1357, 1168">s</char>
                    <char conf="82%" shape="rect" coords="1361, 1140, 1385, 1168">e</char>
                    <char conf="100%" shape="rect" coords="1390, 1140, 1405, 1167">r</char>
                    <char conf="96%" shape="rect" coords="1405, 1141, 1428, 1167">v</char>
                    <char conf="100%" shape="rect" coords="1430, 1140, 1453, 1168">a</char>
                    <char conf="100%" shape="rect" coords="1457, 1132, 1469, 1168">t</char>
                    <char conf="100%" shape="rect" coords="1474, 1131, 1478, 1167">i</char>
                    <char conf="100%" shape="rect" coords="1483, 1140, 1507, 1168">o</char>
                    <char conf="100%" shape="rect" coords="1513, 1140, 1533, 1167">n</char>
                    <char conf="100%" shape="rect" coords="1542, 1162, 1546, 1167">.</char>
                    <char conf="100%" shape="rect" coords="1546, 1131, 1564, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="1564, 1131, 1596, 1167">Y</char>
                    <char conf="100%" shape="rect" coords="1600, 1140, 1624, 1168">o</char>
                    <char conf="100%" shape="rect" coords="1630, 1141, 1650, 1168">u</char>
                    <char conf="100%" shape="rect" coords="1656, 1140, 1670, 1167">r</char>
                    <char conf="100%" shape="rect" coords="1670, 1131, 1688, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="1688, 1131, 1708, 1167">h</char>
                    <char conf="100%" shape="rect" coords="1714, 1140, 1738, 1168">o</char>
                    <char conf="100%" shape="rect" coords="1741, 1132, 1753, 1168">t</char>
                    <char conf="82%" shape="rect" coords="1756, 1140, 1780, 1168">e</char>
                    <char conf="100%" shape="rect" coords="1785, 1131, 1789, 1167">l</char>
                    <char conf="100%" shape="rect" coords="1789, 1131, 1810, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="1810, 1131, 1814, 1167">i</char>
                    <char conf="100%" shape="rect" coords="1819, 1140, 1841, 1168">s</char>
                    <char conf="100%" shape="rect" coords="1841, 1131, 1860, 1168"> </char>
                </word>
                <word class="format-ffArial format-fs12 format-underline ">
                    <char conf="100%" shape="rect" coords="1860, 1131, 1887, 1167">P</char>
                    <char conf="100%" shape="rect" coords="1890, 1141, 1914, 1171">y</char>
                    <char conf="38%" shape="rect" coords="1917, 1140, 1931, 1167">r</char>
                    <char conf="56%" shape="rect" coords="1933, 1140, 1956, 1168">a</char>
                    <char conf="100%" shape="rect" coords="1962, 1140, 1996, 1167">m</char>
                    <char conf="100%" shape="rect" coords="2004, 1131, 2008, 1167">i</char>
                    <char conf="100%" shape="rect" coords="2014, 1131, 2036, 1168">d</char>
                </word>
            </line>
            <line shape="rect" coords="375, 1187, 1966, 1227">
```

```
                    <word class="format-ffArial format-fs12 format-underline ">
                        <char conf="100%" shape="rect" coords="375, 1188, 408, 1224">V</char>
                        <char conf="100%" shape="rect" coords="412, 1188, 416, 1224">i</char>
                        <char conf="91%" shape="rect" coords="421, 1197, 445, 1225">e</char>
                        <char conf="100%" shape="rect" coords="447, 1198, 483, 1224">w</char>
                        <char conf="100%" shape="rect" coords="483, 1188, 501, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-underline ">
                        <char conf="100%" shape="rect" coords="501, 1188, 532, 1224">R</char>
                        <char conf="91%" shape="rect" coords="535, 1197, 559, 1225">e</char>
                        <char conf="100%" shape="rect" coords="562, 1197, 584, 1225">s</char>
                        <char conf="100%" shape="rect" coords="588, 1197, 612, 1225">o</char>
                        <char conf="37%" shape="rect" coords="617, 1197, 631, 1224">r</char>
                        <char conf="70%" shape="rect" coords="632, 1189, 644, 1225">t</char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="676, 1187, 708, 1225">G</char>
                        <char conf="100%" shape="rect" coords="715, 1188, 719, 1224">i</char>
                        <char conf="100%" shape="rect" coords="724, 1198, 747, 1224">z</char>
                        <char conf="100%" shape="rect" coords="750, 1197, 773, 1225">a</char>
                        <char conf="100%" shape="rect" coords="780, 1219, 784, 1224">.</char>
                        <char conf="100%" shape="rect" coords="784, 1188, 803, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="803, 1188, 835, 1224">Y</char>
                        <char conf="100%" shape="rect" coords="839, 1197, 863, 1225">o</char>
                        <char conf="100%" shape="rect" coords="868, 1198, 888, 1225">u</char>
                        <char conf="100%" shape="rect" coords="888, 1188, 910, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="910, 1188, 932, 1225">b</char>
                        <char conf="100%" shape="rect" coords="936, 1197, 960, 1225">o</char>
                        <char conf="100%" shape="rect" coords="964, 1197, 988, 1225">o</char>
                        <char conf="100%" shape="rect" coords="994, 1188, 1015, 1224">k</char>
                        <char conf="84%" shape="rect" coords="1017, 1197, 1041, 1225">e</char>
                        <char conf="100%" shape="rect" coords="1044, 1188, 1066, 1225">d</char>
                        <char conf="100%" shape="rect" coords="1066, 1188, 1086, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1086, 1197, 1109, 1225">a</char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1109, 1189, 1127, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1127, 1189, 1139, 1225">t</char>
                        <char conf="100%" shape="rect" coords="1140, 1198, 1176, 1224">w</char>
                        <char conf="100%" shape="rect" coords="1180, 1188, 1184, 1224">i</char>
                        <char conf="100%" shape="rect" coords="1191, 1197, 1211, 1224">n</char>
                        <char conf="100%" shape="rect" coords="1211, 1197, 1232, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1232, 1197, 1246, 1224">r</char>
                        <char conf="100%" shape="rect" coords="1247, 1197, 1271, 1225">o</char>
                        <char conf="100%" shape="rect" coords="1275, 1197, 1299, 1225">o</char>
                        <char conf="100%" shape="rect" coords="1304, 1197, 1338, 1224">m</char>
                        <char conf="100%" shape="rect" coords="1338, 1187, 1357, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1357, 1187, 1372, 1224">f</char>
                        <char conf="100%" shape="rect" coords="1373, 1197, 1387, 1224">r</char>
                        <char conf="100%" shape="rect" coords="1389, 1197, 1413, 1225">o</char>
                        <char conf="100%" shape="rect" coords="1417, 1197, 1451, 1224">m</char>
                        <char conf="100%" shape="rect" coords="1451, 1188, 1472, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1472, 1188, 1495, 1225">3</char>
                        <char conf="100%" shape="rect" coords="1498, 1188, 1512, 1225">/</char>
                        <char conf="100%" shape="rect" coords="1514, 1189, 1537, 1224">7</char>
                        <char conf="100%" shape="rect" coords="1540, 1188, 1554, 1225">/</char>
                        <char conf="100%" shape="rect" coords="1555, 1188, 1578, 1224">2</char>
                        <char conf="100%" shape="rect" coords="1584, 1188, 1606, 1225">0</char>
                        <char conf="100%" shape="rect" coords="1612, 1188, 1634, 1225">0</char>
                        <char conf="100%" shape="rect" coords="1639, 1188, 1662, 1225">6</char>
                        <char conf="100%" shape="rect" coords="1662, 1188, 1680, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1680, 1189, 1692, 1225">t</char>
                        <char conf="100%" shape="rect" coords="1695, 1197, 1719, 1225">o</char>
                        <char conf="100%" shape="rect" coords="1719, 1188, 1740, 1225"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1740, 1188, 1753, 1224">1</char>
                        <char conf="100%" shape="rect" coords="1764, 1188, 1787, 1225">6</char>
                        <char conf="100%" shape="rect" coords="1790, 1188, 1804, 1225">/</char>
                        <char conf="100%" shape="rect" coords="1806, 1189, 1829, 1224">7</char>
                        <char conf="100%" shape="rect" coords="1832, 1188, 1846, 1225">/</char>
                        <char conf="100%" shape="rect" coords="1847, 1188, 1870, 1224">2</char>
                        <char conf="100%" shape="rect" coords="1876, 1188, 1898, 1225">0</char>
                        <char conf="100%" shape="rect" coords="1904, 1188, 1926, 1225">0</char>
                        <char conf="100%" shape="rect" coords="1931, 1188, 1954, 1225">6</char>
                        <char conf="100%" shape="rect" coords="1962, 1219, 1966, 1224">.</char>
                    </word>
                </line>
                <line shape="rect" coords="377, 1245, 1220, 1293">
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="377, 1246, 410, 1282">P</char>
                        <char conf="100%" shape="rect" coords="410, 1246, 425, 1282">l</char>
                        <char conf="100%" shape="rect" coords="425, 1255, 450, 1283">e</char>
                        <char conf="100%" shape="rect" coords="452, 1255, 477, 1283">a</char>
                        <char conf="100%" shape="rect" coords="479, 1255, 506, 1283">s</char>
                        <char conf="100%" shape="rect" coords="508, 1255, 533, 1283">e</char>
                        <char conf="100%" shape="rect" coords="533, 1255, 549, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="549, 1255, 574, 1283">a</char>
                        <char conf="100%" shape="rect" coords="577, 1255, 599, 1282">r</char>
                        <char conf="100%" shape="rect" coords="596, 1255, 618, 1282">r</char>
                        <char conf="100%" shape="rect" coords="616, 1246, 631, 1282">i</char>
                        <char conf="100%" shape="rect" coords="632, 1256, 659, 1282">v</char>
                        <char conf="100%" shape="rect" coords="658, 1255, 683, 1283">e</char>
                        <char conf="100%" shape="rect" coords="683, 1246, 699, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="699, 1246, 727, 1283">b</char>
                        <char conf="100%" shape="rect" coords="728, 1256, 759, 1293">y</char>
                        <char conf="100%" shape="rect" coords="759, 1246, 771, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="771, 1246, 798, 1282">4</char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="798, 1246, 810, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="810, 1255, 841, 1292">p</char>
                        <char conf="100%" shape="rect" coords="844, 1275, 852, 1282">.</char>
                        <char conf="100%" shape="rect" coords="858, 1255, 900, 1282">m</char>
                        <char conf="100%" shape="rect" coords="902, 1275, 910, 1282">.</char>
                        <char conf="100%" shape="rect" coords="910, 1247, 932, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="932, 1247, 947, 1283">t</char>
                        <char conf="100%" shape="rect" coords="947, 1246, 974, 1282">h</char>
                        <char conf="100%" shape="rect" coords="978, 1255, 1003, 1283">e</char>
                        <char conf="100%" shape="rect" coords="1003, 1245, 1020, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="1020, 1245, 1041, 1282">f</char>
                        <char conf="100%" shape="rect" coords="1035, 1246, 1050, 1282">i</char>
                        <char conf="100%" shape="rect" coords="1049, 1255, 1071, 1282">r</char>
                        <char conf="100%" shape="rect" coords="1068, 1255, 1095, 1283">s</char>
                        <char conf="100%" shape="rect" coords="1098, 1247, 1113, 1283">t</char>
                        <char conf="100%" shape="rect" coords="1113, 1246, 1128, 1283"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 format-bold format-italic ">
                        <char conf="100%" shape="rect" coords="1128, 1246, 1159, 1283">d</char>
                        <char conf="100%" shape="rect" coords="1158, 1255, 1183, 1283">a</char>
                        <char conf="100%" shape="rect" coords="1183, 1256, 1214, 1293">y</char>
                        <char conf="100%" shape="rect" coords="1216, 1277, 1220, 1282">.</char>
                    </word>
                </line>
            </paragraph>
            <paragraph>
                <line shape="rect" coords="380, 1360, 1688, 1407">
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="380, 1361, 384, 1397">I</char>
                        <char conf="100%" shape="rect" coords="393, 1370, 413, 1397">n</char>
                        <char conf="100%" shape="rect" coords="413, 1370, 433, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="433, 1370, 455, 1398">c</char>
                        <char conf="100%" shape="rect" coords="458, 1370, 481, 1398">a</char>
                        <char conf="100%" shape="rect" coords="484, 1370, 506, 1398">s</char>
                        <char conf="82%" shape="rect" coords="510, 1370, 534, 1398">e</char>
                        <char conf="82%" shape="rect" coords="534, 1370, 552, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="552, 1370, 576, 1398">o</char>
                        <char conf="100%" shape="rect" coords="579, 1360, 594, 1397">f</char>
                        <char conf="100%" shape="rect" coords="594, 1360, 610, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="610, 1370, 632, 1407">p</char>
                        <char conf="100%" shape="rect" coords="637, 1370, 651, 1397">r</char>
                        <char conf="100%" shape="rect" coords="652, 1370, 676, 1398">o</char>
                        <char conf="100%" shape="rect" coords="682, 1361, 704, 1398">b</char>
                        <char conf="100%" shape="rect" coords="710, 1361, 714, 1397">l</char>
                        <char conf="82%" shape="rect" coords="719, 1370, 743, 1398">e</char>
                        <char conf="100%" shape="rect" coords="748, 1370, 782, 1397">m</char>
                        <char conf="100%" shape="rect" coords="787, 1370, 809, 1398">s</char>
                        <char conf="100%" shape="rect" coords="809, 1370, 827, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="827, 1370, 851, 1398">o</char>
                        <char conf="100%" shape="rect" coords="856, 1370, 870, 1397">r</char>
                        <char conf="100%" shape="rect" coords="870, 1370, 886, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="886, 1370, 908, 1407">q</char>
                        <char conf="100%" shape="rect" coords="916, 1371, 936, 1398">u</char>
                        <char conf="82%" shape="rect" coords="941, 1370, 965, 1398">e</char>
                        <char conf="100%" shape="rect" coords="968, 1370, 990, 1398">s</char>
                        <char conf="100%" shape="rect" coords="993, 1362, 1005, 1398">t</char>
                        <char conf="100%" shape="rect" coords="1010, 1361, 1014, 1397">i</char>
                        <char conf="100%" shape="rect" coords="1019, 1370, 1043, 1398">o</char>
                        <char conf="100%" shape="rect" coords="1049, 1370, 1069, 1397">n</char>
                        <char conf="100%" shape="rect" coords="1074, 1370, 1096, 1398">s</char>
                        <char conf="100%" shape="rect" coords="1103, 1392, 1107, 1404">,</char>
                        <char conf="100%" shape="rect" coords="1107, 1370, 1130, 1404"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1130, 1370, 1152, 1407">p</char>
                        <char conf="100%" shape="rect" coords="1157, 1361, 1161, 1397">l</char>
                        <char conf="82%" shape="rect" coords="1166, 1370, 1190, 1398">e</char>
                        <char conf="100%" shape="rect" coords="1194, 1370, 1217, 1398">a</char>
                        <char conf="100%" shape="rect" coords="1221, 1370, 1243, 1398">s</char>
                        <char conf="82%" shape="rect" coords="1247, 1370, 1271, 1398">e</char>
                        <char conf="82%" shape="rect" coords="1271, 1360, 1288, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1288, 1360, 1303, 1397">f</char>
                        <char conf="82%" shape="rect" coords="1303, 1370, 1327, 1398">e</char>
                        <char conf="82%" shape="rect" coords="1330, 1370, 1354, 1398">e</char>
                        <char conf="100%" shape="rect" coords="1360, 1361, 1364, 1397">l</char>
                        <char conf="100%" shape="rect" coords="1364, 1360, 1382, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1382, 1360, 1397, 1397">f</char>
                        <char conf="100%" shape="rect" coords="1398, 1370, 1412, 1397">r</char>
                        <char conf="84%" shape="rect" coords="1414, 1370, 1438, 1398">e</char>
                        <char conf="82%" shape="rect" coords="1442, 1370, 1466, 1398">e</char>
                        <char conf="82%" shape="rect" coords="1466, 1362, 1482, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1482, 1362, 1494, 1398">t</char>
                        <char conf="100%" shape="rect" coords="1497, 1370, 1521, 1398">o</char>
                        <char conf="100%" shape="rect" coords="1521, 1370, 1539, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1539, 1370, 1561, 1398">c</char>
                        <char conf="100%" shape="rect" coords="1564, 1370, 1587, 1398">a</char>
                        <char conf="100%" shape="rect" coords="1594, 1361, 1598, 1397">l</char>
                        <char conf="100%" shape="rect" coords="1605, 1361, 1609, 1397">l</char>
                        <char conf="100%" shape="rect" coords="1609, 1361, 1630, 1398"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="1630, 1371, 1650, 1398">u</char>
                        <char conf="100%" shape="rect" coords="1655, 1370, 1677, 1398">s</char>
                        <char conf="100%" shape="rect" coords="1684, 1371, 1688, 1397">:</char>
                    </word>
                </line>
            </paragraph>
            <paragraph>
                <line shape="rect" coords="1011, 1475, 1471, 1523">
                    <word class="format-ffArial format-fs12 format-color14942208 format-bold ">
                        <char conf="100%" shape="rect" coords="1011, 1482, 1034, 1506">+</char>
                        <char conf="100%" shape="rect" coords="1039, 1482, 1063, 1506">+</char>
                        <char conf="100%" shape="rect" coords="1069, 1476, 1085, 1512">1</char>
                        <char conf="100%" shape="rect" coords="1094, 1476, 1118, 1512">2</char>
                        <char conf="66%" shape="rect" coords="1136, 1475, 1149, 1523">(</char>
                        <char conf="100%" shape="rect" coords="1153, 1476, 1177, 1513">3</char>
                        <char conf="100%" shape="rect" coords="1180, 1476, 1206, 1512">4</char>
                        <char conf="67%" shape="rect" coords="1208, 1475, 1221, 1523">)</char>
                    <char conf="100%" shape="rect" coords="1241, 1476, 1257, 1512">1</char>
                    <char conf="100%" shape="rect" coords="1257, 1476, 1280, 1512"> </char>
                </word>
                <word class="format-ffArial format-fs12 format-color14942208 format-bold ">
                    <char conf="100%" shape="rect" coords="1280, 1476, 1304, 1512">2</char>
                    <char conf="100%" shape="rect" coords="1309, 1476, 1333, 1513">3</char>
                    <char conf="100%" shape="rect" coords="1333, 1476, 1349, 1512"> </char>
                </word>
                <word class="format-ffArial format-fs12 format-color14942208 format-bold ">
                    <char conf="100%" shape="rect" coords="1349, 1476, 1375, 1512">4</char>
                    <char conf="100%" shape="rect" coords="1378, 1477, 1402, 1513">5</char>
                    <char conf="100%" shape="rect" coords="1402, 1476, 1420, 1513"> </char>
                </word>
                <word class="format-ffArial format-fs12 format-color14942208 format-bold ">
                    <char conf="100%" shape="rect" coords="1420, 1476, 1444, 1513">6</char>
                    <char conf="74%" shape="rect" coords="1448, 1477, 1471, 1512">7</char>
                </word>
            </line>
        </paragraph>
        <paragraph>
            <line shape="rect" coords="376, 1591, 1009, 1638">
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="376, 1591, 422, 1627">W</char>
                    <char conf="84%" shape="rect" coords="424, 1600, 448, 1628">e</char>
                    <char conf="84%" shape="rect" coords="448, 1600, 464, 1628"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="464, 1601, 500, 1627">w</char>
                    <char conf="100%" shape="rect" coords="504, 1591, 508, 1627">i</char>
                    <char conf="100%" shape="rect" coords="512, 1600, 534, 1628">s</char>
                    <char conf="100%" shape="rect" coords="540, 1591, 560, 1627">h</char>
                    <char conf="100%" shape="rect" coords="560, 1591, 579, 1628"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="579, 1601, 603, 1638">y</char>
                    <char conf="100%" shape="rect" coords="605, 1600, 629, 1628">o</char>
                    <char conf="100%" shape="rect" coords="635, 1601, 655, 1628">u</char>
                    <char conf="100%" shape="rect" coords="655, 1600, 674, 1628"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="674, 1600, 697, 1628">a</char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="697, 1600, 716, 1628"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="716, 1600, 739, 1638">g</char>
                    <char conf="100%" shape="rect" coords="745, 1600, 759, 1627">r</char>
                    <char conf="84%" shape="rect" coords="761, 1600, 785, 1628">e</char>
                    <char conf="100%" shape="rect" coords="788, 1600, 811, 1628">a</char>
                    <char conf="100%" shape="rect" coords="815, 1592, 827, 1628">t</char>
                    <char conf="100%" shape="rect" coords="827, 1591, 846, 1628"> </char>
                </word>
                <word class="format-ffArial format-fs12 ">
                    <char conf="100%" shape="rect" coords="846, 1591, 866, 1627">h</char>
                    <char conf="100%" shape="rect" coords="872, 1600, 896, 1628">o</char>
                    <char conf="100%" shape="rect" coords="902, 1591, 906, 1627">l</char>
                    <char conf="100%" shape="rect" coords="913, 1591, 917, 1627">i</char>
                    <char conf="100%" shape="rect" coords="922, 1591, 944, 1628">d</char>
                    <char conf="100%" shape="rect" coords="950, 1600, 973, 1628">a</char>
                    <char conf="100%" shape="rect" coords="976, 1601, 1000, 1638">y</char>
                    <char conf="100%" shape="rect" coords="1004, 1591, 1009, 1627">!</char>
                </word>
            </line>
            </paragraph>
            <paragraph>
                <line shape="rect" coords="379, 1706, 657, 1753">
                    <word class="format-ffArial format-fs12 ">
                        <char conf="88%" shape="rect" coords="379, 1706, 405, 1742">B</char>
                        <char conf="84%" shape="rect" coords="410, 1715, 434, 1743">e</char>
                        <char conf="100%" shape="rect" coords="437, 1715, 459, 1743">s</char>
                        <char conf="100%" shape="rect" coords="462, 1707, 474, 1743">t</char>
                        <char conf="100%" shape="rect" coords="474, 1707, 492, 1743"> </char>
                    </word>
                    <word class="format-ffArial format-fs12 ">
                        <char conf="100%" shape="rect" coords="492, 1715, 506, 1742">r</char>
                        <char conf="84%" shape="rect" coords="508, 1715, 532, 1743">e</char>
                        <char conf="100%" shape="rect" coords="535, 1715, 558, 1753">g</char>
                        <char conf="100%" shape="rect" coords="563, 1715, 586, 1743">a</char>
                        <char conf="100%" shape="rect" coords="592, 1715, 606, 1742">r</char>
                        <char conf="100%" shape="rect" coords="608, 1706, 630, 1743">d</char>
                        <char conf="100%" shape="rect" coords="635, 1715, 657, 1743">s</char>
                    </word>
                </line>
            </paragraph>
        </region>
        <region shape="rect" coords="364, 1862, 830, 2018">
            <paragraph>
                <line shape="rect" coords="380, 1892, 814, 2014">
                    <word class="format-ffTimes-New-Roman format-fs32 format-bold format-italic ">
                        <char conf="1%" shape="rect" coords="380, 1892, 412, 1978">1</char>
                        <char conf="15%" shape="rect" coords="402, 1894, 460, 1986">2</char>
                        <char conf="100%" shape="rect" coords="450, 1914, 570, 1994">^</char>
                        <char conf="28%" shape="rect" coords="586, 1906, 628, 2002">!</char>
                        <char conf="100%" shape="rect" coords="610, 1908, 814, 2014">^</char>
                    </word>
                </line>
            </paragraph>
        </region>
        <region shape="rect" coords="362, 2028, 596, 2080">
            <paragraph>
                <line shape="rect" coords="378, 2036, 579, 2075">
                    <word class="format-ffArial format-fs10 ">
                        <char conf="100%" shape="rect" coords="378, 2036, 401, 2066">B</char>
                        <char conf="100%" shape="rect" coords="404, 2043, 425, 2067">o</char>
                        <char conf="100%" shape="rect" coords="429, 2036, 448, 2067">b</char>
                        <char conf="100%" shape="rect" coords="448, 2036, 464, 2067"> </char>
                    </word>
                    <word class="format-ffArial format-fs10 ">
                        <char conf="100%" shape="rect" coords="464, 2036, 487, 2066">B</char>
                        <char conf="100%" shape="rect" coords="490, 2043, 511, 2067">o</char>
                        <char conf="100%" shape="rect" coords="515, 2036, 534, 2067">b</char>
                        <char conf="100%" shape="rect" coords="538, 2036, 557, 2067">b</char>
                        <char conf="100%" shape="rect" coords="559, 2044, 579, 2075">y</char>
                    </word>
                </line>
            </paragraph>
        </region>
    </hiddentext>
</htx>
```

The following XSLT can be used to translate HTX for viewing in a web browser. It is not the intension of the HTX to replace the original image. Translation to HTML might be used to review the OCR results, or to help in creating JPM viewers that provide copy and paste operations of the hidden text underneath the page image.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    XSL Script for translating from HTX to HTML.

    Tested with
        - Xalan-C++ 1.10.0
        - Internet Explorer 6
        - Mozilla Firefox 1.5

    Author: Andreas Hirth, LuraTech Imaging GmbH, www.luratech.com

    Todo:
        - supporting polygonic bounding shapes

    Note:
        - altchar and altword are skipped
        - absolute positioning is used for every element with coordinates
-->

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:xalan="http://xml.apache.org/xalan"
  xmlns:htx="http://www.jpeg.org/hiddentext/htx"
  exclude-result-prefixes="xsl xsi xhtml xalan htx">

    <xsl:output method="html" indent="yes" xalan:indent-amount="4"/>

    <!-- Make the character enclosing <span> tags be written without linebreak
    or whitespaces in between. This is important because the browser will put
    a blank between each character otherwise. -->
    <xsl:strip-space elements="*"/>

    <!-- Make a <title> tag from a xhtml <title> tag > -->
    <xsl:template match="xhtml:head/xhtml:title">
        <title><xsl:value-of select="text()"/></title>
    </xsl:template>

    <!-- Make a <style> tag from a xhtml <style> tag > -->
    <xsl:template match="xhtml:head/xhtml:style">
        <style type="text/css"><xsl:value-of select="text()"/></style>
    </xsl:template>

    <!-- Translation entry point -->
    <xsl:template match="/">
        <html>
            <head>
                <xsl:apply-templates select="//xhtml:title"/>
                <xsl:apply-templates select="//xhtml:style"/>
            </head>
            <body>
                <xsl:apply-templates select="htx:htx"/>
            </body>
        </html>
    </xsl:template>
    <!-- There is nothing to do with the root element -->
    <xsl:template match="htx:htx">
        <xsl:apply-templates/>
    </xsl:template>
    <!-- Skip annotations -->
    <xsl:template match="htx:annotations"/>

    <!-- There is nothing to do with the hiddentext element -->
    <xsl:template match="htx:hiddentext">
        <xsl:apply-templates/>
    </xsl:template>

    <!-- Translate the <region> element into a <div> tag with core attributes and positioning -->
    <xsl:template match="htx:region">
        <xsl:element name="div">
            <xsl:call-template name="outputCoreAttributes"/>
            <xsl:call-template name="outputPositionAttributes"/>

            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <!-- Translate the <paragraph> element into a <p> tag with core attributes and positioning -->
    <xsl:template match="htx:paragraph">
        <xsl:element name="p">
            <xsl:call-template name="outputCoreAttributes"/>
            <xsl:call-template name="outputPositionAttributes"/>

            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <!-- Translate the <line> element into a <span> tag with core attributes and positioning -->
    <xsl:template match="htx:line">
        <xsl:element name="span">
            <xsl:call-template name="outputCoreAttributes"/>
            <xsl:call-template name="outputPositionAttributes"/>

            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <!-- Translate the <word> element into a <span> tag with core attributes and positioning -->
    <xsl:template match="htx:word">
        <xsl:element name="span">
            <xsl:call-template name="outputCoreAttributes"/>
            <xsl:call-template name="outputPositionAttributes"/>

            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <!-- Translate the <char> element into a <span> tag with core attributes and
    special positioning -->
    <xsl:template match="htx:char">
        <xsl:element name="span">
            <xsl:call-template name="outputCoreAttributes"/>
            <xsl:call-template name="outputCharacterPositionAttributes"/>

            <xsl:value-of select="text()"/>
        </xsl:element>
    </xsl:template>

    <!-- Do not translate any other elements, like <altword> or <altchar> -->
    <xsl:template match="*"/>

    <!-- Parse and output the core attributes of an element -->
    <xsl:template name="outputCoreAttributes">
        <xsl:if test="@class">
            <xsl:attribute name="class"><xsl:value-of select="@class"/></xsl:attribute>
        </xsl:if>

        <xsl:if test="@lang">
            <xsl:attribute name="lang"><xsl:value-of select="@lang"/></xsl:attribute>
        </xsl:if>

        <xsl:if test="@xml:lang">
            <xsl:attribute name="xml:lang"><xsl:value-of select="@xml:lang"/></xsl:attribute>
        </xsl:if>

        <xsl:if test="@dir">
            <xsl:attribute name="dir"><xsl:value-of select="@dir"/></xsl:attribute>
        </xsl:if>

        <xsl:if test="@id">
            <xsl:attribute name="id"><xsl:value-of select="@id"/></xsl:attribute>
        </xsl:if>
    </xsl:template>

    <!-- Parse and output positioning attributes -->
    <xsl:template name="outputPositionAttributes">
        <xsl:choose>
            <xsl:when test="@shape='poly'">
                <xsl:message>Polygonic shapes are not supported yet. No positioning attributes are
written for element &lt;<xsl:value-of select="name()"/>&gt;. </xsl:message>
            </xsl:when>
            <xsl:otherwise>
                <xsl:call-template name="outputRelativeCoordinatesAttribute"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>

    <!-- Parse and output positioning attributes especialy for characters -->
    <xsl:template name="outputCharacterPositionAttributes">
        <xsl:choose>
            <xsl:when test="@shape='poly'">
                <xsl:message>Polygonic shapes are not supported yet. No positioning attributes are
written for element <xsl:value-of select="name()"/></xsl:message>
            </xsl:when>
            <xsl:otherwise>
                <xsl:call-template name="outputRelativeCharacterCoordinatesAttribute"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>

    <!-- Calculate and output positioning attributes from coords. Must be recalculated to be
    relative to the nearest parent with coordinates. -->
    <xsl:template name="outputRelativeCoordinatesAttribute">
        <xsl:variable name="horRes">
            <xsl:choose>
                <xsl:when test="/htx:htx[(position()=1) and (@res)]">
                    <xsl:variable name="res"><xsl:value-of
select="/htx:htx[1]/@res"/></xsl:variable>
                    <xsl:choose>
                        <xsl:when test="contains($res, ',')">
                            <xsl:value-of select="normalize-space(substring-before($res, ','))"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$res"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>0</xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="vertRes">
            <xsl:choose>
                <xsl:when test="/htx:htx[(position()=1) and (@res)]">
                    <xsl:variable name="res"><xsl:value-of
select="/htx:htx[1]/@res"/></xsl:variable>
                    <xsl:choose>
                        <xsl:when test="contains($res, ',')">
                            <xsl:value-of select="normalize-space(substring-after($res, ','))"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$res"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>0</xsl:otherwise>
            </xsl:choose>
        </xsl:variable>

        <xsl:variable name="left">
            <xsl:variable name="coord"><xsl:value-of select="number(normalize-space(substring-
before(@coords, ',')))"/></xsl:variable>
            <xsl:choose>
                <xsl:when test="number($horRes) &gt; 0">
                    <xsl:value-of select="$coord div $horRes"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$coord"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="top">
            <xsl:variable name="coord"><xsl:value-of select="number(normalize-space(substring-
before(substring-after(@coords, ','), ',')))"/></xsl:variable>
            <xsl:choose>
                <xsl:when test="number($vertRes) &gt; 0">
                    <xsl:value-of select="$coord div $vertRes"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$coord"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>

        <xsl:if test="(@shape='rect') and (string-length($left)>0) and (string-length($top)>0)">
            <xsl:variable name="parentLeft">
                <xsl:choose>
                    <xsl:when test="ancestor::*[(@shape='rect') and (@coords)]">
                        <xsl:variable name="coord"><xsl:value-of select="number(normalize-
space(substring-before(ancestor::*[(@shape='rect') and (@coords)][1]/@coords,
',')))"/></xsl:variable>
                        <xsl:choose>
                            <xsl:when test="number($horRes) &gt; 0">
                                <xsl:value-of select="$coord div $horRes"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="$coord"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:when>
                    <xsl:otherwise>0</xsl:otherwise>
                </xsl:choose>
            </xsl:variable>

            <xsl:variable name="parentTop">
                <xsl:choose>
                    <xsl:when test="ancestor::*[(@shape='rect') and (@coords)]">
                        <xsl:variable name="coord"><xsl:value-of select="number(normalize-
space(substring-before(substring-after(ancestor::*[(@shape='rect') and (@coords)][1]/@coords,
','), ',')))"/></xsl:variable>
                        <xsl:choose>
                            <xsl:when test="number($vertRes) &gt; 0">
                                <xsl:value-of select="$coord div $vertRes"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="$coord"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:when>
                    <xsl:otherwise>0</xsl:otherwise>
                </xsl:choose>
            </xsl:variable>

            <xsl:variable name="unit">
                <xsl:choose>
                    <xsl:when test="(number($horRes) &gt; 0) and (number($vertRes) &gt;
0)">in</xsl:when>
                    <xsl:otherwise>px</xsl:otherwise>
                </xsl:choose>
            </xsl:variable>
            <xsl:attribute name="style">position:absolute; left:<xsl:value-of select="$left -
$parentLeft"/><xsl:value-of select="$unit"/>; top:<xsl:value-of select="$top -
$parentTop"/><xsl:value-of select="$unit"/>;</xsl:attribute>
        </xsl:if>
    </xsl:template>

    <!-- Calculate and output positioning attributes from coords especialy for characters.
    Must be recalculated to be relative to the nearest parent with coordinates. The top coordinate
    is used from the parent due to get straight lines because the bounding boxes of characters in
    HTML are equal height for large and small letters of a font. -->
    <xsl:template name="outputRelativeCharacterCoordinatesAttribute">
        <xsl:variable name="horRes">
            <xsl:choose>
                <xsl:when test="/htx:htx[(position()=1) and (@res)]">
                    <xsl:variable name="res"><xsl:value-of
select="/htx:htx[1]/@res"/></xsl:variable>
                    <xsl:choose>
                        <xsl:when test="contains($res, ',')">
                            <xsl:value-of select="normalize-space(substring-before($res, ','))"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$res"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>0</xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="vertRes">
            <xsl:choose>
                <xsl:when test="/htx:htx[(position()=1) and (@res)]">
```

```
                    <xsl:variable name="res"><xsl:value-of
select="/htx:htx[1]/@res"/></xsl:variable>
                    <xsl:choose>
                        <xsl:when test="contains($res, ',')">
                            <xsl:value-of select="normalize-space(substring-after($res, ','))"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$res"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>0</xsl:otherwise>
            </xsl:choose>
        </xsl:variable>

        <xsl:variable name="left">
            <xsl:variable name="coord"><xsl:value-of select="number(normalize-space(substring-
before(@coords, ',')))"/></xsl:variable>
            <xsl:choose>
                <xsl:when test="number($horRes) &gt; 0">
                    <xsl:value-of select="$coord div $horRes"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$coord"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="top">
            <xsl:variable name="coord"><xsl:value-of select="number(normalize-space(substring-
before(substring-after(@coords, ','), ',')))"/></xsl:variable>
            <xsl:choose>
                <xsl:when test="number($vertRes) &gt; 0">
                    <xsl:value-of select="$coord div $vertRes"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$coord"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>

        <xsl:if test="(string-length($left)>0) and (string-length($top)>0)">
            <xsl:variable name="parentLeft">
                <xsl:choose>
                    <xsl:when test="ancestor::*[(@shape='rect') and (@coords)]">
                        <xsl:variable name="coord"><xsl:value-of select="number(normalize-
space(substring-before(ancestor::*[(@shape='rect') and (@coords)][1]/@coords,
',')))"/></xsl:variable>
                        <xsl:choose>
                            <xsl:when test="number($horRes) &gt; 0">
                                <xsl:value-of select="$coord div $horRes"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="$coord"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:when>
                    <xsl:otherwise>0</xsl:otherwise>
                </xsl:choose>
            </xsl:variable>

            <xsl:variable name="parentTop">
                <xsl:choose>
                    <xsl:when test="ancestor::*[(@shape='rect') and (@coords)]">
```

```
                        <xsl:variable name="coord"><xsl:value-of select="number(normalize-
space(substring-before(substring-after(ancestor::*[(@shape='rect') and (@coords)][1]/@coords,
','), ',')))"/></xsl:variable>
                        <xsl:choose>
                            <xsl:when test="number($vertRes) &gt; 0">
                                <xsl:value-of select="$coord div $vertRes"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="$coord"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:when>
                </xsl:choose>
            </xsl:variable>

            <xsl:variable name="finalTop">
                <xsl:choose>
                    <xsl:when test="string-length($parentTop) > 0">0</xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="$top"/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:variable>

            <xsl:variable name="unit">
                <xsl:choose>
                    <xsl:when test="(number($horRes) &gt; 0) and (number($vertRes) &gt;
0)">in</xsl:when>
                    <xsl:otherwise>px</xsl:otherwise>
                </xsl:choose>
            </xsl:variable>
            <xsl:attribute name="style">position:absolute; left:<xsl:value-of select="$left -
$parentLeft"/><xsl:value-of select="$unit"/>; top:<xsl:value-of
select="number($finalTop)"/><xsl:value-of select="$unit"/>;</xsl:attribute>
        </xsl:if>
    </xsl:template>
</xsl:stylesheet>
```

Annex J

Guidelines for constructing URLs for JPM files

(This annex does not form an integral part of this Recommendation | International Standard.)

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

Rec. ITU-T T.808 | ISO/IEC 15444-9 will define further how to remotely access JPM files. Implementers should refer to Rec. ITU-T T.808 | ISO/IEC 15444-9 for additional guidance on constructing URLs.

These URLs are constructed by appending a question mark ("?") character to the path and file name of the JPM file and then appending attribute value pairs joined by equal signs ("="), separated by ampersands ("&"), that identify the sub-element for which access is desired.

J.1 Pages and layout objects

This shows how to reference a single layout object on a given page via its LObjID.

foo.jpm?page=3&obj=32

page=3 is a page reference

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

J.2 Metadata boxes

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

foo.jpm?page=43&type=meta

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

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

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

foo.jpm?page=91&type=meta&mtype=text

J.3 Labels

Label boxes may be used in a JPM file to attach labels to other boxes in the file. Metadata, such as an XML box for example, may also be attached to these same boxes. In such a case, the metadata is returned to the requester of the URL rather than any image data. XHTML pages may occur inside an XML box, since they represent valid XML data.

foo.jpm?label=album.html

foo.jpm?label=IPR-info&type=meta&index=2

J.4 Page collections

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

foo.jpm?coll=toc

foo.jpm?coll=main

foo.jpm?coll=list-of-figs

foo.jpm?coll=list-of-tables foo.jpm?coll=toc&page=31 foo.jpm?coll=main&page=7

foo.jpm?page=7

foo.jpm?

J.5 Page thumbnails

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

foo.jpm?page=3&type=img

foo.jpm?page=3&type=thumb

J.6 Document thumbnail

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

foo.jpm?page=thumb

J.7 Byte ranges

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

foo.jpm?off=12384&len=1024

Printed in Switzerland
Geneva, 2013