INTERNATIONAL STANDARD

ISO/IEC 19566-5

First edition
2019-07

Information technologies — JPEG systems —

Part 5: JPEG universal metadata box format (JUMBF)

Technologies de l'information — Systèmes JPEG —

Partie 5: Format universel de fichier de métadonnées pour JPEG (JUMBF)

Reference number
ISO/IEC 19566-5:2019(E)

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

Published in Switzerland

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.

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 ISO documents 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 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) or the IEC list of patent declarations received (see http://patents.iec.ch).

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.

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 website.

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.

Introduction

The JPEG universal metadata box format (JUMBF) provides a mechanism to embed and refer generic metadata in JPEG files. Specific content types can be assigned to identify the specific type of the embedded metadata. In addition to the content types defined in this document, other types can be defined by other standards or by third parties. ISO/IEC 19566-4 and ISO/IEC 19566-6 both use JUMBF to embed additional metadata in JPEG images. The JPEG XT file format (see ISO/IEC 18477-3) is used to embed JUMBF boxes in JPEG-1 images (see ITU-T.81 | ISO/IEC 10918-1).

Information technologies — JPEG systems —

Part 5:
JPEG universal metadata box format (JUMBF)

1 Scope

This document describes the JPEG universal metadata box format (JUMBF), which provides a universal format to embed any type of metadata in any box-based JPEG file format. This document defines the syntax of the JUMBF box and the mechanism to assign specific content types. In particular, this document specifies XML, JSON, codestream and UUID types. In addition, this document defines the syntax to reference or request the embedded metadata content within or outside the image.

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.

ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 11578, *Information technology — Open Systems Interconnection — Remote Procedure Call (RPC)*

ISO/IEC 21778, *Information technology — The JSON data interchange syntax*

FIPS PUB 180-4[1)], *Secure Hash Standard (SHS)*

W3C, Extensible Markup Language (XML 1.0), 5th edition, <https://www.w3.org/TR/xml>

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

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

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

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

3.1.1
box
binary structure that encapsulates an object embedded in a file

3.1.2
codestream
sequence of bits representing a compressed image and associated metadata

1) http://dx.doi.org/10.6028/NIST.FIPS.180-4

3.1.3
JUMBF content box
content box
box (3.1.1) of any type embedded in a *JUMBF superbox* (3.1.8) except the JUMBF description box

3.1.4
JUMBF content
content
set of all *content boxes* (3.1.3) embedded in a *JUMBF superbox* (3.1.8)

3.1.5
JUMBF content type
content type
specific type of *content* (3.1.4) embedded in a JUMBF box with an associated *JUMBF type* (3.1.6)

3.1.6
JUMBF type
UUID that implies the type of *content* (3.1.4) embedded in a JUMBF box

3.1.7
parent image
main image *content* (3.1.4) of the image file in which the JUMBF box is embedded

3.1.8
superbox
box (3.1.1) that only contains other boxes

3.2 Abbreviated terms

JPEG	Joint Photographic Experts Group
JPEG-1	image complying to ISO/IEC 10918-1
JSON	JavaScript object notation
JUMBF	JPEG universal metadata box format
URI	uniform resource identifier
XML	extensible markup language

4 Conventions

4.1 Conformance language

The keyword "reserved" indicates a provision that is not specified at this time, shall not be used, and may be specified in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be specified in the future.

4.2 Naming conventions for numerical values

Integer numbers are expressed as bit patterns, hexadecimal values, or decimal numbers. Bit patterns and hexadecimal values have both a numerical value and an associated particular length in bits.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation to denote a bit pattern having a length that is an integer multiple of 4. For example, 0x41 represents an eight-bit pattern having only its second most significant bit and its least significant bit equal to 1. Numerical values that are indicated as "binary" are bit pattern values (specified as a string of digits equal to 0, 1 or x in which the left-most bit is considered the most-significant bit and 'x' means

either 0 or 1). Other numerical values not prefixed by "0x" are decimal values. When used in expressions, a hexadecimal value is interpreted as having a value equal to the value of the corresponding bit pattern evaluated as a binary representation of an unsigned integer (i.e., as the value of the number formed by prefixing the bit pattern with a sign bit equal to 0 and interpreting the result as a two's complement representation of an integer value). For example, the hexadecimal value 0xF is equivalent to the 4-bit pattern '1111' and is interpreted in expressions as being equal to the decimal number 15.

4.3 Boxes and superboxes

The annexes of this document focus on the definition of boxes. The details for embedding boxes in specific file formats are defined in the particular documents, for example ISO/IEC 15444-1 for JPEG 2000, ISO/IEC 18477-3 for JPEG-1 / JPEG XT or the more generic ISO/IEC 14496-12 ISO base media file format (ISOBMFF).

In general, each object in the file is encapsulated within a binary structure called a box. A box that only contains other boxes is called a superbox. The binary structure is given in Figure 1.

Figure 1 — Binary structure of a box

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

— **TBox**: box type. This field specifies the type of information found in the Payload Data 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 document, 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 document are reserved for ITU-T | ISO/IEC 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.

— **Payload Data**: 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.

4.4 Graphical descriptions

Box definitions contain graphical description figures to illustrate the structure of the box. These figures should be interpreted as follows.

— The figures do not include box type and size fields.

— A sequence of rectangles is used to indicate the fields of the box and their order.

— The width of the rectangle indicates the length of the field, a square rectangle indicates a 16 bit field.

— A grey background indicates a variable length field.

— Optional fields have a dashed border.

Figure 2 shows an illustrative example of a box with four fields:

— A: 8 bit required field;

— B: 16 bit required field;

— C: variable length required field;

— D: optional 32 bit field.

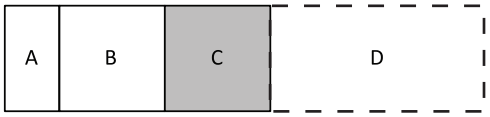

Figure 2 — Box with four fields

5 Organization of the document

JUMBF allows embedding any type of metadata in JPEG images that adopt a box-based file format and enables referencing to the embedded data internally or externally to the image.

The JUMBF Box Format shall be implemented as defined in Annex A. JUMBF Content Types for common metadata such as XML, JSON and image codestreams shall be used in accordance with Annex B. The referencing and requesting mechanism shall be as specified in Annex C. Annex D describes how to embed JUMBF boxes in JPEG-1 images using the JPEG XT Box Format.

Annex A
(normative)

JUMBF box file format

A.1 Overview

This annex defines the JUMBF Box Format. Table A.1 lists all boxes defined in this annex. Indentation within the table indicates the hierarchical containment structure of the boxes.

Table A.1 — Defined boxes

Box name	Type	Superbox	Required?	Comments
JUMBF box	'jumb' (0x6A75 6D62)	Yes	Required	This superbox encapsulates the JUMBF Description box and JUMBF Content Boxes.
JUMBF Description box	'jumd' (0x6A75 6D64)	No	Required	This box is always contained within a JUMBF superbox and specifies the content and behaviour of the JUMBF box.

A.2 JUMBF superbox

A JUMBF box is a superbox that shall contain exactly one JUMBF Description box followed by one or more Content Boxes. The type of the Content Boxes is implied by the JUMBF type field in the JUMBF Description box. JUMBF boxes can be nested, i.e. the Content Boxes may be JUMBF boxes, if implied by the JUMBF type. The JUMBF Description box shall always be the first box in the JUMBF superbox.

The type of a JUMBF box shall be 'jumb' (0x6A75 6D62). The structure of the JUMBF box is illustrated in Figure A.1.

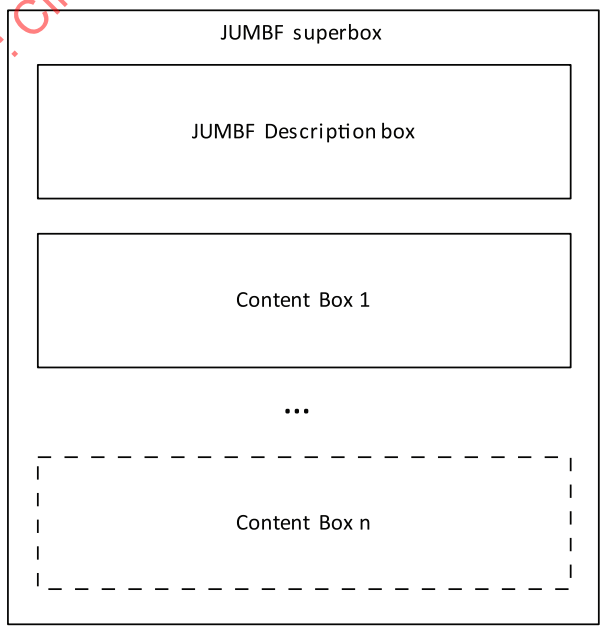

Figure A.1 — Structure of the JUMBF superbox

A.3 JUMBF Description box

The JUMBF Description box provides additional information about the behaviour and content of the parent JUMBF box.

The type of a JUMBF Description box shall be 'jumd' (0x6A75 6D64). The contents of the JUMBF Description box is illustrated in Figure A.2, the field values are summarized in Table A.3.

Figure A.2 — Structure of the JUMBF Description box

— **TYPE**: this field shall contain a 16-byte UUID as specified in ISO/IEC 11578. The value of this UUID specifies the format of the Content Boxes embedded in the JUMBF box and the interpretation of that information. This field is referred to as the JUMBF type.

— **TOGGLES (T)**: this 1-byte field signals the values of options related to the JUMBF box as specified in Table A.2 where bit value 1 means enabled and bit 0 disabled.

Table A.2 — TOGGLES

Binary value	Meaning	TOGGLE Details
0000 xx11	**Requestable**	**Requestable.** This option signals whether the content of the JUMBF box can be requested using the request mechanism as specified in Annex C. If Requestable is set to 1, the "label present" option should be set to 1 as well. * If set to zero, any implementation i) ***shall*** provide an error message, and ii) ***shall not*** act upon the request in a way that can possibly expose the data. * If set to 1, i) a supporting implementation will return the requested data, and ii) a non-supporting implementation shall return an error message.
0000 xxx0	**Not requestable**	
0000 xx1x	**Label present**	**Label present.** This option signals if a Label field is present.
0000 xx0x	**No label present**	
0000 x1xx	**ID present**	**ID present.** This option signals if an ID field is present.
0000 x0xx	**No ID present**	
0000 1xxx	**Signature present**	**Signature present**. This option signals if a Signature field is present.
0000 0xxx	**No signature present**	
All other values are reserved for future use.		

— **LABEL**: this optional field contains a variable length textual label that can be used to reference or request the content of the JUMBF box as specified in Annex C. Its presence is signalled in the TOGGLES field. This value shall be stored as ISO/IEC 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. The Label string shall be null-terminated. The user is responsible for assigning a label which is unique within its scope.

— **ID**: this optional field contains a user assigned unique 4-byte ID that can be used for binary references to this JUMBF box. Its presence is signalled in the TOGGLES field. The user is responsible for assigning an ID that is unique within its scope.

— **SIGNATURE**: this optional field shall contain an SHA-256 (FIPS PUB 180-4) checksum of the JUMBF Content. Its presence is signalled in the TOGGLES field.

Table A.3 — Format of the contents of the JUMBF Description box

Parameter	Size (bits)	Value
TYPE	128	UUID
TOGGLES	8	0-15
LABEL	variable or 0	Null terminated UTF-8 string
ID	32 or 0	User assigned
SIGNATURE	256 or 0	SHA-256 checksum

Annex B
(normative)

JUMBF content types

B.1 Overview

This annex defines JUMBF Content Types that can be used to embed image codestreams, XML data, JSON data or other content wrapped in a UUID box. Table B.1 gives an overview of these Content Types and their assigned JUMBF type UUID values to be used in the JUMBF Description box.

Table B.1 — Overview of JUMBF Content Types

JUMBF type	Meaning
0x6579D6FB-DBA2-446B-B2AC-1B82FEEB89D1	**Codestream Content Type.** The JUMBF box contains exactly one Codestream box.
0x786D6C20-0011-0010-8000-00AA00389B71	**XML Content Type.** The JUMBF box contains exactly one XML box.
0x6A736F6E-0011-0010-8000-00AA00389B71	**JSON Content Type.** The JUMBF box contains exactly one JSON box.
0x75756964-0011-0010-8000-00AA00389B71	**UUID Content Type.** The JUMBF box contains exactly one UUID box.
	Other Content Types may be specified in other specifications. In case the JUMBF Content is exactly one box, it is recommended to specify the JUMBF type UUID by composing the 4-byte type value of the box with the 12-byte ISO reserved value, 0xXXXXXXXX-0011-0010-8000-00AA00389B71 (see ISO/IEC 14496-12). The four-character code replaces XXXXXXXX in the preceding number. In other cases, the user shall use a random generated UUID.

If the last 12 bytes of the JUMBF type are the ISO reserved value 0x0011 0010 8000 00AA 0038 9B71, a decoder can replace the JUMBF box with the single Content Box of the JUMBF box and proceed decoding. If the JUMBF type is not known by the reader, the reader shall ignore the entire JUMBF box.

Table B.2 lists all boxes defined in this annex.

Table B.2 — Boxes defined in this annex

Box name	Type	Superbox	Required?	Comments
Contiguous Codestream box	'jp2c' (0x6A70 3263)	No	Optional	This box allows to embed image codestreams.
XML box	'xml\040' (0x786D 6C20)	No	Optional	This allows to embed XML (W3C, Extensible Markup Language (XML 1.0)) formatted information.
JSON box	'json' (0x6A73 6F6E)	No	Optional	This allows to embed JSON (ISO/IEC 21778) formatted information.
UUID box	'uuid' (0x7575 6964)	No	Optional	This box provides a tool by which vendors can add additional information to a file without risking conflict with other vendors.

B.2 Codestream Content Type

B.2.1 JUMBF box Content

JUMBF boxes that embed image data shall use the 0x6579D6FB-DBA2-446B-B2AC-1B82FEEB89D1 JUMBF type. The Content of the JUMBF box shall contain exactly one Contiguous Codestream box as illustrated in Figure B.1.

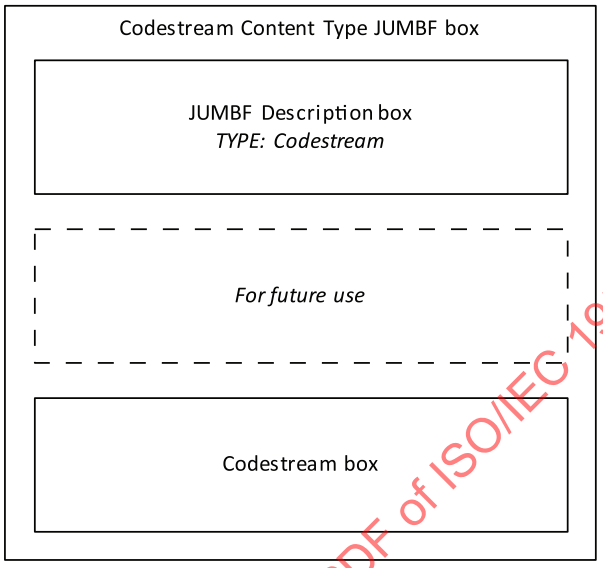

Figure B.1 — Structure of a Codestream Content Type JUMBF box

The Contiguous Codestream box contains a valid and complete image codestream. The image codestream shall be of the same type than the parent image in which the JUMBF box is embedded.

Codestream Content Type JUMBF boxes that do not contain other boxes can also use the 0x6A703263-0011-0010-8000-00AA00389B71 JUMBF type.

B.2.2 Contiguous Codestream box

The type of a Contiguous Codestream box shall be 'jp2c' (0x6A70 3263). The contents of the box shall be as in Figure B.2, the fields are summarized in Table B.3.

NOTE Despite the 'jp2c' type name, a Contiguous Codestream box is not restricted to JPEG 2000 codestreams. The codestream type is determined by the Image Header box if present or the parent image codestream type otherwise.

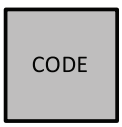

Figure B.2 — Organization of the contents of the Contiguous Codestream box

— **CODE**: this field contains a valid and complete codestream of the same type than the parent image in which the JUMBF box is embedded.

Table B.3 — – Format of the contents of the Contiguous Codestream box

Field name	Size (bits)	Value
Code	Variable	Variable

B.3 XML Content Type

JUMBF boxes that embed XML shall use the 0x786D6C20-0011-0010-8000-00AA00389B71 JUMBF type. The Content of the JUMBF box is exactly one XML box. An XML shall contain XML (W3C, Extensible Markup Language (XML 1.0)) formatted information.

The type of an XML box is 'xml\040' (0x786D 6C20). The contents of the box shall be as in Figure B.3.

Figure B.3 — Organization of the contents of the XML box

— **DATA**: this field shall contain a well-formed XML document as defined by W3C, Extensible Markup Language (XML 1.0).

B.4 JSON Content Type

JUMBF boxes that embed JSON shall use the 0x6A736F6E-0011-0010-8000-00AA00389B71 JUMBF type. The Content of the JUMBF box is exactly one JSON box. A JSON box contains JSON (ISO/IEC 21778) formatted information.

The type of a JSON box is 'json' (0x6A73 6F6E). The contents of the box shall be as in Figure B.4.

Figure B.4 — Organization of the contents of the JSON box

— **DATA**: this field shall contain a well-formed JSON document as defined by ISO/IEC 21778.

B.5 UUID Content Type

JUMBF boxes can embed any type of data using the 0x75756964-0011-0010-8000-00AA00389B71 JUMBF type. The Content of the JUMBF box is exactly one UUID box.

A UUID box contains any information other than the information contained within boxes defined within this document.

The type of a UUID box shall be 'uuid' (0x7575 6964). The contents of the box shall be as in Figure B.5, the field values are summarized in Table B.4.

Figure B.5 — Organization of the contents of the UUID box

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

— **DATA**: this field contains vendor-specific information. The format of this information is defined outside of the scope of this document, but it is indicated by the value of the UUID field.

Table B.4 — Format of the contents of a UUID box

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

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

Annex C
(normative)

JUMBF references and requests

C.1 General

This annex defines the mechanism that can be used to reference or request the content of JUMBF boxes. The references can be binary or use a URI scheme. The URI scheme can be used to make a reference within textual or serialized metadata. References can be made internally within the image and/or externally. To enable references or requests, the JUMBF box shall have a label or ID associated with it, as specified in Annex A.

C.2 URI references

URI references can be used to reference JUMBF boxes from textual metadata such as XML or JSON formatted documents. References can only be valid if a label is assigned to the referenced JUMBF box. The user is responsible for assigning labels which are unique within their scope.

The syntax for references outside the image is as follows:

https://jpeg.org/image.jpg**#jumbf=labelname**

where the URI is exemplary and only the fragment part indicated in bold is normative. The "labelname" argument corresponds with the label embedded in the JUMBF Description box.

If JUMBF boxes are nested, child boxes can be referenced by concatenating their label to the parent label separated by a "/" character. For example:

https://jpeg.org/image.jpg**#jumbf=parentlabel/childlabel**

Metadata schemas which are embedded within the image can reference JUMBF boxes inside the image by using the keyword "self":

self#jumbf=labelname

C.3 Binary references

In addition to the URI references, users can assign a binary 32bit ID that can be used for referencing the JUMBF box in binary formats. The user is responsible for assigning IDs which are unique within their scope.

There is no specific syntax defined for formatting the binary references.

C.4 Requests syntax

Requests can only be made outside the scope of the image if a label is assigned to the requested JUMBF box and if the Requestable TOGGLE as defined in Table A.2 is set to 1. The returned content and associated Media Type is determined JUMBF Content Type. The syntax for JUMBF requests is as follows:

https://jpeg.org/image.jpg**?jumbf=labelname**

where the URI is exemplary and only the request part indicated in bold is normative. The "labelname" argument corresponds with the label embedded in the JUMBF Description box.

If JUMBF boxes are nested, child boxes can be requested by concatenating their label to the parent label separated by a "/" character. For example:

https://jpeg.org/image.jpg**?jumbf=parentlabel/childlabel**

C.5 Requests returned content

C.5.1 General

The following subclauses specify the returned result of requests to JUMBF boxes of Content Types defined in this document.

C.5.2 Codestream Content Type Requests

When a request is made to a Codestream Content Type JUMBF box, the request shall return the content of the Codestream box with the same Media Type as the parent image.

C.5.3 XML Content Type Requests

When a request is made to an XML Content Type JUMBF box, the request shall return the content of the XML box with Media Type application/xml.

C.5.4 JSON Content Type Requests

When a request is made to a JSON Content Type JUMBF box, the request shall return the content of the JSON box with Media Type application/json.

C.5.5 UUID Content Type Requests

When a request is made to a UUID Content Type JUMBF box, the request shall return the entire embedded data of the UUID excluding the UUID. The Media Type shall be defined by the vendor who registered the UUID of the UUID box.

Annex D
(informative)

JUMBF backwards compatibility and integration

D.1 General

This annex provides additional information on the JPEG XT Box File Format (defined in ISO/IEC 18477-3) that is used to embed JUMBF boxes in JPEG-1 images (ITU-T.81 | ISO/IEC 10918-1). This information is for reference only: for full information, see ISO/IEC 18477-3 and ITU-T.81 | ISO/IEC 10918-1.

Unlike other standards, in JPEG XT, boxes are not top-level syntax elements, but are themselves wrapped in JPEG XT (APP11) Marker segments. Since boxes may logically carry more than 64K (65536) bytes of payload data, but marker segments can at most carry 64K of data, a single logical box sometimes needs to be broken up into several marker segments. Syntax elements within the marker segment then instruct the decoder how to put the contents in the marker segment back into a single box.

A JPEG XT file may contain several boxes of the same box type, though with differing content. The syntax of the marker segment provides a mechanism to distinguish between two logically different boxes of the same box type.

Since JPEG XT boxes are represented by APP11 marker segments, non-compliant decoders will ignore embedded boxes.

D.2 JPEG XT Boxes

JPEG XT structures any additional data that remains invisible to legacy decoders in JPEG XT boxes. A box is a generic data container that has both a type, and a body that carries its actual payload. The type is a four-byte identifier that allows decoders to identify its purpose and the structure of its content. A JPEG XT file may also carry several boxes of identical type. These boxes are logically distinct and differ in the value of the Box Instance Number En of the JPEG Extensions marker segment; see Figure D.1.

Boxes are embedded into the codestream format by encapsulating them into one or several JPEG XT marker segments. Since boxes can grow large in size, a single box may extend over multiple JPEG XT marker segments, and decoders sometimes have to merge multiple marker segments before they can attempt to decode the box content. JPEG XT Marker segments that belong to the same logical box and require merging prior to interpretation have identical Box Instance Number fields En, but differ in the Packet Sequence Number Z.

The JPEG XT marker segment consists of the APP11 marker that is reserved for ISO/IEC 18477-3, the size of the marker segment in bytes (not including the marker), a common identifier identical for all boxes and box types, the box instance number field, the packet sequence number field, the box length, the box type and the actual box payload data. The box length field can be extended by a Box Length Extension field that allows box sizes beyond 2^{32}-1 bytes. Figure D.1 depicts the high-level syntax of a JPEG XT Marker segment, Table D.1 summarizes the field values.

0xFFEB	Le	CI Common Identifier	En Box Instance Number	Z Packet Sequence Number	LBOX Box Length	TBOX Box Type	*XLBox* *Box Length Extension (optional)*	Payload Data

Figure D.1 — Organization of the JPEG XT Marker Segment