Technical Report

ISO/IEC TR 19566-9

Information technology — JPEG Systems —

Part 9: JPEG extensions mechanisms to facilitate forwards and backwards compatibility

Technologies de l'information — Systèmes JPEG —

Partie 9: Mécanismes d'extension JPEG pour faciliter la compatibilité ascendante et descendante

First edition
2024-08

Reference number
ISO/IEC TR 19566-9:2024(en)

Contents

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

A list of all parts in the ISO/IEC 19566 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document collects guiding principles on how standards discussed in ISO/IEC TR 19566-1 provide mechanisms for forwards compatibility, so called extension mechanisms, both on the basis of the codestream and the file format and lists specific implementations of these guiding principles in particular standards.

The purpose of this document is to provide documentation on these principles for the preparation of future extensions of these standards, and to ensure consistency of extension principles amongst standards.

Information technology — JPEG Systems —

Part 9: JPEG extensions mechanisms to facilitate forwards and backwards compatibility

1 Scope

This document summarizes mechanisms by which existing file formats and codestreams, such as those specified in ISO/IEC TR 19566-1, can be extended in a forward and backward-compatible way.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Rec. ITU-T T.800 | ISO/IEC 15444-1, *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*

Rec. ITU-T T.801 | ISO/IEC 15444-2, *Information technology — JPEG 2000 image coding system — Part 2: Extensions*

Rec. ITU-T T.802 | ISO/IEC 15444-3, *Information technology — JPEG 2000 image coding system — Part 3: Motion JPEG 2000*

ISO/IEC 18181-1, *Information technology — JPEG XL image coding system — Part 1: Core coding system*

ISO/IEC TR 19566-1, *Information technology — JPEG Systems — Part 1: Packaging of information using codestreams and file formats*

ISO/IEC 21122-1, *Information technology — JPEG XS low-latency lightweight image coding system — Part 1: Core coding system*

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in Rec. ITU-T T.800 | ISO/IEC 15444-1, Rec. ITU-T T.801 | ISO/IEC 15444-2, Rec. ITU-T T.802 | ISO/IEC 15444-3, ISO/IEC 18181-1, ISO/IEC TR 19566-1, ISO/IEC 21122-1 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
codestream
compressed image data representation which includes all necessary data to allow a (full or approximate) reconstruction of the sample values of a digital image, which can require additional data to define the interpretation of the sample data, such as colour space or the spatial dimensions of the samples

3.1.2
file format
encapsulation of a *codestream* (3.1.1) in a file, that is in a form that allows random access from a file system of a computer system, along with additional metadata that define the interpretation of the samples reconstructed by the codestream within

3.1.3
encoder
embodiment of an encoding process, which takes digital source image data and encoder specifications as input and, by means of a specified set of procedures, generates *a codestream* (3.1.1) or a file as output

3.1.4
data producer
encoder generating *file formats* (3.1.2)

3.1.5
decoder
embodiment of a *decoding process*, which takes a *codestream* or a *file* as input and, by means of a specified set of procedures, generates *digital reconstructed image data* as output

3.1.6
data consumer
decoder capable of parsing file formats

3.1.7
forward compatibility
design principle allowing future extensions in existing codestreams or files

3.1.8
backward compatibility
design principle for extensions which allows existing decoders to successfully reconstruct data (i.e. images) from extended codestreams and files, albeit some information embedded in these codestreams or files are potentially not be fully accessible to them

3.2 Abbreviated terms

ID	Identifier
JP2	JPEG 2000 file format
JPEG	Joint Photographic Experts Group
JPIP	JPEG 2000 Interactive Protocol
XML	Extensible Markup Language

4 Conventions - Operators

NOTE Many of the operators used in this document are similar to those used in the C programming language.

4.1 Arithmetic operators

+	Addition
−	Subtraction (as a binary operator) or negation (as a unary prefix operator)
*	Multiplication
/	Division without truncation or rounding.
smod	x smod a is the unique value y between $-\lceil (a-1)/2 \rceil$ and $\lfloor (a-1)/2 \rfloor$ for which y+Na = x with a suitable integer N.
umod	x mod a is the unique value y between 0 and a-1 for which y+Na = y with a suitable integer N.

4.2 Logical operators

||	Logical OR
&&	Logical AND
!	Logical NOT
∈	x ∈ {A, B} is defined as (x == A || x == B)
∉	x ∉ {A, B} is defined as (x != A && x != B)

4.3 Relational operators

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

4.4 Precedence order of operators

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

Operators	**Type of operation**	**Associativity**
(), [ ], .	Expression	Left to Right
−	Unary negation	
*, /	Multiplication	Left to Right
umod, smod	Modulo (remainder)	Left to Right
+, −	Addition and Subtraction	Left to Right
< , >, <=, >=	Relational	Left to Right

4.5 Mathematical functions

$\lceil x \rceil$	Ceil of x. Returns the smallest integer that is greater than or equal to x.
$\lfloor x \rfloor$	Floor of x. Returns the largest integer that is lesser than or equal to x.
$\lvert x \rvert$	Absolute value, is –x for x < 0, otherwise x.
sign(x)	Sign of x, zero if x is zero, +1 if x is positive, -1 if x is negative.
clamp(x,min,max)	Clamps x to the range [min,max]: returns min if x < min, max if x > max or otherwise x.
power(x,a)	Raises the value of x to the power of a. x is a non-negative real number, a is a real number. Power(x,a) is equal to exp(a×log(x)) where exp is the exponential function and log() the natural logarithm. If x is zero and a is positive, power(x,a) is defined to be zero.

5 General extensions mechanisms

5.1 General

This clause lists general best practices for cross-standard extensions mechanism on the codestream and on the file format level.

5.2 Extensions mechanisms for codestreams

As specified in ISO/IEC TR 19566-1, codestreams consist of multiple markers and marker segments. While markers stand alone, marker segments include a size and following data that allow readers of such codestreams to skip over data they do not intend to interpret or do not understand.

Generally, the following two types of marker segments are present:

— A *capability marker segment* that allows registrations of future extensions. Such a marker segment allows a reader of a particular codestream to identify which extensions are required for a successful decode, and to abort decoding if it cannot provide the required extensions. That is, capability marker segments provide *early-out* conditions to decoders.

— Purely informative *comment marker segments* that contain additional information a decoder can safely skip over without compromising the decoding process, but that can be informative to the user of the media reconstructed by the codestream. Such marker sequences can, for example, embed information on the software used to create the codestream. A decoder can potentially use this information to enable workarounds for known vendor-specific defects or display such information upon request of its user.

— Comment markers can be further classified into vendor-specific information, and vendor neutral information.

Marker sequences often contain bit-fields that enable or disable particular functionalities of a decoder. As the size of such bit fields is often aligned to multiples of 8 bits, a particular edition of a standard will not always not require all such bits. Unused bits are reserved for future use by ITU-T/ISO/IEC purposes, and require encoders to write them as 0. Depending on application, decoders can be either required to abort on bits that are defined as reserved or ignore them. It can be advisable to reserve one bit (e.g. the topmost bit of such a bit field) to extend the size of the bit field in future generations of a standard.

The same bitfield can be defined within multiple parts of a series of standards, with some bits only applicable to a specific part of a standard. It is advisable that those bits within the bitfield that are not used within a particular standard are marked as “Reserved for ISO/IEC purposes”, and that their default values are selected as 0 such that an encoder conforming to a part of a standard series writes them as 0 without compromising the codestream for decoders that support multiple parts of the same series.

The following example lists a bitfield whose topmost bit is used to signal a syntax extension whose origin is signalled by bits 2 and 3, if it is set, and which is 0 if base signalling is used. For base signalling, bits 6 to 2 are currently unused and reserved, and for extended signalling, bits 6 to 4 are reserved. Bits 0 and 1 signal optional features by the selector bits 7 and (optionally) bits 2 and 3. Assume further that the base signalling is specified in one part of the standard, and the extension signalling in a further part.

Then, the specification of the bitfield in the base system would read as shown in Table 1.

Table 1 — Example of a bit field (base part)

Field value	Function
`0000 00x0`	Disable feature A
`0000 00x1`	Enable feature A
`0000 000x`	Disable feature B
`0000 001x`	Enable feature B
All other combinations	Reserved

In particular, by this bit combinations that employ bits for extended signalling are marked as "Reserved" in the base part. Assume further that the extension part of the standard also employs "feature A" and "feature B" which are mutually exclusive to features C and D of the extended. The extension part would then define a meaning for such extension bits, while preserving the meaning of the existing bits as shown in Table 2.

Table 2 — Example of a bit field (extension part)

Field value	Function
`0000 00x0`	Disable feature A
`0000 00x1`	Enable feature A
`0000 000x`	Disable feature B
`0000 001x`	Enable feature B
`1000 01x0`	Disable feature C
`1000 01x1`	Enable feature C
`1000 010x`	Disable feature D
`1000 011x`	Enable feature D
All other combinations	Reserved

If, however, the extended features C and D of the extension part are orthogonal to features A and B of the base part, the above signalling cannot be used and instead the extension part will allocate additional bits from the originally reserved bit set for its purpose. In such a case, the specification of the bit field can read as shown in Table 3.

Table 3 — Example of a bit field (extension part)

Field value	Function
`x0xx xxx0`	Disable feature A
`x0xx xxx1`	Enable feature A
`x0xx xx0x`	Disable feature B
`x0xx xx1x`	Enable feature B
`10x0 01xx`	Disable feature C
`10x1 01xx`	Enable feature C
`100x 01xx`	Disable feature D
`101x 01xx`	Enable feature D
All other combinations	Reserved

In this definition, bits 2 and 3 would still identify the part within which the extended features C and D are defined, whereas the features itself are signalled by bits 4 and 5. Yet another part of the same series of specification can then define features E and F, which are mutually exclusive to features C and D, but orthogonal to A and B of the base part as shown in Table 4.

Table 4 — Example of a bit field (second extension part)

Field value	Function
`x0xx xxx0`	Disable feature A
`x0xx xxx1`	Enable feature A
`x0xx xx0x`	Disable feature B
`x0xx xx1x`	Enable feature B
`10x0 10xx`	Disable feature E
`10x1 10xx`	Enable feature E
`100x 10xx`	Disable feature F
`101x 10xx`	Enable feature F
All other combinations	Reserved

The bits 2 and 3 are here different from the table specified in the first extension, allowing decoders to distinguish between the feature set C,D of the first extension, and feature set E,F of the second extension.

5.3 Extension mechanisms for file formats

It is advisable to base file format standards on the box-based file format as for example JPX, specified in Rec. ITU-T T.801 | ISO/IEC 15444-2, or ISOBMFF, as contained in ISO/IEC 14496-12. Both file formats provide syntax elements known as boxes, see ISO/IEC TR 19566-1 for further details, and the encoding of boxes is identical in both specifications. It is generally advisable to specify a box-based file format in such a way that decoders are instructed to skip over boxes they do not implement.

In addition, it is advisable to equip standards with the following specific boxes:

— A *signature box* as first box in the file format that allows to identify the family of standards a file conforms to.

— A *file type box* that identifies those standards the contents are compatible to. A reader implementing one of the listed standards in the compatibility list of standards is expected to make productive use of the contents of the file. A file type box enables decoders to abort decoding quickly, and/or allows applications to select a decoder suitable for a particular file. Such file type boxes can list multiple standards or profiles of standards, and as such also extensions to standards that would be required for a partial or full decoding.

  Rec. ITU-T T.801 | ISO/IEC 15444-2 defines a structure for a file type box using a '**ftyp**' as box type that is encouraged to be used in new specifications. It provides "summary" level information to help the user determine whether it can read and make productive use of the file. Subclause 5.4 provides additional guidelines on the contents of this box.

— File formats based on the JPX format carry an **Image Header box** of type '**ihdr**' that, along with the image dimensions and number of channels, also carry a compression type **C** that identifies the type of the codestream within the file. It is advisable to keep this codestream identifier unique and consistent throughout specifications. Registration of these codestream formats is currently through the definition of JPX, within Rec. ITU-T T.801 |ISO/IEC 15444-2. The value to be used for this field is specified in the corresponding part of the standard defining a JPX based file format. The compression types **C** are kept up to date in Rec. ITU-T T.801 | ISO/IEC 15444-2.

— The codestream carrying the encoded sample values is wrapped in a **Contiguous Codestream box** of type '**jp2c**'. The box type does not depend on the codestream contained in the box, i.e. it is always 'jp2c' even if the codestream is not based on Rec. ITU-T T.80x | ISO/IEC 15444-x. Instead, identification of the codestream syntax is through the **C** field of the **Image Header box** associated to the particular codestream.

- It is advisable to place vendor-specific extensions in *UUID boxes*. This box type identifies extension data by means of a 128-bit identifier. UUIDs to identify vendor-specific data are self-registered and do not require registration through a standardization organization. For example, the `uuidgen` command line tool available on many operating systems can be used to create a unique UUID.
- Another mechanism by which generic metadata, and in particular also vendor specific metadata can be embedded into a file is through XML boxes. Such boxes contain UTC encoded XML data following a particular XML dialect.
- Finally, ISO/IEC 19566-5 defines the JPEG Universal Metadata Box Format (JUMBF). JUMBF specifies a metadata embedding framework that is format agnostic. JUMBF defines a generic container, compatible with ISOBMFF, that can be used to embed domain specific metadata that can be textual or binary, including media content. JUMBF also defines functionalities that facilitate linking between the metadata elements themselves and media content. It also specifies a URL schema that can be used for external references and/or requests into a media asset. Subclause 7.4 illustrates the JUMBF structure in more detail.

5.4 File Type box

The box structure and the ‘ftyp’ box type inform the user that the file is constructed as a set of boxes and can be parsed as such. All files include the File Type box behind the signature box, as recognizing the File Type box allows the reader to catalog the file, even if it does not recognize most of the content of the file (e.g., the reader can find metadata even if it cannot decode or interpret the pixel data).

- The brand field informs the reader of the specification against which this file is conformant. While specifications define the brand for when the file is completely specified by that standard, data producers can define new brands and their definition is not limited to ISO/IEC; conformance to a particular standard is solely defined through the compatibility list within the file type box and not through the brand field. Standards are formulated in such a way that data consumers, e.g. decoders, are instructed to check the elements of the compatibility list to ensure that they are able to make productive use of a file, and not on the brand value. A data consumer can continue to parse a file if it finds an element on the compatibility list that indicates a standard or a profile it implements.

  Currently, the following brand values are found in ISO specifications as shown in Table 5.

Table 5 — Known ISO brand values

Brand value	Specification
'jp2\040'	Rec. ITU-T T.800 | ISO/IEC 15444-1
'jpx\040'	Rec. ITU-T T.801 | ISO/IEC 15444-2
'jph\040'	Rec. ITU-T T.814 | ISO/IEC 15444-15
'jxl\040'	ISO/IEC 18181-2
'jxs\040'	ISO/IEC 21122-3
'jpxt'	ISO/IEC 18477-3

EXAMPLE 1 A JPX file is written from a fictiuous vendor, Camera Model 10. While not formally documented in a JPEG standard, its vendor can choose to define a new brand field “AI10” indicating that the image was written according to the specification defining that camera, including all metadata or specific features of that camera. Data consumers do not generally use the brand field to determine if they can read or make productive use of the file, favoring the compatibility list instead.

- Standards contain a list that identifies which identifiers in the compatibility list correspond to which conformance points, e.g. profiles, of a standard. Data producers, e.g. encoders, create the compatibility list of the file type box in such a way that it indicates all conformance points the file conforms to, thus enabling data consumers to quickly check their ability to make productive use of the file.
- Data producers can also define new compatibility codes for their own purpose as a conformant data consumers will ignore compatibility codes it does not support and is advised to only check whether the file formats it supports is on the list of compatibility codes.

— The compatibility list specifies one or more codes indicating specific conformance sets or profiles of the standard (defined by ISO or defined by vendors) to which this file conforms. If the data consumer recognizes any code within that compatibility list and has been coded to correctly interpret that profile, then the consumer can read and make productive use of the file.

EXAMPLE 2 The file from the Camera Model 10 can be a fully conformant JP2 or JPX Baseline file, and thus lists both 'jp2 ' and 'jpxb' in the compatibility list. While it is possible that a data consumer does not recognize the brand (and thus not know exactly what is special about this file), it likely recognizes those two compatibility profiles and knows that it can read and make productive use of the file.

5.5 JPEG Universal Metadata Box Format (JUMBF)

A JUMBF box is a specific box type carrying metadata that can be embedded either directly into box-based file formats such as JPX or ISOBMFF, or can be embedded into a JPEG file by means of the APP11 marker specified in clause 8.2. Its layout is as follows.

A JUMBF box is a superbox of which the first child box is always the JUMBF Description box ('jumd'). The JUMBF Description box is followed by one or more custom type content boxes as illustrated in Figure 1.

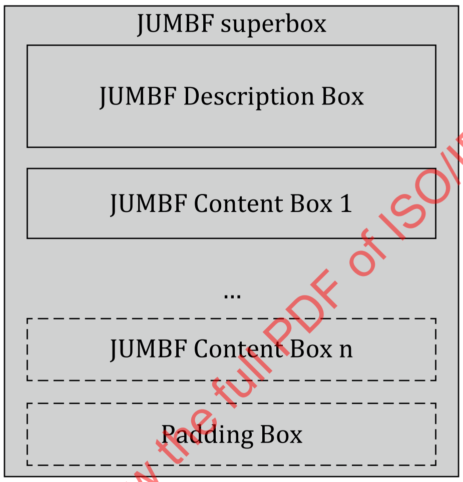

Figure 1 — Illustration of the structure of a JUMBF superbox and its content.

The JUMBF Description box signals the type and properties of the JUMBF box, as illustrated in Figure 2. The type implies the nature of the following child boxes and is signaled by a 128-bit unique identifier or UUID. Some specific types are defined within the JUMBF specification, but additional types can be defined by third parties for dedicated applications. Native JUMBF types include XML, JSON, codestreams and binary files. In addition, dedicated types are defined in other JPEG Systems specifications such as JPEG Privacy & Security and JPEG 360.

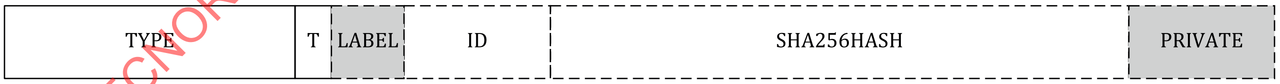

Figure 2 — Illustration of the JUMBF Description box structure.

In addition to the type, several properties can be enabled or disabled via a toggles byte (T), a byte of which each bit represents a binary selection of a specific property. These properties include the presence of a label, a numeric identifier, and a hash signature. They enable functionalities such as referencing, making requests and content integrity checking.

5.6 Compressed boxes

General-purpose compression (Brotli, IETF RFC 7932) can be applied to arbitrary ISOBMFF boxes by using the **brob** Box type as defined in 18181-2. The **brob** Box helps to improve density of compression in the presence of ICC color profiles, XML-based metadata like XMP, and other complex metadata (e.g. JUMBF), in particular when this metadata is represented using plaintext formats like XML, JSON, etc. This approach can bring some challenges to interoperability during the time when tools are catching up with the **brob** Box support.

The mechanism to compress boxes is currently defined in ISO/IEC 18181-3 (JPEG XL), though is generic enough to be adopted by other standards in the future as well. Rec. ITU-T T.808 | ISO/IEC 15444-9 (JPIP) does not currently support this box type.

6 Extension mechanisms for ITU Recommendation T.81 | ISO/IEC 10918-1 (JPEG 1)

6.1 Application markers

ITU Recommendation T.81 | ISO/IEC 10918-1 is equipped with 16 *application markers* using the identifiers `0xffe0` through `0xffef`. These markers introduce marker sequences, i.e. they can carry up to 64K payload data. While the above document reserves these application markers for vendor-specific extensions, at least two markers have been standardized by ISO, namely:

— The APP0 marker (`0xffe0`) is used to introduce files conforming to ISO/IEC 10918-5 (JFIF) and is not to be used for any other type of extension.

— While not standardized through ISO, but through CIPA, the APP1 marker (`0xffe1`) is used to introduce files conforming to EXIF and is not to be used for any other type of extension.

— The APP11 marker (`0xffeb`) standardized by ISO provides a generic extension mechanisms by allowing to embed JPX or ISOBMFF conforming boxes into a Rec. ITU-T T.81 | ISO/IEC 10918-1 conforming codestream. The embedding mechanism based on this marker is specified in ISO/IEC 18477-3.

As the application marker space is limited to 16 markers, it is advisable to enable future extensions by means of mechanisms based on the above marker, i.e. by the mechanism offered by ISO/IEC 18477-3. This mechanism allows embedding boxes based on the JPX/ISOBMFF syntax into JPEG codestreams, and is also used by generic metadata extension mechanisms such as JUMBF.

Only if this generic extension mechanism is not suitable, another application marker can be allocated. Since most of the APP markers are already put in some use, it is advisable to provide additional mechanisms to identify the extension mechanism, for example by placing particular byte sequences ahead of the payload data identifying their encoding.

6.2 APP11 based extensions

The APP11 marker along with the specifications in ISO/IEC 18477-3 provides a generic mechanism to embed box-based data into codestreams. As marker space is limited, ISO/IEC 18477-3 identifies this extension mechanism by the first two bytes of the marker payload data being `0x4a 0x50` giving the marker the following layout (see ISO/IEC 18477-3 for details) as shown in Table 6.

Table 6 — Layout of the APP11 marker

Name	Value	Size in bits	Notes
APP11	0xffeb	16	Identifies this marker
Le	2—65535	16	Size of this marker segment, including the length, but not the marker itself
CI	0x4a50	16	Identifies this application marker
En	0—65535	16	Box instance number, identifies the box of multiple boxes with identical box type
Z	0—65535	16	Packet sequence number, identifies the box contents in sequence. Content of one box type with identical box instance number are concatenated in increasing Z order to form the box.
LBox	0—4294967295	32	Box length. Length of the box including box header and payload data.
TBox	0—4294967295	32	Box type, identifies the type of payload data
XLBox	0—9223372036854775807	0 or 64	Optional field if LBox = 0, identifies the box length if the size of the box exceeds 32 bits.
Payload data	Variable	Variable	Payload data of the box, to be concatenated for the same box type and box enumeration in increasing Z order.

6.3 JUMBF Boxes

JUMBF Boxes can be embedded into JPEG 1 files by means of APP11 marker segments as specified in subclause 8.2.

7 Extension mechanisms for Rec. ITU-T T.800 | ISO/IEC 15444 (JPEG 2000) series

7.1 General

The ISO/IEC 15444 series is based on a codestream and a file format following the principles of Clause 5. However, the marker range allocated for the ISO/IEC 15444 series is separated into two groups:

— The first group, using marker IDs below `0xff90` indicates markers or marker segments that can only be present outside of entropy coded segments.

— The second group, from `0xff90` and up, can also be present within entropy coded data and can hence interrupt or terminate it.

This marker space allocation is due to the bit-stuffing procedure in both of the MQ-Coder deployed in Rec. ITU-T T.800 | ISO/IEC 15444-1 and the HT block coder defined in Rec. ITU-T T.814 | ISO/IEC 15444-15 that cannot generate two-byte sequences that make up marker IDs of the second type. That is, the bit stuffing procedure ensures that a byte of value `0xff` is followed by a value of `0x8f` or below.

7.2 CAP marker

The CAP marker, marker ID `0xff50`, specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, specifies additional capabilities on a codestream level a decoder requires to successfully decode a codestream.

Signalling bits of the CAP marker segment are allocated by the part number. The 32 bit field **Pcap** of the capabilities marker segment indicates the part whose capabilities are present in the marker segment. **Pcap** bits are allocated from MSB to LSB, such that the MSB bit of **Pcap** signals whether any extended capabilities of Rec. ITU-T T.800 | ISO/IEC 15444-1 are present, and the next most significant bit indicates whether any extended capabilities of Rec. ITU-T T.801 | ISO/IEC 15444-2 are present, etc.

For each **Pcap** bit set, a 16 bit $\mathbf{Ccap}^{\mathbf{i}}$ field is present which indicates extended syntax elements within the selected part. The assignment of bits in $\mathbf{Ccap}^{\mathbf{i}}$ is specified in Rec. ITU-T T.800 + (i-1) | ISO/IEC 15444 part i.

It is advisable to indicate new capabilities by allocating bits from the $\mathbf{Ccap}^{\mathbf{i}}$ field, and signalling the presence of such extensions by writing a CAP marker with the allocated bit set.

If a codestream is created for a use case in which it is desired to induce Rec. ITU-T T.800 | ISO/IEC 15444-1 decoders to attempt best-effort decoding of a codestream containing features defined outside Rec. ITU-T T.800 | ISO/IEC 15444-1, the codestream are created such that the second-most-significant bit in Rsiz is set to 0 in order to avoid signalling the presence of the CAP marker segment. According to Rec. ITU-T T.800 | ISO/IEC 15444-1, decoder implementations are expected to deal with unrecognized marker segments by using the length parameter to discard the marker segment and thus could disregard the CAP marker segment.

Because of this possibility, decoders that recognize any of the capabilities that would be indicated in a CAP marker segment can choose to parse for the CAP marker segment in all codestreams, regardless of the value of the second-most-significant bit in Rsiz. The CAP marker segment always appears before any other marker segments that support extended capabilities indicated in the CAP marker segment, so that such capabilities will have been indicated to the decoder prior to encountering any related marker segments.

If a codestream is created for a use case in which it is desired to deter Rec. ITU-T T.800 | ISO/IEC 15444-1 decoders from attempting to decode a codestream that truly requires capabilities indicated in the CAP marker segment, the codestream would ordinarily be constructed such that the second-most-significant bit in Rsiz is set to 1.

7.3 Profile indicators

Profile indicators within Rec. ITU-T T.800 | ISO/IEC 15444-1 are allocated from the **Rsiz** field in the **SIZ** marker segment, marker ID `0xff51`, and the **PRF** marker segment, marker ID `0xff56`. The particular mechanism is profile dependent and is signalled by the topmost 2 bits of the **Rsiz** field:

— If the MSB (bit 15) of the **Rsiz** field is 1, bits 11 to 0 signal the presence of codestream elements conforming to Rec. ITU-T T.801 | ISO/IEC 15444-2. A set bit indicates the presence of a particular feature. In particular, all remaining bits then do not signal a profile anymore, but instead indicate codestream features. In such a case, the profile indicators of Rec. ITU-T T.800 | ISO/IEC 15444-1 do not apply.

— If bit 14 of the **Rsiz** field is 1, this indicates the presence of a capability marker segment (**CAP** marker) which can signal additional codestream elements of Rec. ITU-T T.801 | ISO/IEC 15444-2 or other parts of the ISO/IEC 15444 series.

— If both bit 15 and bit 14 of the **Rsiz** field are 0, the legacy profile encoding of Rec. ITU-T T.800 | ISO/IEC 15444-1 or profile indication by the **PRF** marker are used. If no **PRF** marker is present, bit allocation for profiles within the **Rsiz** field is then according to the following principles:

  — Bits 13 and 12 are both 0 and are reserved for ITU | ISO/IEC use.

  — Bits 11 to 8 signal the family of the profile and broadly classify the use case. In particular, if bits 11 to 0 are all 1, the **PRF** marker is used to indicate profiles.

— Bits 7 to 4 indicate the sublevel within a profile family, or are all 0 in case the profile family does not use sublevels

— Bits 3 to 0 indicate the main level within a profile family.

The **PRF** marker segment, marker ID `0xff56`, is used to signal those profiles that cannot be represented in the **Rsiz** parameter due to lack of signalling bits. That is, any profile whose profile number exceeds 4096 is signalled within the **PRF** marker rather than the **Rsiz** field of the **SIZ** marker segment. The PRF marker segment contains the profile number-4096 as an arbitrary precision integer.

Future extensions deploy the PRF marker segment to indicate conformance to profiles instead of using the legacy protocol of allocating additional bits from **Rsiz**.

7.4 COD marker

The COD marker, marker ID `0xff52`, is an extensible marker with optional syntax elements whose presence is indicated by bit 5 of the **Scod** field. If this bit 5 is 1, an additional 16 bit field **SXcod** field is present which contains signalling of optional features. This extension mechanism is specified in Rec. ITU-T T.801 | ISO/IEC 15444-2.

7.5 File Type Box

The File Type Box, specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, Rec. ITU-T T.801 | ISO/IEC 15444-2, and Rec. ITU-T T.814 | ISO/IEC 15444-15 identifies which codestreams are contained in a file, and which profiles and levels they conform to.

Brands allocated by ISO/IEC 15444 are ‘jp2 ’ for files completely defined by Rec. ITU-T T.800 | ISO/IEC 15444-1, and ‘jpx’ for files completely defined by Rec. ITU-T T.801 | ISO/IEC 15444-2. Brands are not used to determine whether the file can be consumed; the compatibility list specifies the tokens indicating whether the file can be read by a file conformant to one or more standards..

Rec. ITU-T T.800 | ISO/IEC 15444-1 allocates the compatibility list entries as shown in Table 7.

Table 7 — Compatibilites defined within JPEG 2000

Value	Meaning
jp2\040	Compatibility to Rec. ITU-T T.800 | ISO/IEC 15444-1
JP20	Compatibility to Profile 0, defined in Rec. ITU-T T.803 | ISO/IEC 15444-4
JP21	Compatibility to Profile 1, defined in Rec. ITU-T T.803 | ISO/IEC 15444-4
jpx\040	Compatibility to Rec. ITU-T T.801 | ISO/IEC 15444-2
jpm\040	Compatibility to Rec. ITU-T T.805 | ISO/IEC 15444-6
jph\040	Compatibility to Rec. ITU-T T.814 | ISO/IEC 15444-15

7.6 Reader Requirements Box

This box, specified in Rec. ITU-T T.801 | ISO/IEC 15444-2 classifies additional capabilities on a file format level that are either required to fully decode an image, or that are required to display it correctly. The Reader Requirements Box is a particular feature of the JPX file format. It is best to avoid adopting the reader requirements box into other ISO/IEC standards beyond ISO/IEC 15444, yet it is maintained for extensions within Rec. ITU-T T.801 | ISO/IEC 15444-2.

The Reader Requirements Box provides low level information to help the user determine whether it can display the file, and in addition whether it understands every single feature of the standard or extension included within this file. This is specified through two logical masks included within the box (display contents, and fully understand). Both masks are based on the combination of specific low-level feature indicators (e.g, “Animated, and no layer is reused”, “Compositing layer uses CIELab enumerated colourspace with default parameters”).

The **display contents mask** specifies what specific set of features is required to display the contents of the file in a manner acceptable to the data producer. While this seems similar to the compatibility list in the File Type Box, it is not dependent on a specific 4cc-labeled profile for the necessary features being defined.

The **fully understand mask** specifies the complete set of features required to understand the entire file, including the correct interpretation of all metadata within the file. This second expression might be most useful to less capable readers to signal to them that the file contains something they do not understand, and possibly share this information with the end user.

As capabilities are added to the format and to formats derived from JPX, new standard feature and vendor feature codes can be defined. When defining new codes, it is advisable to to ensure that the logical masks do not become ambiguous or contradictory:

All feature codes are defined in such a way that they signal the inclusion of data or a requirement to support a capability; “negative features” (e.g., “the file does not require full ICC profile support”) create ambiguities and make the logical masks overly complex. It is advisable not to use them in the future; in fact, negative features have been phased out of the most recent edition of Rec. ITU-T T.801 | ISO/IEC 15444-2.

Feature codes are designed to be indivisible enabling data consumers to decide whether they can display or fully understand a file given the implementation choices that have been made during its design. Features that group multiple implementation choices together make such decisions complex or impossible. For example, a new feature specifying that the file includes a Rec. ITU-T T.801 | ISO/IEC 15444-2 codestream is undesirable because it is likely that most decoders will only implement specific capabilities of this document, such as the multi-component transform. Features that are subdivided into several features creates a situation where the Reader Requirements Box does not really tell a data consumer, e.g. a reader, which features are needed for successful decoding or full interpretation of the file.

8 Extension mechanisms for the ISO/IEC 21122 series (JPEG XS)

8.1 General

The ISO/IEC 21122 series is based on a codestream and a file format following the principles of Clause 5. As such, they define both a codestream format in ISO/IEC 21122-1, and a file format in ISO/IEC 21122-3. Codestream extensions are signalled by a CAP marker, file format extensions by the File Type Box.

8.2 CAP marker

The CAP marker consists of a bitfield that identifies for each capability whether a capability is required (bit = 1) or not required (bit = 0). Future capabilities extend the bit field at its end. As such, the payload data of the **CAP** marker consists of a variable length bit field starting with capability 0 and extending towards its end. The bit field is has a size of multiples of 8 bits. Any trailing octet (group of 8 bits) of all-zero bits is not written, and therefore not included in the marker. Table 8 lists the syntax of the **CAP** marker.

Table 8 — JPEG XS CAP marker

Syntax	Notes	Size	Values
capabilities_marker() {			
CAP		u(16)	0xff50
Lcap	Size of the capabilities marker in bytes (not including the marker)	u(16)	Variable
`for(i=0;i<(Lcap-2)×8;i=i+1) {`	Loop over capabilities bits		
cap[i]	Requirement of capability i. cap[i] is 1 if capability i is required for decoding a codestream, and 0 otherwise.	u(1)	