TECHNICAL REPORT

ISO/IEC TR 19566-1

First edition
2016-03-15

Information technology — JPEG Systems —

Part 1: Packaging of information using codestreams and file formats

Technologies de l'information — Systèmes JPEG —

Partie 1: Emballage des informations utilisant les flux de codes et les formats de fichier

Reference number
ISO/IEC TR 19566-1:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

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

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).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Introduction

This part of ISO/TR 19566 provides an overview for users of ISO/IEC standards in the use of common JPEG syntax elements at the systems layer for current and future International Standards developed by ISO/IEC.

With the development of coding technologies, ISO/IEC has defined a number of different file formats and multiple variants of codestream syntax. Many of these are specialized to dedicated use cases or compression algorithms. Consequently, it is difficult to maintain an overview about existing file formats, their capabilities and their architectures.

This part of ISO/TR 19566 aims to describe common architectural concepts for file formats and codestream formats. By these means, it lays out guidelines for future file formats and codestream syntax. By observing these guidelines, future International Standards may fit into an overall operable Systems infrastructure that can handle all tools standardized by the ISO/IEC body.

Information technology — JPEG Systems —

Part 1: Packaging of information using codestreams and file formats

1 Scope

This part of ISO/TR 19566 describes common elements of a system layer for JPEG standards, referred to as JPEG Systems.

This part of ISO/TR 19566 describes the common architecture of file formats and codestream formats used in JPEG standards. It is intended that all future Systems components support codestreams and file formats following these guidelines.

2 Terms, definitions, abbreviated terms and symbols

2.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1.1
backward compatibility
inclusive of the old specification within the new specification

Note 1 to entry: Any devices implementing the new standard can also interpret all data compliant with the old version of the standard. However, an old device only compliant with the old version of the standard might not be able to interpret the data compliant with the new version of the standard.

2.1.2
bit stream
partially encoded or decoded sequence of bits comprising an entropy-coded segment

2.1.3
box
structured collection of data describing the image or the image decoding process

Note 1 to entry: See B.2 for the definition of boxes.

2.1.4
box-based file format
file format whose composing elements are well-defined, hierarchically structured boxes

2.1.5
byte
group of 8 bits

2.1.6
coder
embodiment of a coding process

2.1.7
coding
umbrella term that defines both the encoding/compression of a signal as well as the decoding/decompression of a signal

2.1.8
coding model
procedure used to convert input data into symbols to be coded

2.1.9
coding process
process which transforms compressed data into a continuous-tone image and/or a continuous-tone image into its compressed representation

Note 1 to entry: It presents thus an umbrella term for “encoding process” and “decoding process”.

2.1.10
compression
reduction in the number of bits used to represent source image data

2.1.11
component
two-dimensional array of samples having the same designation in the output or display device

Note 1 to entry: An image typically consists of several components, for example, red, green and blue.

2.1.12
continuous-tone image
image whose components have more than one bit per sample

2.1.13
decoder
embodiment of a decoding process

3.1.14
decoding process
process which takes as its input compressed image data and outputs a continuous-tone image

2.1.15
dequantization
inverse procedure to quantization by which the decoder recovers a representation of the DCT coefficients

2.1.16
downsampling
procedure by which the spatial resolution of a component is reduced

2.1.17
encoder
embodiment of an encoding process

2.1.18
encoding process
process which takes as its input a continuous-tone image and outputs compressed image data

2.1.19
entropy-coded (data) segment
independently decodable sequence of entropy encoded bytes of compressed image data

2.1.20
entropy decoder
embodiment of an entropy decoding procedure

2.1.21
entropy decoding
lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder

2.1.22
entropy encoder
embodiment of an entropy encoding procedure

2.1.23
entropy encoding
lossless procedure which converts a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols

2.1.24
forward compatibility
only compliant with the old specification but able to interpret the new specification

Note 1 to entry: Although devices only compliant with the old version of the standard are nevertheless able to interpret the data conforming with the new standard, it is possible that the obtained results are not as good as when using a device compliant with the new version of the standard.

2.1.25
grayscale image
continuous-tone image that has only one component

2.1.26
high dynamic range
image or image data comprised of more than eight bits per sample, coded in floating point representation

2.1.27
intermediate dynamic range
image or image data comprised of more than eight bits per sample

2.1.28
Joint Photographic Experts Group
JPEG
informal name of the committee which created this part of ISO/TR 19566

2.1.29
JPEG standards
collection of ISO/IEC/ITU standards developed by the Joint Photographic Experts Group for still imaging application as listed in the Bibliography

2.1.30
JPEG Systems
common elements of a system layer for JPEG standards

2.1.31
lossless
descriptive term for encoding and decoding processes and procedures in which the output of the decoding procedure(s) is identical to the input to the encoding procedure(s)

2.1.32
lossless coding
mode of operation which refers to any one of the coding processes defined in this part of ISO/TR 19566 in which all of the procedures are lossless

2.1.33
lossy
descriptive term for encoding and decoding processes which are not lossless

2.1.34
low-dynamic range
image or image data comprised of data with no more than eight bits per sample

2.1.35
marker
two-byte code in which the first byte is hexadecimal FF and the second byte is a value between 1 and hexadecimal FE

2.1.36
marker segment
marker together with its associated set of parameters

2.1.37
metadata
additional data associated with the image data beyond the image data

2.1.38
minimum coded unit
MCU
smallest group of data units that is coded

2.1.39
pixel
collection of sample values in the spatial image domain having all the same sample coordinates

EXAMPLE A pixel may consist of three samples describing its red, green and blue value.

2.1.40
point transform
scaling of a sample or DCT coefficient by a factor

2.1.41
precision
number of bits allocated to a particular sample or DCT coefficient

2.1.42
procedure
set of steps which accomplishes one of the tasks which comprise an encoding or decoding process

2.1.43
quantization value
integer value used in the quantization procedure

2.1.44
quantize
act of performing the quantization procedure for a DCT coefficient

2.1.45
sample
one element in the two-dimensional image array which comprises a component

2.1.46
sample grid
common coordinate system for all samples of an image

Note 1 to entry: The samples at the top left edge of the image have the coordinates (0,0), the first coordinate increases towards the right, the second towards the bottom.

2.1.47
table specification data
coded representation from which the tables used in the encoder and decoder are generated and their destinations specified

2.1.48
(uniform) quantization
procedure by which DCT coefficients are linearly scaled in order to achieve compression

2.1.49
upsampling
procedure by which the spatial resolution of a component is increased

2.1.50
vertical sampling factor
relative number of vertical data units of a particular component with respect to the number of vertical data units in the other components in the frame

2.1.51
zero byte
0x00 byte

2.2 Symbols

X	width of the sample grid in positions
Y	height of the sample grid in positions
Nf	number of components in an image
$s_{i,x}$	subsampling factor of component i in horizontal direction
$s_{i,y}$	subsampling factor of component i in vertical direction

2.3 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

API	Application Programming Interface
AR	Augmented Reality
ASCII	American Standard Code for Information Interchange
DCT	Discrete Cosine Transformation
EXIF	Exchangeable Image File Format
HDR	High Dynamic Range
IDR	Intermediate Dynamic Range
JBIG	Joint Bi-level Image experts Group
JFIF	JPEG File Interchange Format
JP2	JPEG 2000 file format
JPEG	Joint Photographic Experts Group

JPSec	JPEG 2000 Secured
JPIP	JPEG 2000 Interactive Protocol
JPWL	JPEG 2000 Wireless
JPX	Extended JPEG 2000 file format
LDR	Low Dynamic Range
LSB	Least Significant Bit
MJ2	Motion JPEG 2000 file format
MJPEG2000	Motion JPEG 2000
MSB	Most Significant Bit
SPIFF	Still Picture Interchange File Format
TIFF	Tagged Image File Format
TMO	Tone Mapping Operator
XML	Extensible Markup Language

3 Conventions

3.1 Operators

NOTE Many of the operators used in this part of ISO/TR 19566 are similar to those used in the C programming language.

3.1.1 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 $-\lfloor (a-1)/2 \rfloor$ 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.

3.1.2 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)

3.1.3 Relational operators

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

3.1.4 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

3.1.5 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.
|x|	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() is the natural logarithm. If x is zero and a is positive, power(x,a) is defined to be zero.

4 General

The purpose of this Clause is to give an informative overview of the elements specified in this part of ISO/TR 19566. Another purpose is to introduce many of the terms which are defined in Clause 2. These terms are printed in *italics* upon first usage in this Clause.

There are three elements specified in this part of ISO/TR 19566:

a) An *encoder* is an embodiment of an *encoding process*. An encoder takes as input *digital source image data* and *encoder specifications*, and by means of a specified set of *procedures* generates as output *a codestream*.

b) A *decoder* is an embodiment of a *decoding process*. A decoder takes as input a codestream, and by means of a specified set of procedures generates as output *digital reconstructed image data*.

c) The *codestream* is a compressed image data representation which includes all necessary data to allow a (full or approximate) reconstruction of the sample values of a digital image. Additional data might be required that define the interpretation of the sample data, such as colour space or the spatial dimensions of the samples.

5 Background

5.1 General

Systems designed to work with still images consist of different functionalities ranging from basic functionalities such as image compression up to image interpretation and search. These functionalities can be grouped into a layered architecture as exemplified in Figure 1.

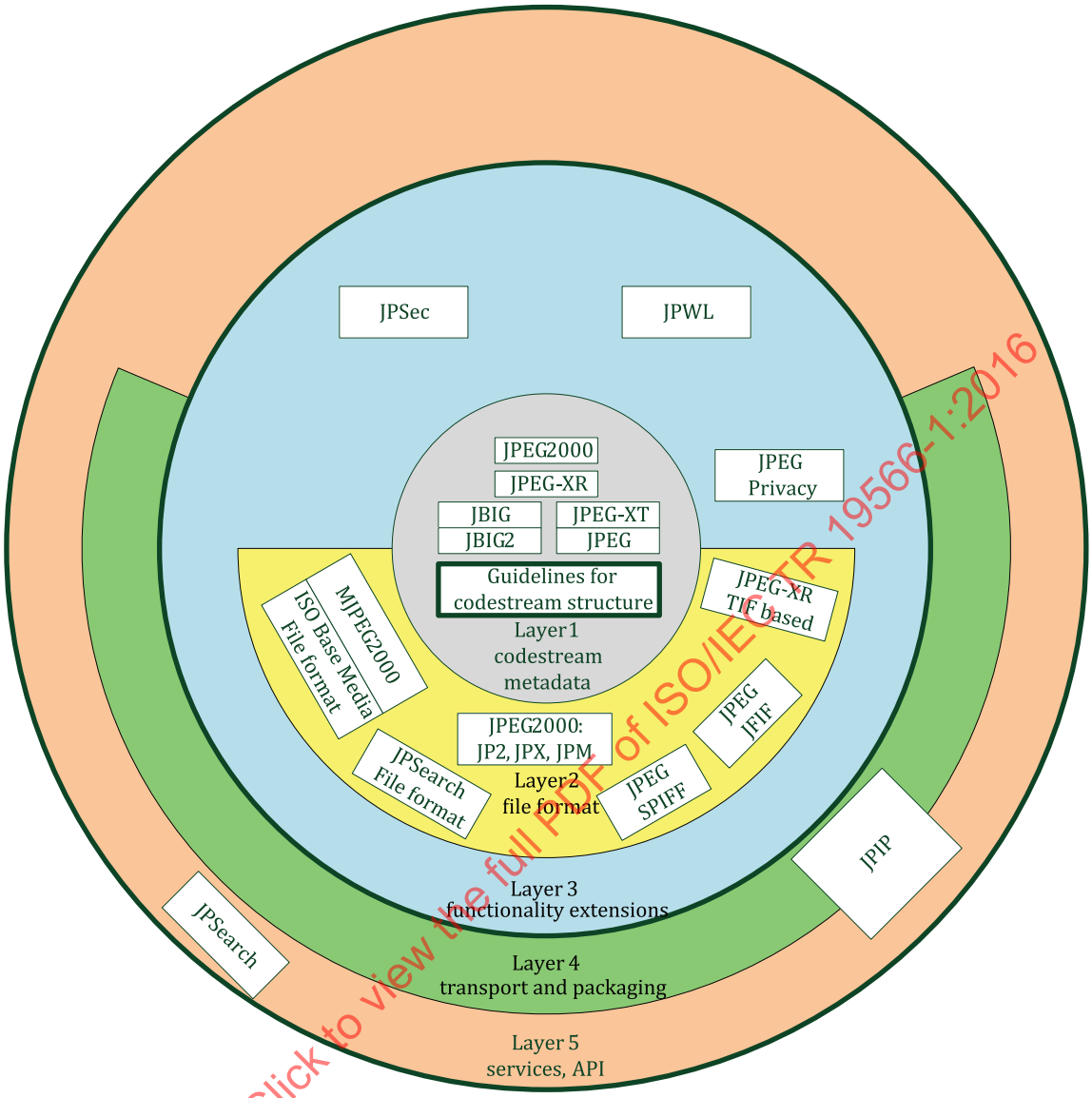


Figure 1 — Layered architecture of the JPEG Systems

Layer 1 is the basic layer and groups different coding algorithms and metadata schemes that allow representing picture data and descriptive information.

Layer 2 groups the different file formats that embed one or multiple codestreams emitted by the image codecs. By these means, it is possible to specify supplemental information, such as the represented colour space, or provide means for compositing multiple images.

Layer 3 defines functionality extensions that enhance the basic functionalities of the lower layers. Corresponding examples are, for instance, privacy protection or error robustness. Such extensions might operate on the codestream layer or the file format layer.

Layer 4 provides means for transport of image data over networks. This comes along with the definition of appropriate ways of dividing codestreams or file formats into units that can be carried with existing network protocols.

Layer 5 finally groups services operating with images, together with associated APIs. Examples are, for instance, facilities for image search.

This part of ISO/IEC 19566-1 is about Layer 1 and Layer 2 of the architecture depicted in Figure 1.

Annex A lays out the general strategies for storage of information.

Annex B explains the preferred syntax for organization of information, while Annex C derives resulting examples and guidelines for the format of a compressed file.

Annex D then explains a more compact form of information storage.

Annex E finally lists a set of functionalities that are typically to be exposed on the file format level.

5.2 Organization of information

In the vast majority of cases, data as considered in JPEG standards can be hierarchically structured. An image, for instance, consists of descriptive data such as its extensions, number of colours, etc. This block of descriptive data is followed by the individual pixels. These pixels can be grouped into tiles, subbands and blocks or some similar structure. In addition, metadata, such as the colour space used to encode the information, might be associated with the image.

Consequently, these data are typically stored in hierarchical data structures having the form of lists, trees, tables or similar concepts. It is the task of every International Standard describing a compression format to define its own syntax in order to map such a hierarchical data structure into a linear sequence of bytes as they can be stored by today's electronic devices. Such an approach permits to optimally adjust the way how the data are coded to the desired needs.

On the other hand, this comes along with the drawback of a multitude of differently looking storage and coding schemes that are difficult to handle in a harmonized manner. Applications that need to decompose a junk of data into its pieces, and supporting different file or codestream formats, need to implement multiple mechanisms for identifying the desired subpieces. Moreover, it is difficult to reuse software and hardware components when implementing a standard because each of them uses its own approach to organize the data. Finally, general operations such as cropping a resolution are only possible when understanding the coded file in all its detail, even when the data will not be decoded by the application.

Consequently, this part of ISO/TR 19566 lays out guidelines for harmonized storage of image and image related data covering layers 1 and 2 of the JPEG Systems.

Annex A
(informative)

Organization of information — Signalling, packaging and storage

A.1 General

This Annex gives an overview to the organization of codestreams and files containing coded images.

A.2 Storage layers

Storage of information takes place on two different layers, namely the codestream layer and the file format layer. The codestream layer comprises all the data that are necessary to transform a coded representation into a sequence of pixels. Corresponding examples include image extensions, number of colour components and the coded data itself. The file format layer provides supplementary information that help to place the data into the correct context, such as colour spaces or composition rules.

Information that is stored on the file format level includes the following:

— colour spaces and profiles (such as ICC) defining the colour space in which the pixels are to be interpreted;

— thumbnail information;

— composition and animation information;

— annotations of regions of interest;

— EXIF data;

— XML data;

— information on resolutions (physical information);

— translation from pixel domain to presentation on physical devices;

— vendor-specific information;

— descriptive data (for search);

— other metadata.

A.3 Interface aspects

File formats and codestream formats offer an interface that can be used by the other layers of the Systems framework shown in Figure 1. This interface covers two aspects:

— The semantics define what functionalities are offered by the codestream or file format, such as the capability to extract a lower resolution.

— The syntax describes how this functionality can be accessed. For instance, it describes the syntax of the box that covers the low resolution representation and how it can be located in the file. A client thus knowing the syntax is hence capable of accessing the corresponding information.

A.4 Interoperability

In order to be able to create an interoperable Systems framework, it is desirable to have a common interface on the file format and codestream levels. Otherwise, every Systems tool, such as error protection, needs to be specifically adapted to every file format and the contained codestream. Ideally, both syntax and semantics are compatible on both the codestream and the file format layer.

Unfortunately, this ideal situation cannot be reached in all cases because of several reasons:

— legacy codecs might not be compliant to the selected interface definition;

— codecs or file formats might be designed in such a way that they cannot offer a certain functionality in order to limit complexity or to be specifically adapted to certain applications.

The Systems framework thus needs to be constructed as shown in Figure A.1. Ideally, both common syntax and semantics are employed. In case file formats or codestreams share the same semantics, but use different syntax, a syntax adaptation needs to take place. This might be by adapting the desired Systems functionality directly to the syntax or semantics, or by first translating the syntax into an alternative format before applying the desired Systems functionality. In case both semantics and syntax differs, only a subset of the JPEG Systems functionality may be provided for the codestream or file format.

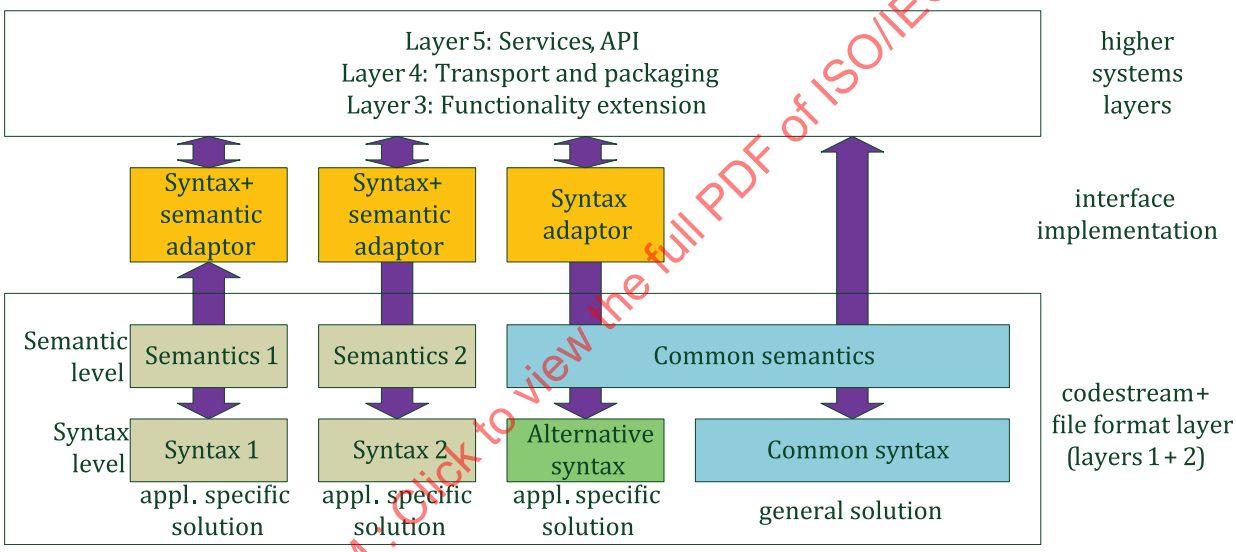


Figure A.1 — File format interface for the Systems framework

A.5 Syntax elements for information signalling, transport and storage

Carriage of information needs to be signalled on both the codestream and the file format level. For both of them, different syntax descriptions have been elaborated in the past.

On the codestream level, APP markers are a simple and lightweight mechanism in order to carry information. During the continued development of the standards, it became, however, obvious that they are rather difficult to extend in a consistent manner.

The box-based structure is a much more generic approach that has its origins in different file formats (i.e. Rec. ITU-T T.800 | ISO/IEC 15444-1). Nevertheless, it can also be used to carry information within codestreams as standardized in ISO/IEC 18477-3. Due to its flexibility, this is the preferred approach for all future coding standards.

Finally, alternative file format syntax (i.e. TIFF-based) have been developed in the past. Nevertheless, box-based file formats are preferred for future developments.

Annex B
(informative)

Concept of boxes

B.1 General

Rec. ITU-T T.800 | ISO/IEC 15444-1 introduced the concept of boxes which is now commonly used in JPEG standards. This Annex is an example of the box structure and how new boxes should be constructed for illustrative purposes.

B.2 Box definition

This subclause is a copy of Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, I.4, in order to understand the fundamental fields of all boxes. Note that future versions of Rec. ITU-T T.800 | ISO/IEC 15444-1 might deviate from the text copied here, but the concepts should still remain.

Physically, each object in the file is encapsulated within a binary structure called a box. That binary structure is as in Figure B.1.

Figure B.1 — Organization of a box

— LBox: Box length. This field specifies the length of the box, stored as a 4-byte big-endian unsigned integer. This value includes all of the fields of the box, including the length and type. As an example, Rec. ITU-T T.800 | ISO/IEC 15444-1 states that "if the value of this field is 1, then the XLBox field shall exist and the value of that field shall be the actual length of the box. If the value of this field is 0, then the length of the box was not known when the LBox field was written. In this case, this box contains all bytes up to the end of the file. If a box of length 0 is contained within another box (its superbox), then the length of that superbox shall also be 0. This means that this box is the last box in the file. The values 2-7 are reserved for ISO use."

— TBox: Box type. This field specifies the type of information found in the DBox field. The value of this field is encoded as a 4-byte big-endian unsigned integer. However, boxes are generally referred to by an ISO/IEC 646 character string translation of the integer value. For all box types defined within this part of ISO/IEC 19566-1, box types will be indicated as both character string (normative) and as 4-byte hexadecimal integers (informative). Also, a space character is shown in the character string translation of the box type as "¥040". All values of TBox not defined within this part of ISO/IEC 19566-1 are reserved for ISO use.

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

— DBox: Box contents. This field contains the actual information contained within this box. The format of the box contents depends on the box type and will be defined individually for each type. Depending on the concrete box type and its specification, it can include other boxes allowing thus the description of hierarchical data structures.

Table B.1 — Binary structure of a box

Field name	Size (bits)	Value
LBox	32	0, 1, or 8 to (2^{32}– 1)
TBox	32	Variable
XLBox	64 0	16 to (2^{64}– 1); if LBox = 1 Not applicable; if LBox ≠ 1
DBox	Variable	Variable

For example, consider the illustration in Figure B.2 of a sequence of boxes, including one box that contains other boxes.

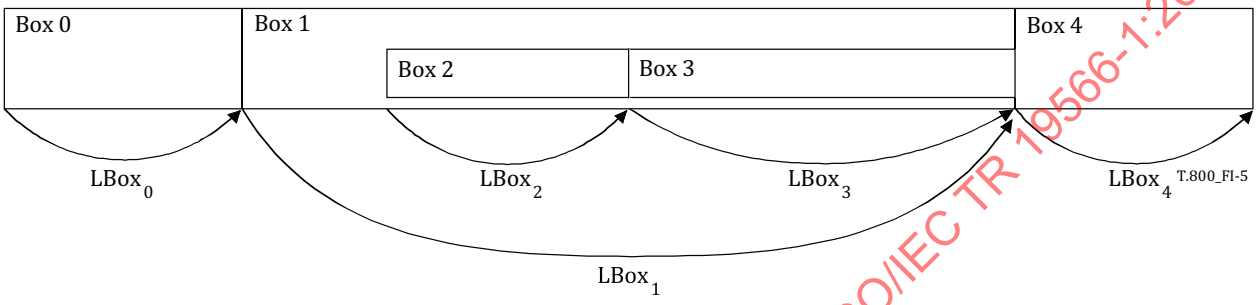


Figure B.2 — Illustration of box lengths

As shown in Figure B.2, the length of each box includes any boxes contained within that box. For example, the length of Box 1 includes the length of Boxes 2 and 3, in addition to the LBox and TBox fields for Box 1 itself. In this case, if the type of Box 1 was not understood by a reader, it would not recognize the existence of Boxes 2 and 3 because they would be completely skipped by jumping the length of Box 1 from the beginning of Box 1.

B.3 Derivation of new box-based file formats

ISO/IEC 15444-12 standardizes means to carry multimedia data. In particular, it lays out the principle structure of boxes and defines multiple box types carrying different kind of information.

Annex C explains how to derive a specification from the concepts provided by ISO/IEC 15444-12. File formats defined within JPEG should follow this information. This includes, in particular, the registration of new box types at http://www.mp4ra.org/atoms.html (ISO/IEC 15444-12, Annex D).

ISO/TR 19566-3 lists all box types used in JPEG standards.

B.4 Carriage mechanisms for box structures within codestreams

Box structures enable the storage of huge amounts of data, possibly hierarchically structured. They provide thus a powerful means for flexible interaction with coded data.

While future standards are hence encouraged to follow this design strategy, early JPEG standards do not follow this approach. Nevertheless, it is recommended to design possible extensions in a box-based approach in order to support a harmonized Systems ecosystem.

An example of how boxes can be carried in a forward compatible way is given in ISO/IEC 18477-3:2015, Annex A. The basic idea is to carry boxes in APP11 markers, as they are ignored by all decoders not being able to interpret these markers. As, however, APP11 markers are limited in size, boxes need to be truncated into junks that can be carried by individual APP11 markers. Moreover, the order of the APP11 is not enforced and hence subject to change.

Given these constraints, the following design rules should be followed.

— In case multiple instances of a certain box type can appear within a file, it should be ensured that the decoder can reliably identify the different box instances. While this is a very general problem (see B.3), cutting boxes into multiple segments and transporting them individually makes this much more difficult because a given box segment might not contain this identifier. Hence, the decoder is not able to identify the box instance to which the box segment belongs to. Consequently, unique box segment identifiers should be supplied as exemplified in ISO/IEC 18477-3:2015, Annex A.

B.5 Placement of boxes

The File Type box should be placed as early as possible in the file (e.g. after any obligatory signature, but before any significant variable-size boxes such as a Movie Box, Media Data Box or Free Space). It identifies which specification is the "best use" of the file, and a minor version of that specification; and also a set of other specifications to which the file complies. Readers implementing this format should attempt to read files that are marked as compatible with any of the specifications that the reader implements. Any incompatible change in a specification should therefore register a new "brand" identifier to identify files conformant to the new specification.

Some boxes should be placed at the appropriate position in the contiguous sequence of boxes so that decoding process will be executed efficiently. There is no mandatory or recommended placement rule for each box in. Instead, for certain boxes, placement rules can be described in the corresponding International Standards. The rules for placement of boxes could provide the following properties:

— uniqueness, all files should contain one and only one box of a given type (e.g. File Type box defined in Rec. ITU-T T.800 | ISO/IEC 15444-1);

— head position, a box should be the first box in the file or superbox (e.g. Image Header box defined in Rec. ITU-T T.800 | ISO/IEC 15444-1);

— medium position, a box should be located anywhere between two boxes (e.g. JP2 Header box defined in Rec. ITU-T T.800 | ISO/IEC 15444-1);

— tail position, a box should be placed last in their superbox (e.g. User Data box defined in ISO/IEC 15444-12);

— neighbouring, a box should be placed immediately after a specific box (e.g. Compound Image Header box defined in Rec. ITU-T T.805 | ISO/IEC 15444-6);

— nesting, a box that itself contains a contiguous sequence of boxes (and only a contiguous sequence of boxes), which is named superbox (e.g. Resolution box defined in Rec. ITU-T T.800 | ISO/IEC 15444-1);

— earliness, a box should be placed as early as possible in the file (e.g. Progressive Download Information box as defined in ISO/IEC 15444-12);

— absolute position, a box should be placed in specific box (e.g. Codestream Registration box as defined in Rec. ITU-T T.801 | ISO/IEC 15444-2).

Note that restricting the placement of boxes limits the efficiency and flexibility of the transport using for instance Rec. ITU-T T.808 | ISO/IEC 15444-9 (see also B.3). Hence, placement constraints should only be specified for well-defined reasons.

Annex C
(informative)

Box-based file format structure

C.1 General

This Annex illustrates how file formats can be built from individual boxes as described in Annex B.

C.2 Boxed-base file format syntax

The binary structure of a file is a contiguous sequence of boxes. Some of those boxes are independent and some of those boxes contain other boxes. The start of the first box should be the first byte of the file and the last byte of the last box should be the last byte of the file. The binary structure of a box is defined in B.2.

C.3 Overview about existing file formats

Several file formats haven been defined in the past relying on the box format as illustrated in Figure C.1:

— the compound image file format in Rec. ITU-T T.805 | ISO/IEC 15444-6;

— the JP2 file format in Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, Annex I;

— the JPX file format extended metadata definition and syntax in Rec. ITU-T T.801 | ISO/IEC 15444-2;

— the Motion JPEG MJ2 file format defined in Rec. ITU-T T.802 | ISO/IEC 15444-3.

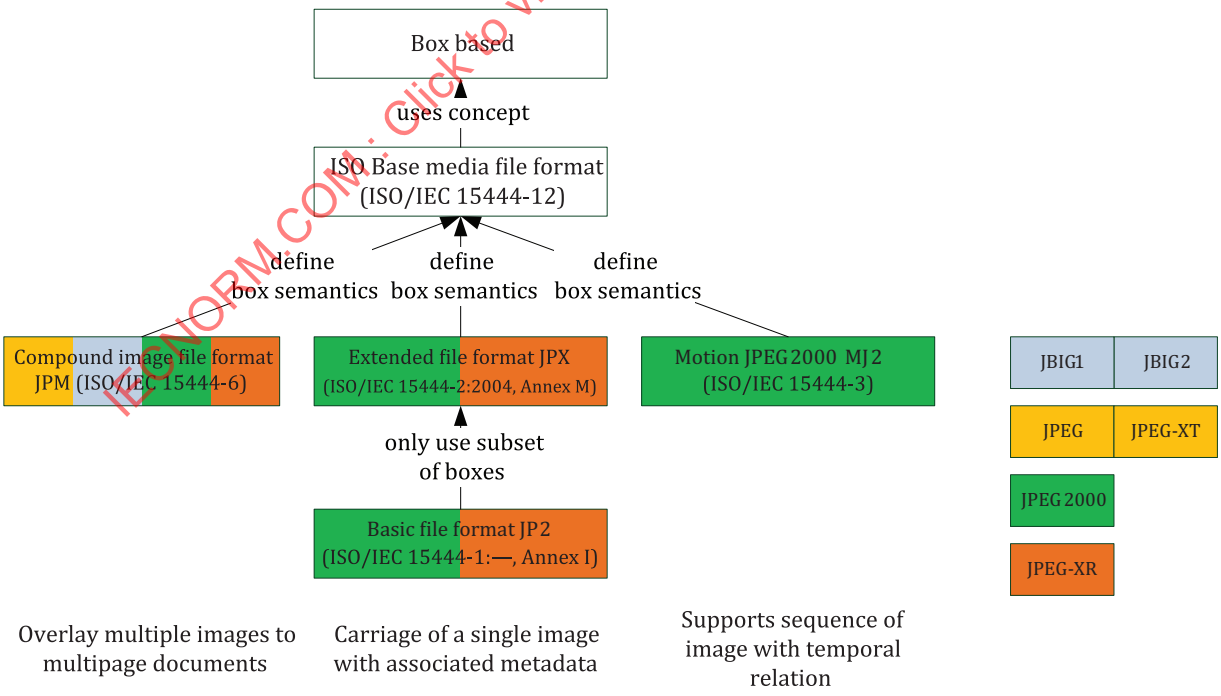
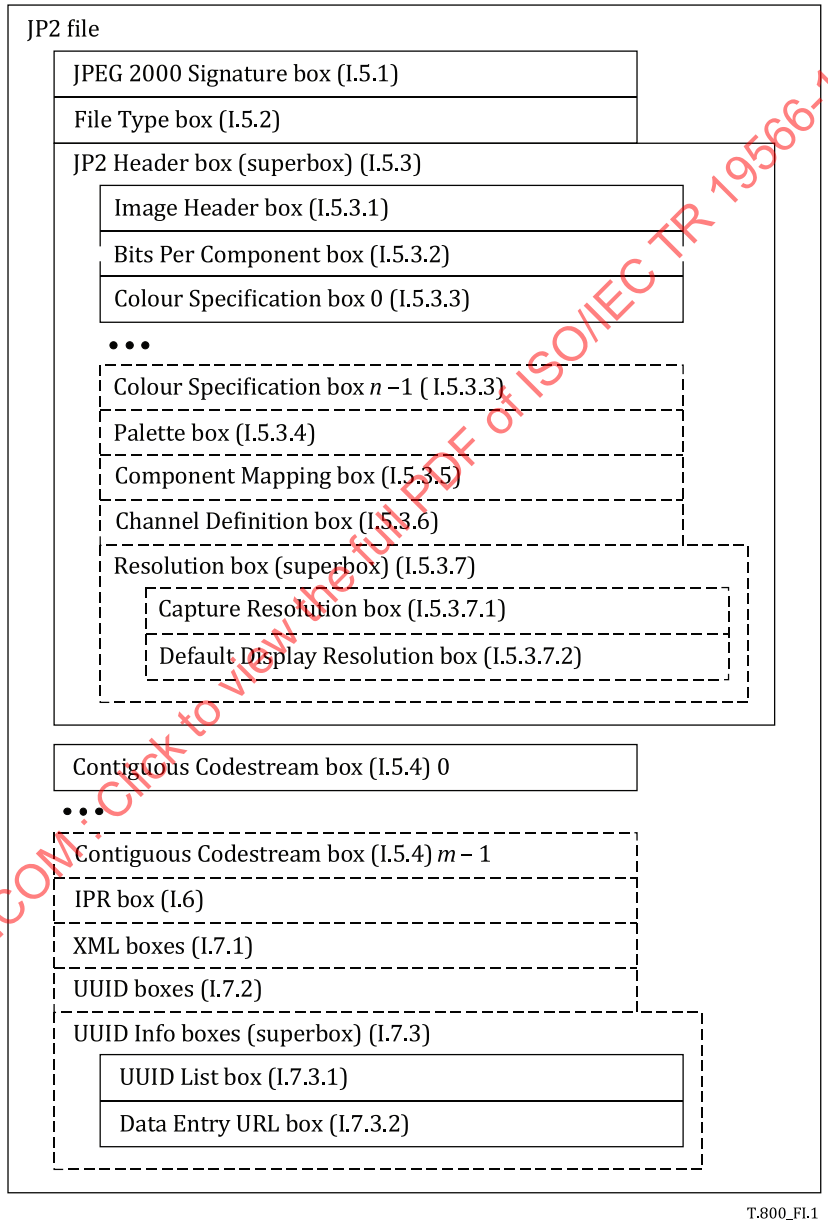


Figure C.1 — Box-based file formats in JPEG Standards

All of them were based on the ISO base media file format and define particular semantics.

C.4 Example file format structure as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1

As an example to understand the preferred file format layout, Figure C.2 shows the structure of a JP2 file. Boxes with dashed borders are optional in conforming JP2 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. If the optional box does not exist, then the mandatory boxes within those boxes shall also not exist.



NOTE References point to Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, Annex I.

Figure C.2 — Conceptual structure of a JP2 file

Figure C.2 specifies only the containment relationship between the boxes in the file. A particular order of those boxes in the file is not generally implied. However, the JPEG 2000 Signature box is the first box

in a JP2 file, the File Type box immediately follows the JPEG 2000 Signature box and the JP2 Header box falls before the Contiguous Codestream box.

The file shown in Figure C.2 is a sequence of boxes. Other boxes may be found between the boxes defined in Rec. ITU-T T.800 | ISO/IEC 15444-1:2004, Annex I. However, all information contained within a JP2 file is in the box format; byte-streams not in the box format are not allowed to be found in the file.

As shown in Figure C.2, a JP2 file contains a JPEG 2000 Signature box, JP2 Header box and one or more Contiguous Codestream boxes. A JP2 file may also contain other boxes as determined by the file writer. For example, a JP2 file may contain several XML boxes (containing metadata) between the JP2 Header box and the first Contiguous Codestream box.

Annex D
(informative)

Codestream syntax

D.1 General

While the box structure defined in Annex B is very powerful and flexible, it comes along with a certain overhead. Furthermore, it is not possible to just insert them into a bit stream of data for special signalling. Solving these aspects is the purpose of the markers.

Markers enable signalling of data that is relevant for decoding of the associated bit stream content. Additionally, markers serve as synchronization points to facilitate features, such as error detection, error correction and random access functionality. Markers that embody extra information are called marker segments. A bit stream annotated with markers and marker segments is called a codestream.

This Annex collects the information from the various JPEG standards in order to explain a unified marker and marker segment syntax that is preferred to be used in future specifications.

This part of ISO/TR 19566 does not include a definition of compliance or conformance. The parameter values of the syntax described in this Annex are not intended to portray the capabilities required to be compliant.

D.2 Markers, marker segments and headers

D.2.1 General

In order to foster the creation of compatible solutions, this part of ISO/TR 19566 explains to codestream and file format implementers the mechanism of markers and marker segments to delimit and signal information contained in a codestream. The general marker syntax is defined in Rec. ITU-T T.81 | ISO/IEC 10918-1. Headers are collections of markers and marker segments.

Every marker is two bytes long. The first byte consists of a single 0xFF byte. The second byte denotes the specific marker code and can have any value in the range 0x01 to 0xFE.

A marker segment includes a marker and associated parameters, called marker segment fields. Every marker segment has a length field that consists of two bytes following the marker. The length field is a big-endian unsigned integer value that denotes the size in bytes of all the marker segment fields (including the two bytes of this length parameter but excluding the two bytes of the marker itself). Following this strict convention allows a decoder to discard unknown marker segments by making use of the length field.

D.2.2 Types of markers and marker segments

Six types of markers and marker segments are used so far: delimiting, fixed information, functional, in bit stream, pointer and informational. Delimiting markers and marker segments are used to frame subparts of the codestream, such as the main and tile-part headers and the bit-stream data of Rec. ITU-T T.800 | ISO/IEC 15444-1. Fixed information marker segments give required information about the image. Functional marker segments are used to describe the coding functions used. In bit stream markers and marker segments are used for error resilience. Pointer marker segments provide specific offsets in the bit stream. Informational marker segments provide ancillary information.

D.2.3 Marker ranges

Different markers are defined in a variety of JPEG standards. The marker range 0xFF30 to 0xFF3F is reserved by Rec. ITU-T T.800 | ISO/IEC 15444-1 without marker segment parameters. Table D.1 shows in which specification these markers and marker segments are defined.

Table D.1 — Marker definitions

Marker code range	Standard definition
0xFF00, 0xFF01, 0xFFFE, 0xFFC0 to 0xFFDF	Defined in Rec. ITU-T T.81 | ISO/IEC 10918-1
0xFFF0 to 0xFFF6	Defined in Rec. ITU-T T.84 | ISO/IEC 10918-3
0xFFF7 to 0xFFF8	Defined in Rec. ITU-T T.87 | ISO/IEC 14495-1
0xFF4F to 0xFF6F, 0xFF90 to 0xFF93	Defined in Rec. ITU-T T.800 | ISO/IEC 15444-1
0xFF30 to 0xFF3F	Reserved for definition as markers only (no marker segments)
	All other values reserved

D.2.4 Marker and marker segment and codestream rules

— Marker segments, and therefore also headers, are a multiple of 8 bits (one byte). Furthermore, the bit stream data between the headers and before the EOC marker or end of file are padded (with zero bits) for 8-bit alignment.

— Delimiting and fixed information marker and marker segments must appear at specific points in the codestream.

— The marker segments should correctly describe the image as represented by the codestream. If truncation, alteration or editing of the codestream has been performed, the marker segments should be updated, if necessary.

— All parameter values in marker segments should be encoded big-endian.

— Marker segments can appear in any order in a given header, unless explicitly defined otherwise.

— All markers with the marker code between 0xFF30 and 0xFF3F have no marker segment parameters.

— Some marker segments have values assigned to groups of bits within a parameter. In some cases, there are bits, denoted by "x", that are not assigned a value for any field within a parameter. The codestream should contain a value of zero for all such bits. The decoder should ignore these bits.

D.2.5 Bit-stuffing and byte-stuffing in content

D.2.5.1 General

As specified, markers can be recognized by the 0xFFxx format. To facilitate synchronization and error correction in codestreams, two mechanisms, bit stuffing and byte stuffing, were designed to prevent sequences of two bytes from occurring in the range of 0xFF80 to 0xFFFF in a codestream. This allows a decoder to search forward for markers that fall within the 0xFF80 to 0xFFFF range.

D.2.5.2 Bit stuffing

Content bits in a codestream are packed into bytes from the MSB to the LSB. When the value of the byte is 0xFF, the next byte includes an extra zero bit stuffed into the MSB. This guarantees that the content of a codestream will never contain a sequence of two bytes in the range of 0xFF80 to 0xFFFF. The last byte in a sequence is padded to the 8-bit boundary with zero bits. Moreover, the last byte in the codestream should not be a 0xFF value (thus the single zero bit stuffed after a byte with 0xFF must be included even if the 0xFF would otherwise have been the last byte).

D.2.5.3 Byte stuffing

Byte stuffing works identical to bit stuffing but uses a full zero byte after every 0xFF byte. This mechanism is only used in Rec. ITU-T T.81 | ISO/IEC 10918-1 (JPEG) and should no longer be used in newer standards because it unnecessarily wastes 7 bits.

D.2.6 Marker syntax (informative)

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

These descriptions are followed by a figure that shows the order and relationship of the parameters in the marker segment. Figure D.1 shows an example of this type of figure. The marker segments are designated by the three-letter code of the marker associated with the marker segment. The parameter symbols have uppercase letter designations followed by the marker's symbol in lowercase letters. A rectangle is used to indicate a parameter's location in the marker segment. The width of the rectangle is proportional to the number of bytes of the parameter. A shaded rectangle (diagonal stripes) indicates that the parameter is of varying size. Two parameters with superscripts and a grey area between them indicate a run of several of these parameters.

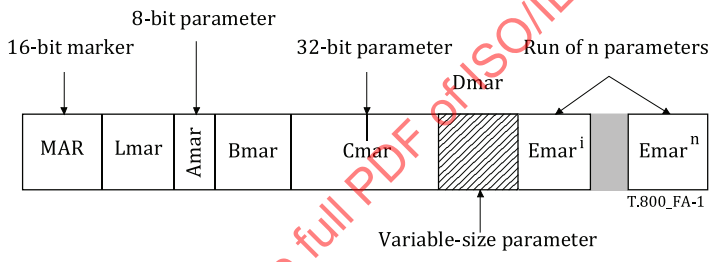


Figure D.1 — Example of the marker segment description figures

The figure is followed by a list that describes the meaning of each parameter in the marker segment. If parameters are repeated, the length and nature of the run of parameters is defined. As an example, in Figure D.1, the first rectangle represents the marker with the symbol MAR. The second rectangle represents the length parameter. Parameters Amar, Bmar, Cmar and Dmar are 8-bit, 16-bit, 32-bit and variable length respectively. The notation $Emar^i$ implies that there are *n* different parameters, $Emar^i$, in a row.

A table is given following the list of parameters that either describes the allowed parameter values or provides references to other tables that describe these values. Tables for individual parameters are provided to describe any parameter without a simple numerical value. In some cases, these parameters are described by a bit value in a bit field. In this case, an "x" is used to denote bits that are not included in the specification of the parameter or subparameter in the corresponding row of the table.

Some marker segment parameters are described using the notation "Sxxx" and "SPxxx" (for a marker symbol, XXX). The Sxxx parameter selects between many possible states of the SPxxx parameter. According to this selection, the SPxxx parameter or parameter list is modified.