TECHNICAL REPORT

ISO/IEC TR 19566-2

First edition
2016-10-15

Information technologies — JPEG Systems —

Part 2: Transport mechanisms and packaging

Technologies de l'information — Systèmes JPEG -- —

Partie 2: Mécanismes de transport et paquetage

Reference number
ISO/IEC TR 19566-2:2016(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*.

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

Introduction

Access to images and metadata using channels with limited bit rates can be significantly speed up by allowing proper decoding of partial representations. In this case, parts of the image can be displayed to the user or processed by an algorithm as soon as parts of the codestream or of the file are available at the client side.

Rec. ITU-T T.808 | ISO/IEC 15444-9 standardizes mechanisms called JPIP for incrementally communicating box-structured files, as well as information found within codestreams that may or may not be embedded within boxes of a box-structured file. By these means, standardized methods for accessing meaningful parts of an image are available.

So far, Rec. ITU-T T.808 | ISO/IEC 15444-9 is part of the JPEG 2000 standards. Other standards like JPEG (Rec. ITU-T T.81 | ISO/IEC 10918-1) or JPEG-XR (ISO/IEC 29199) are either not supported at all, or only in a very limited way. Consequently, application of JPIP is predominantly restricted to JPEG 2000. In the ambition to create an ecosystem of tools that can be applied to many or all standards of the JPEG family, this document gives guidelines for design of future compression standards and transmission formats such that partial access to images can be provided in a uniform manner based on the concepts and ideas of JPIP as defined in Rec. ITU-T T.808 | ISO/IEC 15444-9.

Information technologies — JPEG Systems —

Part 2:
Transport mechanisms and packaging

1 Scope

This document collects important information with the goal of elaborating a system layer for JPEG standards, referred to as JPEG systems.

This document summarizes the principles of incremental codestream and file transport that are intended to form the future building blocks JPEG systems. Industrial implementations, future codecs and systems components are encouraged to follow these guidelines.

2 Normative references

There are no normative references in this document.

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:

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

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

3.1.1
bit stream
partially encoded or decoded sequence of bits comprising an entropy-coded segment

3.1.2
box
structured collection of data describing the image or the image decoding process

Note 1 to entry: See ISO/IEC 19566-1 for the definition of boxes.

3.1.3
box-based file format
file format whose composing elements are well-defined, hierarchically structured boxes

3.1.4
byte
group of 8 bits

3.1.5
coder
embodiment of a coding process

3.1.6
codestream
sequence of bits, representing a compressed image and associated metadata

3.1.7
coding
encoding or decoding

3.1.8
coding model
procedure (3.1.27) used to convert input data into symbols to be coded

3.1.9
(coding) process
general term for referring to an *encoding process* (3.1.16), a *decoding process* (3.1.14) or both

3.1.10
compression
reduction in the number of bits used to represent source image data

3.1.11
component
two-dimensional array of *samples* (3.1.28) having the same designation in the output or display device

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

3.1.12
continuous-tone image
image whose *components* (3.1.11) have more than 1 bit per *sample* (3.1.28)

3.1.13
decoder
embodiment of a *decoding process* (3.1.14)

3.1.14
decoding process
process which takes as its input compressed image data and outputs a *continuous-tone image* (3.1.12)

3.1.15
encoder
embodiment of an *encoding process* (3.1.16)

3.1.16
encoding process
process which takes as its input a *continuous-tone image* (3.1.12) and outputs compressed image data

3.1.17
entropy-coded (data) segment
independently decodable sequence of entropy-encoded bytes of compressed image data

3.1.18
entropy decoder
embodiment of an *entropy decoding* (3.1.19) procedure

3.1.19
entropy decoding
lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the *entropy encoder* (3.1.20)

3.1.20
entropy encoder
embodiment of an *entropy encoding* (3.1.21) procedure

3.1.21
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

3.1.22
Joint Photographic Experts Group
JPEG
informal name of the committee that created this document

Note 1 to entry: The "joint" comes from the ITU-T and ISO/IEC collaboration.

3.1.23
JPEG standards
collection of ISO/IEC/ITU standards developed by the *Joint Photographic Experts Group* (3.1.22) for still imaging application as listed in Clause 2

3.1.24
JPEG systems
common elements of a system layer for *JPEG standards* (3.1.23)

3.1.25
metadata
additional data associated with the image data beyond the image data

3.1.26
pixel
collection of *sample* (3.1.28) 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.

3.1.27
procedure
set of steps which accomplishes one of the tasks which comprise an *encoding* (3.1.16) or *decoding process* (3.1.14)

3.1.28
sample
one element in the two-dimensional image array which comprises a *component* (3.1.11)

3.1.29
superbox
box (3.1.2) that itself contains a contiguous sequence of boxes, and only a contiguous sequence of boxes

3.2 Abbreviated terms

ID	identifier
JP2	JPEG 2000 file format
JPEG	Joint Photographic Experts Group
JPIP	JPEG 2000 interactive protocol
JPX	extended JPEG 2000 file format
MJ2	motion JPEG 2000 file format

OSI open systems interconnection

ROI region of interest

XML extensible markup language

4 Conventions

4.1 Operators

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

4.1.1 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 $-\lceil (a-1)/2 \rceil$ and $\lfloor (a-1)/2 \rfloor$ for which y + Na = x with a suitable integer N
umod	x umod a is the unique value y between 0 and a-1 for which y + Na = y with a suitable integer N

4.1.2 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)

4.1.3 Relational operators

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

4.1.4 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

4.1.5 Mathematical functions

⌈x⌉	Ceil of x. Returns the smallest integer that is greater than or equal to x.
⌊x⌋	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.

5 General

The purpose of this clause is to give an informative overview of the elements referred to in this document. Another purpose is to introduce many of the terms which are defined in Clause 3. These terms are printed in *italics* upon first usage in this clause.

There are three elements referred to in this document.

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.

6 Purpose of the document

Rec. ITU-T T.808 | ISO/IEC 15444-9 standardizes methods for partial access to codestream and image files, such that a partial representation of the latter is meaningful and can be decoded by a decoder. This document gives guidelines for design of future technical systems, compression standards and transmission formats such that partial access to images can be provided in a uniform manner based on the concepts and ideas of JPIP as defined in Rec. ITU-T T.808 | ISO/IEC 15444-9. Since not all applications are compatible with the needs of JPIP, these guidelines are informative, although system designers and compression experts are encouraged to follow them whenever possible. The actual standardization of transmission formats and packaging rules is not part of this document and will take place in other documents.

In order to achieve the purpose of the document, Clauses 7 to 11 give an overview about the use cases of JPIP and its system architecture, generic concepts and capabilities. Clauses 12 and 13 then summarize the preconditions for applicability of JPIP and explain concepts for achieving this compatibility.

7 Technical rationale — Partial representation of image and metadata content

Rec. ITU-T T.808 | ISO/IEC 15444-9 (JPIP) has been developed and standardized primarily to overcome the limitations of linear codestreams in addressing applications for the communication of compressed media. Traditionally, compressed content is organized as a linear stream of bytes (the codestream), which may then be wrapped in a file format that provides additional metadata for interpreting and describing the compressed content. ISO/IEC 19566-1 recommends the use of box-based file formats for this purpose.

In some cases, the information in a codestream or the boxes in a file can be reorganized, without altering the meaning of the content. Such reorganization can be useful in communication systems. For example, a JPEG 2000 codestream can be organized in such a way that information appears in order of increasing spatial resolution, increasing quality/precision of the representation, increasing image component indices or increasing location along a raster-like scan. Region-of-interest metadata within a JPX file can be reordered so that items of greater interest appear earlier, again without altering the fundamental content. In bandwidth-constrained environments, such reorganization may allow the recipient of the codestream or complete file to obtain more important information earlier than what might have occurred without the reorganization.

JPIP standardizes mechanisms to provide a much greater level of flexibility in organization than what can be achieved within the constraints of the linear codestream and file format structures. Specifically, JPIP standardizes streaming media types that are capable of communicating the fundamental elements of one or more codestreams and the boxes of a box-structured file, in any order at all. While resequencing the content to match the preferences of a particular client application can be very useful, in practice, the greatest benefits arise from using the JPIP streaming media types to intentionally communicate only a subset of the content. For example, a server might communicate only a subset of the image components, a limited image resolution or a limited spatial region of interest by using the relevant JPIP streaming media type to send selected elements from the compressed codestream. Similarly, a server might communicate only a limited set of region-of-interest descriptions, a limited set of XML boxes or a limited set of compositing instructions from an animation.

In addition to streaming media types, Rec. ITU-T T.808 | ISO/IEC 15444-9 standardizes the elements of a client request syntax that can be used to specify regions, resolutions, components, metadata and higher level semantic entities of interest, in response to which a server can construct the most appropriate response stream. Many aspects of the JPIP request language are substantially generic in the sense that they are meaningful regardless of how the media has been compressed. The streaming media types, however, should be tailored to match the capabilities and fundamental elements of a particular compression standard. Nonetheless, there are many principles and even detailed design patterns from the JPIP standard that can and should be followed in the development of future standards.

Elements of the compressed representation and its containing box-structured file are first mapped to “data-bins”. A JPIP streaming media type then consists of a sequence of “messages”, each of which communicates a range of bytes from a single data-bin. A JPIP message may hold the entire data-bin,

a prefix of a data-bin or indeed any range of bytes from the data-bin. Each message commences with a variable-length "message header" that serves to identify the data-bin, number of bytes and start of the byte range. In certain circumstances, the information in a message header can be encoded relative to that in previous messages, but apart from this, messages are fully self-describing so that they can be arbitrarily rearranged without losing any information. JPIP messages can describe overlapping byte ranges from any given data-bin. These features are valuable in unreliable and/or distributed communication environments. While JPIP messages are usually streamed over a communication channel, they can be stored in files, made available for web-based download, delivered via email, concatenated, selectively discarded and so forth, all of which provide means for communicating partial representations of the original content that may be of interest to a wide variety of applications and consumers.

Although any given sequence of JPIP messages may provide only some of the information from the original content, a key principle in JPIP is that all messages associated with a single "logical target" are consistent. That is, wherever two messages that describe byte ranges from the same data-bin overlap, the overlapping bytes provided by those messages should be identical. This is true even where the messages are sent in different communication sessions, on different days or after receiving different instructions from a network client. It is conceivable and potentially very useful to develop applications in which the content itself is created or compressed dynamically, in response to the needs of client applications, so that some JPIP messages may be delivered to a client before all the content exists. This is perfectly acceptable, so long as such dynamic generation of content does not lead to changes in any data-bin bytes that have already been disseminated via JPIP messages.

8 Use cases and requirements

As introduced above, JPIP is intended to address the needs of applications that can benefit from flexible resequencing or partial communication of the content from a compressed codestream or from a surrounding box-structured file format. The use cases and requirements for JPIP are derived from such applications. The primary use cases are as follows:

a) browsing of large images over a network by an interactive user, where interaction may involve dynamic selection of a region of interest or a spatial resolution of interest;

b) browsing of volumes (e.g. medical volumes) over a network by an interactive user, where interaction may involve a volumetric region of interest;

c) browsing of animated media over a network by an interactive user, where interaction may involve a spatial region of interest, temporal region of interest or a spatial resolution of interest;

d) interactive browsing of large images, volumes or animations together with associated metadata over a network, where interaction may involve specific metadata of interest (e.g. textual labels, XML documents or region-of-interest metadata);

e) making available a portion (e.g. region of interest, resolution of interest and/or metadata of interest) of a large image or other source over a network via a suitably expressed Universal Resource Identifier (URI);

Beyond these primary use cases, some auxiliary use cases that have similar requirements are the following:

a) storing a portion (e.g. region of interest, resolution of interest and/or metadata of interest) of a large image or other source in a file;

b) sending a portion (e.g. region of interest, resolution of interest and/or metadata of interest) or a large image, volume or animated source over a network based on the needs, privileges of capabilities of the receiving entity or system.

All use cases require means for unambiguously communicating partial content from a compressed codestream or a box-structured file format in which codestreams may be embedded. Additionally, the primary use cases listed above require means for specifying the content that is of interest, i.e. a request

language. To make a pre-defined portion of the content available via a URI, the URI itself is required to embed elements of the request language. For the other primary use cases, a mechanism is required to transport user requests to a JPIP capable server. For the auxiliary use cases, no request language is actually required. For use cases that involve requests, the following additional requirements exist.

a) It should be possible to issue a meaningful JPIP request for a region or resolution of interest, without prior knowledge of the compressed content, its original dimensions or any compression parameters. This requirement serves the needs of browsing applications that require high responsiveness since multiple round-trip communications over a network can be avoided. Effectively, this requirement means that JPIP should support high-level requests.

b) JPIP should provide mechanisms that can be used to avoid redundant transmission of content in cases where multiple requests identify overlapping content from a compressed codestream and/or a box-structured file. This requirement is important for any interactive application where multiple requests are expected, especially since it is not generally possible for the requesting user or network client to know how to express requests whose response will be disjoint without having detailed prior knowledge of the compressed structure or file format organization. The next two requirements are largely corollaries of this one, since two requests can only be considered to overlap if they refer to the same source.

c) *Source identification:* JPIP should provide a mechanism to uniquely identify compressed codestreams and/or box-structured files so that multiple requests can be linked together, whether they occur in close proximity or separated by significant time.

d) *Source immutability:* In response to any request, a JPIP server is required only to return elements from an original codestream or from an original box-structured file; it may not modify the elements that are transmitted in a manner that depends upon the request. In some applications, a JPIP server might determine what the "original" codestream or the "original" box-structured file content actually is based on requests that are received, but it should do so in such a way that the content transmitted to all requesting clients over any period of time is indistinguishable from content that might have been transmitted if the original codestream or box-structured file content had been fixed from the beginning.

9 Mapping content to data-bins

JPIP addresses the requirement for communicating partial content from codestreams and box-structured files by mapping elements from these entities to a generic construct known as a *data-bin*. Except as noted below, each data-bin consists of a contiguous range of bytes from the original codestream or box-structured file. In many cases, the bytes of a single data-bin can only be meaningfully decoded or interpreted in sequence, but this is not necessarily the case. JPIP streaming media types are comprised of a sequence of *messages*, where each message carries a single contiguous range of bytes from exactly one data-bin. A message might contain the entire data-bin; it can be valuable, however, for data-bins to be communicated progressively through a sequence of messages. The partition of a source into data-bins is useful if the content of some data-bins can be meaningfully decoded or interpreted even if some other data-bins are not available.

There are two broad types of data-bins: *codestream data-bins* contain elements of a compressed codestream, while *metadata-bins* contain elements from a box-structured file.

9.1 Codestream data-bins

The mapping of codestream elements to codestream data-bins is necessarily specific to the compression scheme to which JPIP is applied. Currently, codestream data-bins have been defined only for JPEG 2000 codestreams, but see Clauses 12 and 13 for recommendation in regard to future standardization activities. In the JPEG 2000 case, the defined data-bins are as follows.

a) *Precinct data-bins* consist of the complete set of bytes formed by concatenating all of the JPEG 2000 codestream packets that are associated with a single JPEG 2000 precinct, in order. Note that JPEG 2000 codestreams support a variety of packet progression orders, only some of which

involve all packets of each precinct appearing consecutively. A server might transcode an initial codestream to one that has such a packet progression order, considering this to be the "original" codestream, from which precinct data-bins are extracted. In practice, the server might not need to actually do this.

b) *Tile header data-bins* consist of all marker segments except start of tile-part (SOT) and progression-order-change (POC) that belong to a single tile in the original codestream. Note that JPEG 2000 tile-part headers might contain marker segments that are of no value in a JPIP context, such as PLT marker segments that are used for random access into a file. The inclusion of such marker segments in a tile header data-bin is allowed, but it makes more sense for a server to transcode content that contains such marker segments to one that does not, considering the transcoded codestream to be the original one. Inclusion of the final start of data (SOD) marker, that delimits tile headers from precinct data in an original JPEG 2000 codestream, is optional.

c) *Main header data-bins* consist of a concatenated list of all markers and marker segments in the main header, starting from the start of codestream (SOC) marker. It contains no SOT, SOD or end of codestream (EOC) markers. Each JPEG 2000 codestream should have exactly one main header data-bin. Note that a JPEG 2000 codestream's main header might contain marker segments that are of no value in a JPIP context, such as TLM marker segments that are used for random access into a file. The inclusion of such marker segments in the main header data-bin is allowed, but it makes more sense for a server to transcode content that contains such marker segments to one that does not, considering the transcoded codestream to be the original one.

d) *Tile data-bins* consist of the string of bytes formed by concatenating all tile-parts belonging to a single tile, in order, complete with their SOT, SOD and all other relevant marker segments.

JPIP defines two different streaming media types: *JPP-stream*, which is precinct oriented using only the first three types of codestream data-bin and *JPT-stream*, which is tile oriented using only the last two types of codestream data-bin.

9.2 Metadata-bins

The mapping of file boxes to *metadata-bins* is designed to be quite generic, with little dependence on the specifics of a particular compression standard or the semantics of its file format boxes. While there is only one type of metadata-bin, it is helpful to think of them in three categories, as follows.

a) A *simple metadata-bin* consists of the contents of a single box from the box-structured file, which is not a superbox. The metadata-bin does not include the header of the box.

b) *A compound metadata-bin* consists of the entire collection of sibling boxes from any level in the original box-structured file. This means that a compound metadata-bin is either the entire original file, or the entire contents of a superbox from the original file. Note that a superbox is a box whose contents consist entirely of a sequence of sub-boxes. Where a compound metadata-bin represents a superbox, it can be understood as the sequence of all top-level sub-boxes from that superbox; it does not include the superbox header. A compound metadata-bin that represents the entire file can similarly be regarded as the sequence of all top-level boxes from the file. It is useful to refer to these as the top-level boxes of the metadata-bin.

c) An *equivalent metadata-bin* is similar to a compound metadata-bin, except that some or all of its top-level boxes have been replaced with the so-called *placeholder boxes*. Placeholder boxes follow the usual box structure, but they embed information that identifies the size and box type of the original replaced top level box, along with JPIP specific information to identify where the contents of the original box can be found. In most cases, this JPIP specific information is the identifier of another metadata-bin (simple, compound or equivalent) that represents the contents of the box. If the box that is replaced represents a codestream (or potentially multiple codestreams), the placeholder box can indicate that the contents of the box are to be found in codestream data-bins, as opposed to a metadata-bin containing the codestream's bytes.

Equivalent metadata-bins allow JPIP streaming media types to partially communicate boxes from a box-structured file, in such a way that the boxes that are communicated can be properly interpreted.

One way to do this would be to recursively replace all top-level boxes in the file and all top-level sub-boxes of each superbox in the file with placeholder boxes, where each placeholder box identifies the metadata-bin that holds the contents of the replaced box. In this way, every super-box and every top-level box in the file that is not a superbox is assigned its own metadata-bin.

The top level of the file is always represented by the *root metadata-bin*, whose identifier is 0. The hierarchical box structure of the file then becomes a hierarchy of placeholder boxes that form links from metadata-bin 0 to each other metadata-bin. In particular, the contents of each box, at any level in the file can be recovered by following the data-bin identifiers found in a sequence of placeholders P_0, P_1, …, P_{K-1}, such that P_k is found in metadata-bin M_k and references metadata-bin M_{k+1}, for each k = 0,1,…, K – 1 and metadata-bin M_K holds the contents of the box of interest. Evidently, it is sufficient for a JPIP server to transmit only metadata-bins M_0 through M_K in order to fully represent the contents and context (i.e. location in the original file) of the original box. In fact, it is sufficient for the JPIP server to transmit only the first part of metadata-bins M_0 through M_{K-1}, so as to include each of the corresponding placeholder boxes P_0 through P_{K-1}.

Some box-structured file formats include boxes that reference locations within other boxes through explicit file addresses; one example of this is the JPX cross-reference (cref) box. The placeholder box approach described above properly preserves the functionality of the original file, even where such explicit file address references are involved. A reader can always navigate the metadata-bin structure, starting from the root metadata-bin and following placeholders, in order to discover the metadata-bin that holds any addressed byte location from the original file. This is because each placeholder box records the size of the original box that it replaces. The JPIP approach thus allows any box-structured file to be partially transmitted, while fully retaining the functionality and interpretation of the parts of the original file that have been transmitted.

Note that JPIP does not prescribe which superboxes and top-level file boxes should be replaced by placeholder boxes. While it is natural to replace all superboxes with placeholders, this could be quite inefficient if the superboxes are very small. In some cases, a placeholder might occupy more space than a very small original box. For this reason, the mapping of superboxes and top-level boxes of the file to superboxes, as well as the assignment of metadata-bin identifiers, is left to the judgement of a JPIP content server. However, the server should not generate two different such mappings for the same source file, without assigning it a fundamentally different source identifier. This follows from Clause 8 requirements c) and d), according to which each source is unambiguously identified and the elements of each identified source are immutable.

As mentioned above, boxes that represent a codestream, or perhaps multiple codestreams, can be replaced by a special type of placeholder box which indicates that the contents of the box have been replaced by codestream data-bins, as opposed to metadata-bins. The JPIP standard specifically defines which boxes in a JP2 file, a JPX file, an MJ2 file and so forth, can be replaced with codestream data-bins. To extend the functionality of JPIP to other box-structured file formats, attention should be paid to identifying which box types in that file format are the ones that might be replaced by codestream data-bins, in addition to identifying how best to map the codestream elements of the relevant compression standard to codestream data-bins.

In the specific case where the source communicated by JPIP has no wrapping box-structured file so that it consists only of a compressed codestream, the JPIP standard indicates that there should still be a root metadata-bin, i.e. the metadata-bin with identifier 0, but that it should be empty.

9.3 Designing box-structured file formats for JPIP

During the design of JPIP, it became apparent that certain design practices for box-structured files are not amenable to efficient communication of partial representations. In particular, the following two practices are problematic:

a) Boxes that contain other boxes, but are not superboxes should be avoided. Recall that a superbox is a box whose contents are nothing other than a sequence of sub-boxes. The sub-boxes within a superbox can be replaced by placeholders. This allows only a subset of the sub-boxes to be communicated via metadata-bins. On the other hand, a problematic design practice is one in which

a box's contents involve a pre-amble (not a sub-box) which is followed by a sequence of sub-boxes. These sub-boxes cannot be replaced by placeholders because their container is not a superbox. The use of box-specific pre-ambles followed by sub-boxes renders the entire structure opaque to JPIP, or indeed any generic box parsing and mapping machinery.

b) File format designs that encourage the generation of long lists of sibling boxes, or provide no way to avoid this, are also to be discouraged.

EXAMPLE 1 This issue was encountered with JPX files, which were designed to accommodate long animations, including video, but initially provided no other way of doing this than by placing the boxes associated with each individual animation frame at the top level of the file. In this case, the root metadata-bin becomes very large even if every top-level box is replaced with a placeholder. For example, an interactive user whose first frame of interest is N, needs to receive and parse at least N placeholder boxes from the top-level of the file in order to discover the metadata-bins or codestream data-bins that contain information of relevance. If N = 1 000 000, this represents a huge communication and computational overhead. To resolve this issue, the JPX file format was extended to include codestream extension boxes and compositing layer extension boxes, which allow a largely flat file structure to be transformed into a hierarchical one.

EXAMPLE 2 A similar problem can occur in the representation of long lists of metadata items, e.g. a long list of imagery associated labels or a long list of small XML documents, where each item is assigned to a box in the file and these boxes naturally appear consecutively. To address such metadata structures while accommodating efficient partial communication via JPIP, JPX has been extended by the grouping box, which allows for hierarchical reorganization of the boxes in a file while avoiding any semantic implications to the structure. The difference between a JPX grouping box and a JPX association (asoc) box is that structuring boxes hierarchically within association boxes has semantic implications.

In the design of box-based file formats for future compression standards, it is important to be able to identify suitable box types for replacement with equivalent metadata-bin placeholders that serve to identify the presence of codestream data-bins.

10 Typical JPIP architecture

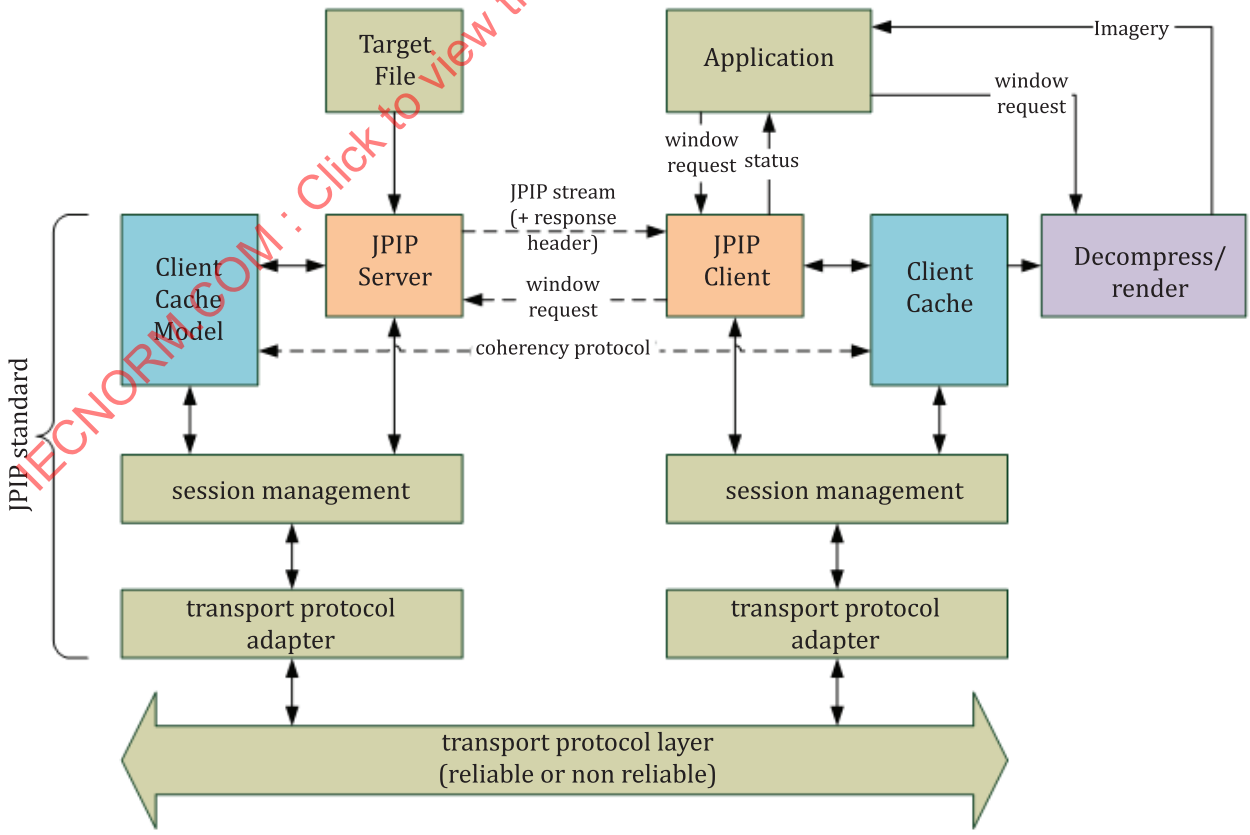

Figure 1 — Overview block diagram

Figure 1 illustrates the typical structure of a system using the JPIP protocol. A JPIP client communicates to a JPIP server and formulates its desire to access a certain image region (called window) at a given resolution or to access certain metadata. The JPIP server disposes of one or several files. Moreover, the JPIP server is able to transcode these files, in order to bring the data into a preferable order if needed. A JPIP server might even be able to convert the file into a different format, such as JPEG, while the original file has been a JPEG 2000 file.

It is the responsibility of the server to decide how the request of the client can be served best. To this end, the server is allowed to modify the request parameters in an appropriate way, and returns them back to the client together with the necessary data (see Clause 11).

Communication and decompression takes place independently from each other. By that way, the client can display a successively refined image to the user, even when the server did not already respond to a certain request. In order to avoid redundant transmission, a client disposes of a cache with limited size. This cache can be reused between sessions and can also be sent to other clients. The server can build a model of the client's cache in order to predict the contained content. By these means, the server can avoid sending redundant data while the client can formulate simple requests, including already partly covered image regions. A cache protocol ensures cache coherence (for instance, if the client decides to discard some data from the cache or when the communication is stateless). In order to ensure coherence, changes in the image, for instance, by editing, need to be signalled to the client as well.

Multiple JPIP servers can be used for the purpose of load balancing or in order to render the data transport more efficient by means of proxy servers. Session management functionality takes care of the appropriate communication establishment. Communication can be both stateless (for simple servers) and stateful (more complex servers). Stateless communication might increase the communication overhead, since the client needs to inform the server about its cache content on a request base.

Finally, JPIP can be used with different transport protocols that can be both reliable and non-reliable. In the latter case, corresponding handshaking is provided in order to determine missed data packets.

11 JPIP request and response schemes

JPIP employs a client-server model in order to disseminate compressed image data (and metadata) from a server to one or more clients. The communication mechanism provided by JPIP is designed to be independent of the underlying OSI network transport layer making it applicable to most, if not all, conceivable setups where image data is interactively sent over a (bandwidth limited) network. As such, JPIP can also automatically function over any of the other OSI-layers that build on top of the network transport layer, being the session layer, the presentation layer and the application layer. Moreover, the JPIP request and response scheme is also independent of the applied compression technology. As such, it can be made to support many image compression technologies, such as JPEG 2000, JPEG and JPEG XT.

In the communication between a server and a client, three different aspects need to be distinguished:

a) client request;

b) server response headers;

c) server response data.

The request and response headers are closely coupled. In the request headers, the server may indicate that it has actually modified the request. Essentially, the request and response headers form the two-way communication that ensures that client and server have the same understanding about what the requested content actually is. The response data communicates elements from the codestream(s) and/or box-structured file that form the complete answer to the (potentially modified) request. The response data may be appended to the response headers in some cases, while it may alternatively be transported differently. The communication of response data can be pre-empted by new requests, but a complete valid set of response headers should always be sent. The response data are allowed to contain elements that do not belong to the requested content, but the response headers should be matched to the request.

Furthermore, the cache coherency strategy, as previously mentioned, is an important part of the JPIP request and response schemes. It enables the server to skip sending data that the client received through previous requests.

With JPIP, the client typically initiates a request to retrieve image data from a specific server. The server then responds by sending the requested data back to the client. Thus, communication between client and server follows a well-defined request-response scheme. This request-response syntax is neutral to the actual applied technology, yet it was designed to closely match the HTTP protocol syntax. This makes it particularly easy to embed JPIP in a regular HTTP web server and browser context.

Moreover, JPIP's request-response scheme was designed to allow the server to respond freely in any way desired, as long as it informs the client of potential request modifications. In practice, this allows the server to do things like transcoding the compressed image data or deviate from the requested image resolution by the client in order to optimally serve the request.

This clause briefly introduces the most common usage of the JPIP client requests and the associated server responses. For a more in-depth coverage, please refer to Rec. ITU-T T.808 | ISO/IEC 15444-9.

11.1 JPIP client request

JPIP requests contain a sequence of request fields that can be categorized into various types. More specifically, types regarding target identification, session and channel management, view-window requests, metadata retrieval, data limiting requests, server control requests, cache management, data upload and client capability and preference querying. The request fields are to be made in compliance with the applied network transport protocol. For example, with HTTP, the request fields can be specified either as parameters of the GET URL or alternatively as HTTP POST header request fields.

A client always initiates its communication with a server by making a target ID request for a specific image resource, which is typically identified by a path location. The server then responds to the target identification request by assigning a unique target ID to the requested image. After this response is sent, the target ID allows the server to consistently and uniquely identify the requested resource. This is required because it allows the server to transcode or to modify the resource depending on specific characteristics of the client. Moreover, this mechanism allows for server-side version updates occurring during multiple simultaneous communication sessions with two or more clients. For a client, it allows caching of resource data for reuse between various sessions, given that the target ID remains the same. Thus, the target ID represents in fact a logical target of a named resource and it can be shared between multiple clients and/or JPIP sessions. Caching is an important aspect of JPIP as it allows saving significant amounts of information from being sent over a bandwidth-limited link.

All client requests are stateless, unless they include either a channel ID or a request for a new channel. In the case that a channel is established, the channel ID represents the link to the session. Furthermore, the channel ID is also used to associate the logical target ID with the session's cache model(s). New channels may be added to the session, on either the same logical target or on new targets, but all channels in a request should belong to the same session. Multiple simultaneous channels on the same target ID allow for implementing flow control for different tasks, e.g. to prioritize foreground over background requests. Each channel has its individual request queue on the server that may be processed on a "first-in" basis, unless a request ID field (qid) is also specified by the client. The qid field allows enforcing that request-response pairs are processed in order.

With the initial request, the client can also inform the server about its supported media types. This happens via the `image-return-types` request field. This field relies on the media types as defined in RFC 2046 (e.g. image/jp2 and image/jpeg).

A very common and typical request, after successfully establishing a channel to a target image resource is the view-window request, which allows a client to retrieve image data, such as metadata and actual pixels, from a server for a particular resource. The view-window is the interactive region of interest (ROI) within a target image. JPIP defines the region size (rsiz) to be relative to the frame size (fsiz), because a client making its first request will not know the true image size and consequently, the sizes of the available resolution levels. Thus, a client will use an arbitrary resolution, typically related to the