INTERNATIONAL STANDARD

ISO/IEC 15444-9

First edition
2005-12-01

Information technology — JPEG 2000 image coding system: Interactivity tools, APIs and protocols

Technologies de l'information — Système de codage d'image JPEG 2000: Outils d'interactivité, API et protocoles

Reference number
ISO/IEC 15444-9:2005(E)

Documents
Document Sharing Hub
© ISO/IEC 2005

Full document details: ISO_IEC 15444-9_2005

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2005

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 15444-9_2005

CONTENTS

		Page
1	Scope	1
2	Normative references	1
3	Definitions	2
3.1	JPEG 2000 Part 1 definitions	2
3.2	HTTP definitions	2
3.3	JPIP definitions	2
3.4	Symbols	3
4	Abbreviations	5
5	Conventions	5
5.1	ABNF rules	5
5.2	File format ABNF rules	6
5.3	Key to graphical descriptions of boxes (informative)	6
6	General description	7
6.1	JPIP protocol	7
6.2	Purpose	8
7	Conformance	9
Annex A (normative) – The JPP-stream and JPT-stream media types		10
A.1	Introduction	10
A.2	Message header structure	11
A.3	Data-bins	13
A.4	Conventions for parsing and delivery of JPP-streams and JPT-streams (informative)	21
A.5	Conventions for JPP-stream or JPT-stream Interoperability (informative)	21
Annex B (normative) – Sessions, channels, cache model and model-sets		22
B.1	Requests within a session vs stateless requests	22
B.2	Channels and sessions	22
B.3	Cache model management	23
B.4	Interrogation and manipulation of model-sets	23
Annex C (normative) – Client request		24
C.1	Request syntax	24
C.2	Target identification fields	25
C.3	Fields for working with sessions and channels	27
C.4	View-window request fields	28
C.5	Metadata request fields	36
C.6	Data limiting request fields	39
C.7	Server control request fields	39
C.8	Cache management request fields	41
C.9	Upload request parameters	47
C.10	Client capability and preference request fields	47
Annex D (normative) – Server response signalling		53
D.1	Reply syntax	53
D.2	JPIP response headers	54
D.3	Response data	59
Annex E (normative) – Uploading images to the server		60
E.1	Introduction	60
E.2	Upload request	60
E.3	Server response	60
E.4	Merging data on the server	61
Annex F (normative) – Using JPIP over HTTP		63
F.1	Introduction	63
F.2	Requests	63
F.3	Session establishment	64

Documents
Document Sharing Hub

© ISO/IEC 2005 – All rights reserved

Full document details: ISO_IEC 15444-9_2005

Page
F.4 Responses ... 64
F.5 Additional HTTP features ... 65
F.6 HTTP and length request field (informative) ... 66
Annex G (normative) – Using JPIP with HTTP requests and TCP returns ... 67
G.1 Introduction ... 67
G.2 Client requests ... 67
G.3 Session establishment ... 67
G.4 Server responses ... 68
G.5 TCP and length request field (informative) ... 68
Annex H (informative) – Using JPIP with alternate transports ... 69
H.1 Introduction ... 69
H.2 Reliable requests with unreliable data ... 69
H.3 Unreliable requests with unreliable data ... 70
H.4 Request and response syntax ... 71
H.5 Session establishment ... 71
Annex I (normative) – Indexing JPEG 2000 files for JPIP ... 72
I.1 Introduction (informative) ... 72
I.2 Identifying the use of JPIP index boxes in the JPEG 2000 file format compatibility list ... 73
I.3 Defined boxes ... 73
I.4 Association of codestream indexes with codestreams ... 81
I.5 Placement restrictions (informative) ... 81
Annex J (normative) – Registration of extensions to this Recommendation | International Standard ... 82
J.1 Introduction to registration ... 82
J.2 Registration elements ... 82
J.3 Registration evaluation criteria ... 82
J.4 Items which can be extended by registration ... 82
J.5 Registration process ... 83
J.6 Timeframes for the registration process ... 83
Annex K (informative) – Application examples ... 84
K.1 Introduction ... 84
K.2 Use of JPIP with codestreams in other file formats ... 84
K.3 Tile-part implementation techniques ... 84
K.4 Precinct-based implementation techniques ... 85
K.5 JPIP protocol transcripts ... 86
K.6 Using JPIP with HTML ... 89
Annex L (informative) – JPIP ABNF collection ... 91
L.1 JPIP Request ABNF ... 91
L.2 JPIP Response BNF ... 98
Annex M (informative) – Patent statements ... 101
Annex N (informative) – Bibliography ... 102

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

FIGURES

Page

Figure 1 – Example of the box description figures 7
Figure 2 – Example of the superbox description figures 7
Figure 3 – JPIP protocol overview 8
Figure 4 – JPIP protocol stack 8
Figure A.1 – Examples of a JPEG 2000 file, JPIP data-bins and JPIP-stream relationships (after G.J. Colyer and R.A. Clark, IEEE Trans. Consumer Electronics, 49 (2003), pp 850–854) 10
Figure A.2 – VBAS structure 11
Figure A.3 – Bin-ID VBAS structure 11
Figure A.4 – Example precinct data-bin 14
Figure A.5 – Metadata-bin example colour scheme 15
Figure A.6 – A sample JP2 file 16
Figure A.7 – A sample JP2 file divided into three metadata-bins 16
Figure A.8 – A superbox with a referenced metadata-bin 17
Figure A.9 – An illegal division of the file into metadata-bins 18
Figure A.10 – Example of the use of stream equivalents 19
Figure A.11 – Placeholder box structure 19
Figure C.1 – Desired region within an image 29
Figure C.2 – Desired region with respect to the subsampled reference grid 29
Figure C.3 – Colourspace specification box selection procedure 50
Figure G.1 – Response data structure on http-tcp connection 68
Figure I.1 – Part of an example JPEG 2000 file containing JPIP index boxes 73
Figure I.2 – Organization of the contents of a Codestream Index box 74
Figure I.3 – Organization of the contents of a Codestream Finder box 75
Figure I.4 – Organization of the contents of a Manifest box 75
Figure I.5 – Organization of the contents of a Fragment Array Index box 76
Figure I.6 – Organization of the contents of a Header Index Table box 77
Figure I.7 – Organization of the contents of a Tile-part Index Table box 78
Figure I.8 – Organization of the contents of a Tile Header Index Table box 78
Figure I.9 – Organization of the contents of a Precinct Packet Index Table box 78
Figure I.10 – Organization of the contents of a Packet Header Index Table box 79
Figure I.11 – Organization of the contents of a File Index box 80
Figure I.12 – Organization of the contents of a File Finder box 80
Figure I.13 – Organization of the contents of a Proxy box 80
Figure I.14 – Organization of the contents of an Index Finder box 81

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

TABLES

Page

Table A.1 – Bin-ID additional VBAS indication 12
Table A.2 – Class identifiers for different data-bin message classes 12
Table A.3 – Legal values for the Flags field of a Placeholder box 20
Table C.1 – Round direction options 31
Table C.2 – Metadata request qualifier flags 39
Table C.3 – Alignment boundaries based on bin type 40
Table C.4 – Legal image return types 40
Table C.5 – Cache descriptor option summary 44
Table C.6 – Legal capabilities of the `processing-capabilities` element 47
Table C.7 – Legal values of the config-capability parameter 48
Table C.8 – View-window handling preferences 49
Table C.9 – Colourspace method client preferences 50
Table C.10 – Placeholder preferences 51
Table C.11 – Codestream sequencing preferences 52
Table D.1 – Legal values of transport-param 55
Table D.2 – Defined reason codes 59
Table I.1 – Defined boxes (Informative) 74
Table I.2 – Container type values 75
Table I.3 – Version values 77
Table K.1 – Example of the use of auxiliary fields in a simple case 85
Table K.2 – Example of the use of auxiliary fields in a more complicated case 85

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 15444-9 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information,* in collaboration with ITU-T. The identical text is published as ITU-T Rec. T.808.

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

— *Part 1: Core coding system*

— *Part 2: Extensions*

— *Part 3: Motion JPEG 2000*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Compound image file format*

— *Part 8: Secure JPEG 2000*

— *Part 9: Interactivity tools, APIs and protocols*

— *Part 11: Wireless JPEG 2000*

— *Part 12: ISO base media file format*

The following parts are under preparation:

— *Part 10: Extensions for three-dimensional data and floating point data*

— *Part 13: An entry level JPEG 2000 encoder*

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

Introduction

ITU-T Rec. T.800 | ISO/IEC 15444-1 (JPEG 2000) is a specification that describes an image compression system that allows great flexibility, not only for the compression of images but also for access into the codestream. The codestream provides a number of mechanisms for locating and extracting portions of the compressed image data for the purpose of retransmission, storage, display, or editing. This access allows storage and retrieval of compressed image data appropriate for a given application without decoding.

The purpose of this Recommendation | International Standard is to provide a network protocol that allows for the interactive and progressive transmission of JPEG 2000 coded data and files from a server to a client. This protocol allows a client to request only the portions of an image (by region, quality or resolution level) that are applicable to the client's needs. The protocol also allows the client to access metadata or other content from the file.

Any organization contemplating the use of this Recommendation | International Standard should carefully consider its applicability.

The International Telecommunication Union (ITU), the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this Recommendation | International Standard may involve the use of a patent.

The ITU, ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the ITU, ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ITU, ISO and IEC. Information may be obtained from the companies listed in Annex M.

Attention is drawn to the possibility that some of the elements of this Recommendation | International Standard may be the subject of patent rights other than those identified in Annex M. ITU, ISO and IEC shall not be held responsible for identifying any or all such patent rights.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: Interactivity tools, APIs and protocols

1 Scope

This Recommendation | International Standard defines, in an extensible manner, syntaxes and methods for the remote interrogation and optional modification of JPEG 2000 codestreams and files in accordance with their definition in the following parts of ISO/IEC 15444:

- ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 and its definition of a JPEG 2000 codestream and JP2 file format.
- the JPEG 2000 family of file formats as defined in further parts of ISO/IEC 15444.

In this Recommendation | International Standard, the defined syntaxes and methods are referred to as the JPEG 2000 Interactive Protocol, "JPIP", and interactive applications using JPIP are referred to as "JPIP systems."

JPIP specifies a protocol consisting of a structured series of interactions between a client and a server by means of which image file metadata, structure and partial or whole image codestreams may be exchanged in a communications efficient manner. This Recommendation | International Standard includes definitions of the semantics and values to be exchanged, and suggests how these may be passed using a variety of existing network transports.

With JPIP, the following tasks may be accomplished in varying, compatible ways:

- the exchange of capabilities;
- the negotiation of capabilities to use in a session;
- the request and transfer of the following elements from a variety of containers, such as JPEG 2000 family files, JPEG 2000 codestreams and other container files:
  - selective data segments;
  - selective and defined structures;
  - parts of an image or its related metadata.

2 Normative references

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

- ITU-T Recommendation T.800 (2002) | ISO/IEC 15444-1:2004, *Information technology – JPEG 2000 image coding system: Core coding system.*
- ITU-T Recommendation T.801 (2002) | ISO/IEC 15444-2:2004, *Information technology – JPEG 2000 image coding system: Extensions.*
- ITU-T Recommendation T.802 (2005) | ISO/IEC 15444-3:2005, *Information technology – JPEG 2000 image coding system: Motion JPEG 2000.*
- ISO/IEC 15444-6:2003, *Information technology – JPEG 2000 image coding system – Part 6: Compound image file format.*
- IETF RFC 768 (1980), *User Datagram Protocol*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc0768.txt>.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

- IETF RFC 793 (1981), *Transmission Control Protocol*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc0793.txt>.
- IETF RFC 2046 (1996), *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2046.txt>.
- IETF RFC 2234 (1997), *Augmented BNF for Syntax Specifications: ABNF*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2234.txt>.
- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2396.txt>.
- IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2616.txt>.

3 Definitions

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

3.1 JPEG 2000 Part 1 definitions

The definitions defined in clause 3 of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 and clause 3 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004 also apply to this Recommendation | International Standard.

3.2 HTTP definitions

The following definitions are intended to match HTTP/1.1. In the case of any difference, these definitions shall be used.

3.2.1 Connection: A transport layer virtual circuit established between two programs for the purpose of communication.

3.2.2 Entity: The information transferred as the payload of a request or response. An entity consists of metainformation in the form of entity-header fields and content in the form of an entity-body.

3.2.3 Proxy: An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers.

3.3 JPIP definitions

The following definitions are used within this Recommendation | International Standard. In some cases, these definitions differ from those used in other standards and/or Recommendations.

3.3.1 cache (client-side): The cache on the Client is the storage of the JPIP data-bins. The Client may have a limited cache and may have to purge cached JPIP data-bins from time to time.

3.3.2 cacheable: A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. Even if a resource is cacheable, there may be additional constraints on whether a cache can use the cached copy for a particular request.

3.3.3 cache-model (server-side): The server's estimation of the portions of the data-bins available in the client's cache. The server may add items to its estimation of the client's cache because it assumes successfully delivery, or because it has received acknowledgements of transmitted data, or because of cache-model update statements.

3.3.4 channel: A mechanism for grouping requests and responses such that only one request/response is active at a time within the group. Multiple simultaneous requests and responses require multiple channels.

3.3.5 client: A program that establishes connections for the purpose of sending requests.

3.3.6 codestream image region: The codestream image region is the intersection between the image and the region defined by the Offset and Region Size. The codestream image region may be empty (no area).

3.3.7 data-bin: A set of bytes of the same type of data which may be partially delivered.

3.3.8 incremental-codestream: The representation of the codestream as a collection of data-bins (main header, tile header, precinct or tile data-bins) having the same codestream identifier.

3.3.9 JPIP index table: A file format box which provides information about the location of portions of a file or codestream.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

3.3.10 logical target: A specific representation of specific original named resource, or a byte range from that specific original named resource, to which the JPIP request is directed. This specific representation might be transcoded from the original named resource.

3.3.11 message: A set of bytes from a single data-bin and the header identifying those bytes and the data-bin.

3.3.12 raw-codestream: The representation of the codestream as a single metadata-bin.

3.3.13 request: A group of fields and values sent from the client to the server to obtain portions of an image or metadata.

3.3.14 resource: A network data object or service that can be identified by a URI. A HTTP target.

3.3.15 response: The bytes sent from the server to the client after receiving a request.

3.3.16 server: An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general.

3.3.17 session: A collection of requests and responses applying to the same resource for which the server maintains a cache model.

3.3.18 session-based: Where the server maintains a cache model.

3.3.19 stateless: A single request where the server does not make use of a cache-model in determining the response.

3.3.20 target: The logical identification of JPIP data. The name of the main target (often the name of a file on the server).

NOTE – JPEG 2000 files or codestreams may be available in multiple representations (e.g., return type, precinct size) or vary in other ways, each identified as a unique logical target.

3.3.21 tile header: All tile-part headers for a specific tile.

3.3.22 view-window: The portion of the image data the client desires, as expressed by the combination of the following fields that appear in the request: Region Size, Offset, Frame Size, Codestream, Codestream Context, Sampling Rate, ROI and Layers. The view-window is often smaller than the whole image data. If a view-window is implied but not specified, then it shall be taken as a view-window on the entire imagery data of the logical target.

3.4 Symbols

For the purposes of this Recommendation | International Standard, the following symbols apply. The symbols defined in clause 4 of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 and clause 4 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004 also apply to this Recommendation | International Standard.

c	An index (starting from 0) of the image component to which the precinct belongs
fx	x-axis frame size for client request view-window
fy	y-axis frame size for client request view-window
fx'	x-axis frame size for suitable codestream resolution
fy'	y-axis frame size for suitable codestream resolution
fx''	Modified jpx x-axis frame size for suitable resolution
fy''	Modified jpx y-axis frame size for suitable resolution
H_{cod}	The codestream height as recorded in the Image Header (ihdr) box (see Annex I.5.3.1 of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004)
H_{comp}	The height of the composited result, supplied in the JPX composition options box (see Annex M.11.10.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
H_{reg}	The height of the compositing layer, as it appears on the compositing layer registration grid
Hs_{inst}	The cropped height
Ht_{inst}	The composited height
l	A unique identifier of the precinct within its codestream

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

N_L	Is the number of decomposition levels
num_components	The number of components encoded
num_tiles	The number of tiles in the codestream
ox	x-axis offset for client request view-window
ox'	x-axis offset for suitable codestream region
ox"	Modified jpx x-axis offset for suitable region
oy	y-axis offset for client request view-window
oy'	y-axis offset for suitable codestream region
oy"	Modified jpx y-axis offset for suitable region
r	Resolution level
s	A sequence number which identifies the precinct within its tile-component
sx	x-axis size of client request view-window
sx'	x-axis size for suitable codestream region
sx"	Modified jpx x-axis size for suitable region
sy	y-axis size of client request view-window
sy'	y-axis size for suitable codestream region
sy"	Modified jpx y-axis size for suitable region
t	An index (starting from 0) of the tile to which the precinct belongs
W_{cod}	The codestream width as recorded in the Image Header (ihdr) box (see Annex I.5.3.1 of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004)
W_{comp}	The width of the composited result, supplied in the JPX composition options box (see Annex M.11.10.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
W_{reg}	The width of the compositing layer, as it appears on the compositing layer registration grid
Ws_{inst}	The cropped width
Wt_{inst}	The composited width
XC_{inst}	The x-axis cropping offset supplied via the relevant instruction (see Annex M.11.10.2.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
XO_{inst}	The x-axis compositing offset, described via the relevant compositing instruction (see Annex M.11.10.2.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
XO_{reg}	The x-axis codestream registration offset
XOsiz	The horizontal offset from the origin of the reference grid of the relevant codestream's SIZ marker segment
XR_{reg}	The x-axis codestream registration sampling factor, described at the beginning of any codestream registration box (see Annex M.11.7.7 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
Xsiz	The width of the reference grid of the relevant codestream's SIZ marker segment
XS_{reg}	The x-axis registration precision described at the beginning of any codestream registration box (see Annex M.11.7.7 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
YC_{inst}	The y-axis cropping offset supplied via the relevant instruction (see Annex M.11.10.2.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

YO_{inst}	The y-axis compositing offset, described via the relevant compositing instruction (see Annex M.11.10.2.1 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
YO_{reg}	The y-axis codestream registration offset
YOsiz	The vertical offset from the origin of the reference grid of the relevant codestream's SIZ marker segment
YR_{reg}	The y-axis codestream registration sampling factor, described at the beginning of any codestream registration box (see Annex M.11.7.7 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)
Ysiz	The height of the reference grid of the relevant codestream's SIZ marker segment
YS_{reg}	The y-axis registration precision described at the beginning of any codestream registration box (see Annex M.11.7.7 of ITU-T Rec. T.801 | ISO/IEC 15444-2:2004)

4 Abbreviations

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

ABNF	Augmented Backus-Naur Form
DICOM	Digital Imaging and Communications in Medicine
DWT	Discrete Wavelet Transformation
EOR	End of Response
HTML	HyperText Markup Language
IP	Internet Protocol
JP3D	JPEG 2000 Part 10: 3-D and floating point data
JPIP	JPEG 2000 Interactive Protocol
JPP	JPIP Precinct
JPSEC	JPEG 2000 Part 8: Secure JPEG 2000
JPT	JPIP Tile-part
JPWL	JPEG 2000 Part 11: Wireless
JTC 1	Joint Technical Committee 1
MTF	Modulation Transfer Function
PDF	Portable Document Format
SC 29	Sub-Committee 29
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UUID	Universal Unique Identifier
VBAS	Variable-length Byte Aligned Segment
WG 1	Working Group 1
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

5 Conventions

5.1 ABNF rules

This Recommendation | International Standard uses the ABNF notation defined in RFC 2234, including the core ABNF syntax rules: ALPHA (letters), CR (carriage return), CRLF (Internet standard newline), CTL (control characters), DIGIT (decimal digits), HEXDIG (hexadecimal digits), LF (line feed), LWSP (linear white space) and SP (space). For the purposes of this Recommendation | International Standard, the following ABNF rules also apply.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

```
NZDIGIT = %x31-39              ; 1-9
UPPER = %x41-5A                ; A-Z
LOWER = %x61-7A                ; a-z
UINT    = 1*DIGIT
NONZERO = *"0" NZDIGIT *DIGIT
UINT-RANGE = UINT ["-" [UINT]]
UFLOAT = 1*DIGIT ["." 1*DIGIT]
ENCODED-CHAR = "%" HEXDIG HEXDIG
UUID = 16(HEXDIG)
TOKEN = 1*(ALPHA / DIGIT / "." / "_")
```

This Recommendation | International Standard also defines `PATH`, representing a file or pathname. In the general case, `PATH` values may contain any character, although for a given server architecture, the server shall reject any characters that are not legal on that particular server. In addition, `PATH` shall be properly encoded as specified by the transport technology.

`UINT-RANGE` specifies a range of integer values. The first integer in the range specifies the beginning of the range. If two values are specified, the first and second values specify the inclusive beginning and ending limits to the range. If only the first value and the `"-"` character are specified, the range includes all values greater than or equal to the first value.

A numerical value immediately preceding an ABNF element refers to a repetition of the parameter that follows the number, for the number of times given by the numerical value, with no intervening spaces between each occurrence.

The construct `"1#"` refers to one or more repetitions of the parameter that follows, each occurrence of which is separated by a comma.

The construct `"1$"` refers to one or more repetitions of the parameter that follows, each occurrence of which is separated by a semicolon.

5.2 File format ABNF rules

```
compatibility-code = 4(ALPHA / DIGIT / "_" / ENCODED-CHAR)
box-type = 4(ALPHA / DIGIT / "_" / ENCODED-CHAR)
box-type-list = "*" / 1#(box-type)
```

`box-type` specifies the four characters of the box type. For each character in the box type, if the character is alpha-numeric (A..Z, a..z or 0..9), the character is written directly into the string. If the character is a space (0x20), then that character shall be encoded as the underscore character ("_"). For any other character, a 3-character string is written in its place, consisting of a percent character ("%") followed by two hexadecimal digits representing the value of the character from the box type in hexadecimal. The `compatibility-code` is encoded the same way that a `box-type` is encoded.

`box-type-list` specifies a list of box types. If the value of a `box-type-list` field is "*", then the field refers to all box types.

5.3 Key to graphical descriptions of boxes (informative)

The description of each box is followed by a figure that shows the order and relationship of the parameters in the box. Figure 1 shows an example of this type of figure. A rectangle is used to indicate the parameters in the box. The width of the rectangle is proportional to the number of bytes in the parameter. A shaded rectangle (diagonal stripes) indicates that the parameter is of varying size. Two parameters with superscripts and a grey area between indicate a run of several of these parameters. A sequence of two groups of multiple parameters with superscripts separated by a grey area indicates a run of that group of parameters (one set of each parameter in the group, followed by the next set of each parameter in the group). Optional parameters or boxes will be shown with a dashed rectangle.

The figure is followed by a list that describes the meaning of each parameter in the box. If parameters are repeated, the length and nature of the run of parameters is defined. As an example, in Figure 1, parameters A, B, C and D are 8, 16, 32 bit and variable length respectively. The notation E^0 and E^{N-1} implies that there are N different parameters, E^i, in a row. The group of parameters F^0 and F^{M-1}, and G^0 and G^{M-1} specify that the box will contain F^0, followed by G^0,

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

followed by F^1 and G^1, continuing to F^{M-1} and G^{M-1} (M instances of each parameter in total). Also, the field D is optional and may not be found in this box.

In addition, in a figure describing the contents of a superbox, an ellipsis (…) will be used to indicate that the contents of the file between two boxes is not specifically defined. Any box (or sequence of boxes), unless otherwise specified by the definition of that box, may be found in place of the ellipsis.

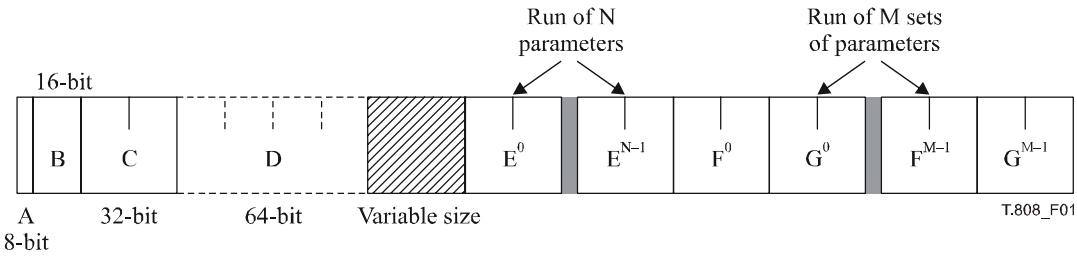

Figure 1 – Example of the box description figures

For example, the superbox shown in Figure 2 must contain an AA box and a BB box, and the BB box must follow the AA box. However, there may be other boxes found between boxes AA and BB. Dealing with unknown boxes is discussed in Annex I.8 of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

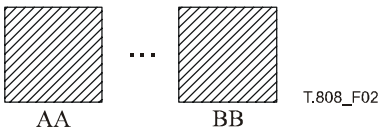

Figure 2 – Example of the superbox description figures

6 General description

6.1 JPIP protocol

This Recommendation | International Standard describes the syntaxes and methods that are used when a client is accessing JPEG 2000 compressed imagery and imagery related data residing on a JPIP-enabled server. This Recommendation | International Standard enables the flexibility and functionality intended in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 to be realized across multiple client/server transports.

JPIP defines the interactive protocol to achieve the efficient exchange of JPEG 2000 imagery and imagery-related data. The protocol defines the Client-Server interactions based on a client request and server response as shown in Figure 3. This Recommendation | International Standard defines the JPIP client requests and the JPIP server responses. HTTP/1.1 (RFC 2616), TCP (RFC 793) and UDP (RFC 768) are shown as examples of possible transports for JPIP. The client uses a View-Window request to define the resolution, size, location, components, layers, and other parameters for the image and imagery related data that is requested by the client. The server responds by delivering imagery and imagery-related data with precinct-based streams, tile-based streams, or whole images. The protocol also allows for the negotiation of client and server capabilities and limitations. The client may request information about an image as defined in JPIP index tables from the server, which enables the client to refine its View-Window request to image specific parameters (e.g., byte range requests). The server's cache model is based on the capabilities defined by the client and the statefulness of the session.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

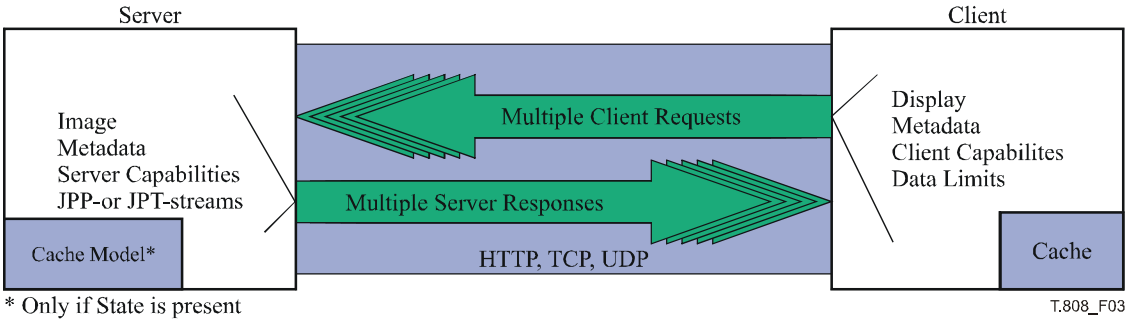

Figure 3 – JPIP protocol overview

This protocol can be used over several different transports as shown in Figure 4. This Recommendation | International Standard includes informative annexes on the use of JPIP protocol over HTTP and TCP, and provides suggestions for other example implementations.

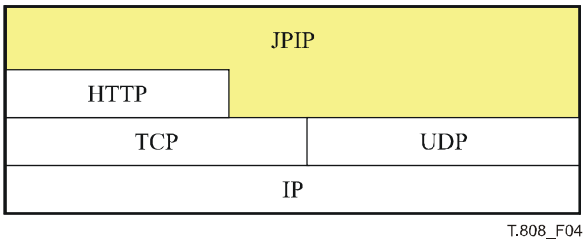

Figure 4 – JPIP protocol stack

Provisions have been included for the extension of the JPIP protocol to support the current JPEG 2000 Standards: ITU-T Rec. T.802 | ISO/IEC 15444-3, Motion JPEG 2000, and ISO/IEC 15444-6, Compound Documents, and the future parts of JPEG 2000 (currently JP3D, JPSEC, and JPWL).

6.2 Purpose

This Recommendation | International Standard defines the syntax and methods required for both the client and server. Each annex defines a component that is required to achieve interoperability and functionality between the client and server over several transports. Each annex may be a requirement of the client, server, or both. The annexes are described below.

- Annex A describes the tile-based and precinct-based streams that are required for both the client and the server. The server is required to produce compliant JPP- and JPT-streams and understand uploaded JPP- and JPT-streams. The client is required to understand and properly decode these streams and is responsible for producing compliant streams when uploading partial imagery to the server.
- Annex B describes the session and cache modelling of a client/server session and is required for both the client and server.
- Annex C defines the client request syntax. The client shall produce compliant requests and the server shall be able to understand and respond to all compliant requests.
- Annex D defines the server response syntax. The server shall produce compliant responses and the client shall be able to understand compliant responses.
- Annex E defines syntax and methods to upload a partial image for systems which use JPIP for upload.
- Annexes F, G and H define the methods and procedures for JPIP client/server interactions over several different transport protocols.
- Annex I defines the indexing information syntax contained in a JPEG 2000 box that can be used by a client and server to more efficiently access imagery and imagery related data.
- Annex J defines how this standard can be extended through registration.
- Annex K describes several examples of using this Recommendation | International Standard for several different applications.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

7 Conformance

Conformance with this Recommendation | International Standard by a client means that the client's JPIP requests are well structured, valid and conformant to the JPIP client requests as defined by this Recommendation | International Standard. Clients shall support all normative requests types.

Conformance with this Recommendation | International Standard by a server means that the server's JPIP responses are well structured, valid and conformant to the JPIP server response signalling as defined by this Recommendation | International Standard. Servers shall support all normative request types.

While it is intended that this Recommendation | International Standard should be implemented such that imagery data is requested on the basis of client-side application requirements in efficient JPIP requests, no conformant behaviour is defined.

Equally, imagery data should be served on the basis of efficient JPIP server responses, minimizing the amount of served data outside the client's signalled interests and redundant data already in possession by the client. However, no conformant behaviour is defined.

It is expected that server applications may reduce efficiency by sending additional data or redundant data depending on the network quality-of-service. Such implementation decisions are application-specific and provide the JPIP system with high utility. However, this Recommendation | International Standard does not define conformance for engineering these implementation decisions.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

Annex A

The JPP-stream and JPT-stream media types

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

A.1 Introduction

JPP-stream and JPT-stream are media types useful for presenting JPEG 2000 codestreams and file format data in an arbitrary order. Each media type consists of a concatenated sequence of messages, where each message contains a portion of a single data-bin preceded by a message header. Data-bins contain portions of a JPEG 2000 compressed image representation, such that it is possible to construct a stream that completely represents the information present in a JPEG 2000 file or codestream. Each message is completely self-describing, so that the sequence of messages may be terminated at any point and messages may be re-ordered subject to minimal constraints without losing their meaning. For these reasons, JPP-stream and JPT-stream media types are useful for JPIP servers and the JPIP protocol is designed with these media types particularly in mind. This annex defines the JPP-stream and JPT-stream media types without reference to the JPIP protocol.

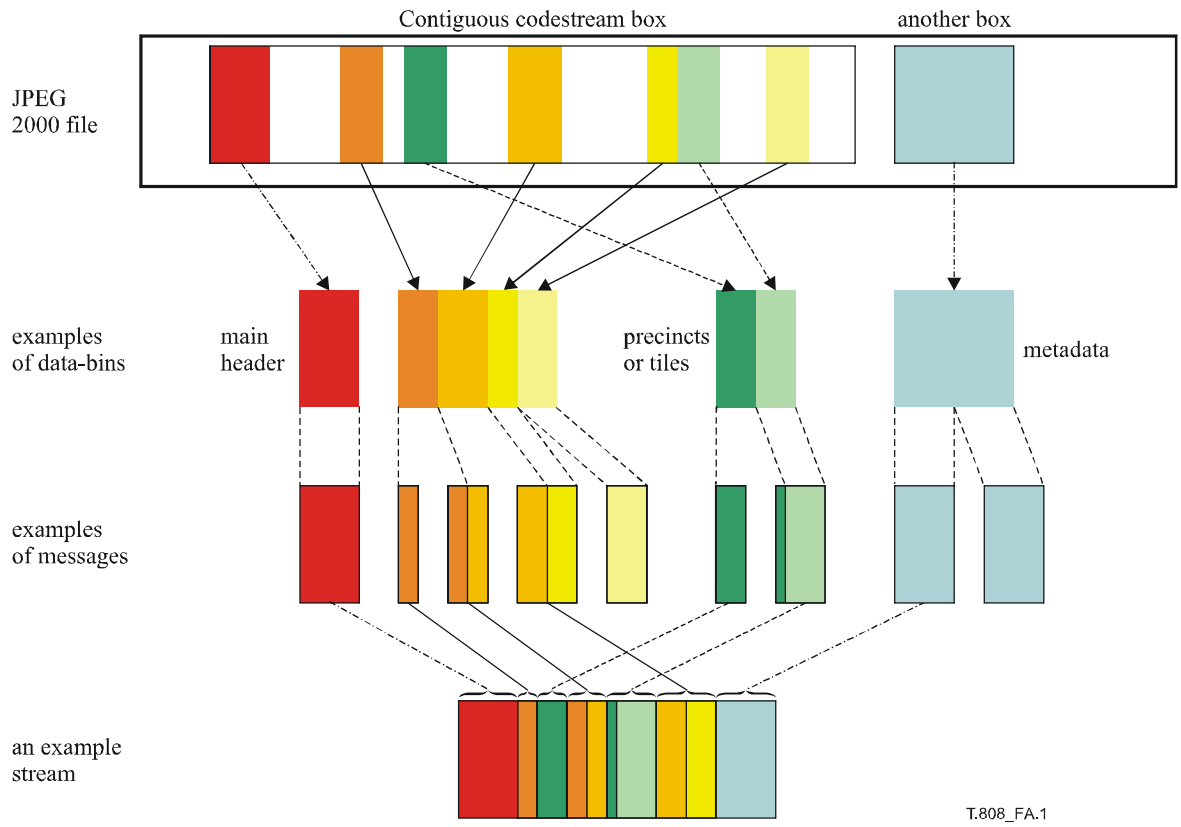

Figure A.1 – Examples of a JPEG 2000 file, JPIP data-bins and JPIP-stream relationships (after G.J. Colyer and R.A. Clark, IEEE Trans. Consumer Electronics, 49 (2003), pp. 850–854)

Figure A.1 is an illustrative example of the relationship between the bit-streams from a JPEG 2000 file, JPIP data-bins, and a JPIP stream. The figure shows the main header colour coded red, 2 precincts with packets coded in shades of orange-yellow and green, and a meta-data box coded blue. Self-describing JPIP messages are formed from these data-bins and concatenated to form a JPIP stream.

A JPIP stream consists of one or more concatenated JPIP messages. Each JPIP message consists of a header and a body. The header provides descriptive information to identify the relevant data-bin. The body is data from that data-bin. Unless further signalling is provided, the message is the concatenation of the header with the body.

NOTE – In this Recommendation | International Standard, all examples provided form binary messages by the concatenation of header and body. It is implementation-specific to the transport and application if other signalling for header and body is provided. For example, auxiliary signalling with variable error protection may be implemented for wireless-based applications.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

A.2 Message header structure

A.2.1 General

Each message represents a portion of exactly one data-bin. The message header consists of a sequence of variable-length byte-aligned segments (VBAS). Each VBAS consists of a sequence of bytes, all but the last of which has a most significant bit (bit 7) of 1, as seen in Figure A.2. The least significant 7 bits of each byte in the VBAS are concatenated to form a bit stream which is used in different ways for different VBASs.

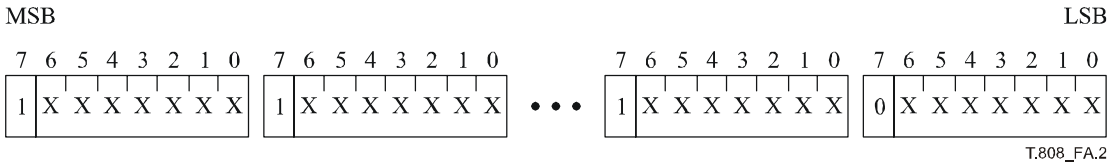

Figure A.2 – VBAS structure

The message header serves to identify the particular data-bin and byte range which is represented by the message body. Message headers can take an independent form and a dependent form. The independent form is a long form where the message headers are completely self-describing; their interpretation is independent of any other message headers. The optional shorter dependent form message headers make use of information in the headers of previous messages; their decoding is dependant on the previous message. Applications may choose to use the long form message headers; these messages can be rearranged in any arbitrary order. Alternatively, applications may use the shorter form message headers that do depend on previous message headers; these are shorter messages but will create erroneous results if the messages are not arranged in the correct sequence when decoded. It is an application decision whether or not the sequence ordering of received messages can be assumed to be reliable and, if so, whether to make use of the shorter form message headers.

The message header consists of the following VBAS's (optional VBAS's identified by the use of square brackets):

Bin-ID [, Class] [, CSn], Msg-Offset, Msg-Length [, Aux]

The existence of the Class and CSn VBASs are determined by examining the Bin-ID VBAS. The existence of the Aux VBAS is determined by the Class VBAS or the previous Class VBAS, if there is no Class VBAS in the current message header.

The Bin-ID VBAS serves several roles. Bits 6 and 5 of the first byte of the Bin-ID VBAS, labelled 'b' in Figure A.3, indicate whether the Class and CSn VBASs are present in the message header. Table A.1 defines the bit values and its meaning.

Bit 4 of the first byte of the Bin-ID VBAS, labelled 'c' in Figure A.3, indicates whether or not this message contains the last byte in the associated data-bin: '0' means it is not the last byte in the data-bin; '1' indicates that it is the last byte in the data-bin. Receiving a message with this bit set allows determination of the length of the complete data-bin, although it does not imply that the complete JPP-stream or JPT-stream contains sufficient messages to assemble all of the bytes from that data-bin.

The remaining 4 bits of the first byte and the 7 low order bits of any remaining bytes in the Bin-ID VBAS (labelled 'd' in Figure A.3) form an "in-class identifier", which is used to uniquely identify the data-bin within its class, in the manner described in A.2.3.

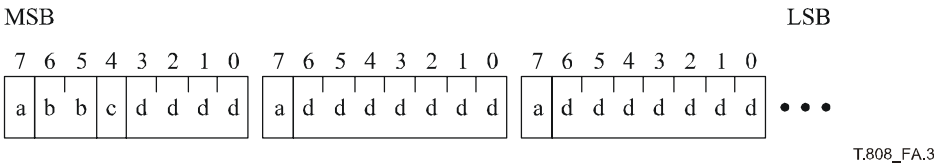

Figure A.3 – Bin-ID VBAS structure

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

Table A.1 – Bin-ID additional VBAS indication

Indicator Bits 'bb'	Meaning
00	Prohibited.
01	No Class or CSn VBAS is present in message header
10	Class VBAS is present but CSn is not present in message header
11	Class and CSn VBAS are both present in the message header.

The Class VBAS, if present, provides a message class identifier. The message class identifier is a non-negative integer, formed by concatenating the least significant 7 bits of each byte of the VBAS in big-endian order. If the Class VBAS is not present, the message class identifier is unchanged from that associated with the previous message. If the Class VBAS is not present and there is no previous message, the message class identifier is 0. Valid message class identifiers are described in A.2.2.

The CSn VBAS, if present, identifies the index (starting from 0) of the codestream to which the data-bin belongs. The codestream index is formed by concatenating the least significant 7 bits of each byte of the VBAS in big-endian order. If the CSn VBAS is not present, the codestream index is unchanged from the previous message. If CSn VBAS is not present and there is no previous message, the codestream index is 0.

The Msg-Offset and Msg-Length VBAS's each represent non-negative integer values, formed by concatenating the least significant 7 bits of each byte in the VBAS in big-endian order. The Msg-Offset integer identifies the offset of the data in this message from the start of the data-bin. The Msg-Length integer identifies the total number of bytes in the body of the message.

An Aux VBAS may be present. Its presence, and meaning if present, is determined by the message class identifier found within the Bin-ID VBAS, as explained in A.2.2. If present, the Aux VBAS represents a non-negative integer value, formed by concatenating the least significant 7 bits of each byte in the VBAS in big-endian order.

NOTE – The information in the Aux VBAS cannot affect the length of the message body.

A.2.2 Message class identifiers

The message class identifiers defined by this Recommendation | International Standard are the non-negative integers shown in Table A.2. The interpretation of the data-bin classes to which they refer is described in A.3. All other values of message class identifier are reserved, and the associated messages should be skipped by decoders not recognizing the value.

Class identifiers are chosen such that an Aux VBAS is present if and only if the identifier is odd. This property allows unrecognized message headers to be correctly parsed and the contents skipped.

Extended precinct data-bin messages have exactly the same interpretation as non-extended precinct data-bin messages and they refer to exactly the same precinct data-bins. The extended precinct messages include an Aux VBAS which identifies the number of complete packets (quality layers) which would be available for the precinct if the bytes in this message were combined with all previous bytes of the same precinct. If this message also contains the last byte of the data-bin, the Aux VBAS indicates the total number of quality layers associated with the precinct in the original codestream. Otherwise, the Aux VBAS indicates the quality layer to which the byte immediately following the last byte in the message belongs. The information in the Aux VBAS may be useful to certain clients.

Table A.2 – Class identifiers for different data-bin message classes

Class identifier	Message class	Data-bin class	Stream type
0	Precinct data-bin message	Precinct data-bin	JPP-stream only
1	Extended precinct data-bin message	Precinct data-bin	JPP-stream only
2	Tile header data-bin message	Tile header data-bin	JPP-stream only
4	Tile data-bin message	Tile data-bin	JPT-stream only
5	Extended tile data-bin message	Tile data-bin	JPT-stream only
6	Main header data-bin message	Main header data-bin	JPP- and JPT-stream
8	Metadata-bin message	Metadata-bin	JPP- and JPT-stream

Extended tile data-bin messages have exactly the same interpretation as non-extended tile data-bin messages and they refer to exactly the same tile data-bins. The extended tile messages include an Aux VBAS which identifies the smallest

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

n such that, in all components for which ($N_L - n$) is non-negative, resolution level ($N_L - n$) and all lower resolution levels have been completed when the bytes in this message are combined with all preceding bytes of the same tile, where N_L is the number of decomposition levels, which may vary by component. If no resolution levels of any component have been completed, the value of the Aux VBAS is one plus the maximum value of N_L across all components. The value zero is reached when all resolutions in all components have been completed. Because resolutions do not necessarily appear in order in a tile, some resolution levels above the value signalled by the VBAS may have been completed, but this cannot be determined from the message header. The information in the Aux VBAS may be useful to certain clients.

A.2.3 In-class identifiers

The least significant 4 bits of the first byte and the least significant 7 bits of all other bytes from the Bin-ID VBAS are concatenated in big-endian order to form a single word, having 7k-3 bits, where k is the number of bytes in the VBAS. This word represents an unsigned integer which serves to uniquely identify the data-bin within its class and codestream. A.3 provides a description of the various data-bin classes, along with the corresponding in-class identifiers.

A.3 Data-bins

A.3.1 Introduction

Data-bins contain portions of a JPEG 2000 file or codestream data. These may be based on imagery elements, such as precinct-based data, tile-based data, and headers. They may also be based on metadata. Whatever the content of a data-bin, each data-bin is treated as an individual bit-stream.

A.3.2 Precinct data-bins

A.3.2.1 Precinct data-bin format

Precinct data-bins appear only within the JPP-stream media-type. Each precinct data-bin corresponds to a single precinct within a single codestream. The in-class identifier is defined by Equation A-1.

$$I = t + (c + s \times \text{num_components}) \times \text{num_tiles} \qquad \text{(A-1)}$$

where:

- I is the unique identifier of the precinct within its codestream;
- t is the index (starting from 0) of the tile to which the precinct belongs;
- c is the index (starting from 0) of the image component to which the precinct belongs;
- s is a sequence number which identifies the precinct within its tile-component.

Within each tile-component, precincts are assigned contiguous sequence numbers, s, as follows. All precincts of the lowest resolution level (that containing only the LL sub-band samples) are sequenced first, starting from 0, following a raster-scan order. The precincts from each successive resolution level are sequenced in turn, again following a raster-scan order within their resolution level.

It follows that a precinct identifier of 0 refers to the upper left hand precinct from the LL sub-band of image component 0 in tile 0.

Each precinct data-bin corresponds to the string of bytes formed by concatenating all codestream packets, complete with all relevant packet headers, which belong to the precinct. It is conceivable that packet headers will be packed into PPM or PPT marker segments which shall then belong to main header or tile header data-bins, in which case the precinct data-bin would hold only packet bodies. In any event, the precinct data stream should coincide with the contiguous segment of bytes that would be found within a JPEG 2000 codestream having one of the layer-subordinate progression sequences (CPRL, PCRL or RPCL).

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

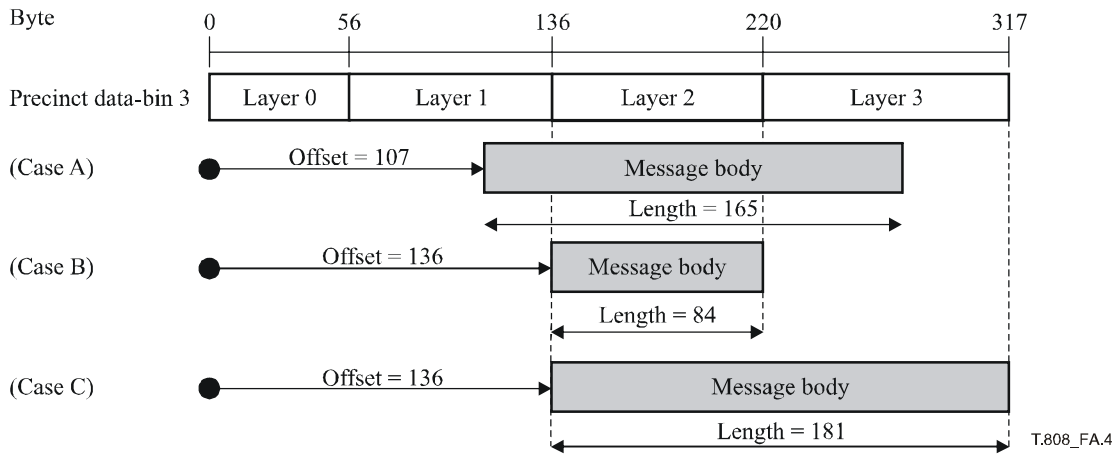

Figure A.4 – Example precinct data-bin

A.3.2.2 Precinct data-bin example (informative)

Figure A.4 shows an example precinct data-bin (in-class identifier 3) with 4 quality layers (or packets).

For Case A, B and C, the message header is shown below, based on the extended and non-extended precinct data-bin message structures. The <u>underlined</u> data denotes the Aux VBAS to identify the number of layers which are completed by the message.

(Case A)

Non-extended header: `00100011 01101011 10000001 00100101 xxxxxxxx …`

The initial 0 bit indicates only one byte is used in the Bin-ID VBAS. The next two bits ("01") indicate that no Class or CSn VBAS is present. The next "0" bit indicates that the data-bin is not completed by this message. The remaining bits of the first byte ("0011") indicate that the bin-ID is 3. The first bit of the second byte indicates that there is only one byte used in the Msg-Offset VBAS. The next 7 bits ("1101011") mean that the offset is 107. The first bit of the 3rd byte indicates that both this byte and at least the next byte are part of the Msg-Length VBAS. The 0 bit starting the 4th byte indicates that it is the last byte of the Msg-Length VBAS. Thus all the low order bits from the 3rd and 4th bytes are concatenated to determine the length. In this case, "0000001 0100101" = 165.

Extended header: `01000011 00000001 01101011 10000001 00100101` <u>`00000011`</u> `xxxxxxxx …`

(Case B)

Non-extended header: `00100011 10000001 00001000 01010100 xxxxxxxx …`

Extended header: `01000011 00000001 10000001 00001000 01010100` <u>`00000011`</u> `xxxxxxxx …`

(Case C)

Non-extended header: `00110011 10000001 00001000 10000001 00110101 xxxxxxxx …`

Extended header: `01010011 00000001 10000001 00001000 10000001 00110101` <u>`00000100`</u> `xxxxxxxx …`

Note that since the response data contains the last byte of the data-bin in Case C, the Bin-ID VBAS indicates that it is a "completed" message.

A.3.3 Tile header data-bins

Tile header data-bins appear only within the JPP-stream media type. For data-bins belonging to this class, the in-class identifier holds the index (starting from 0) of the tile to which the data-bin refers. This data-bin consists of markers and marker segments for tile n. It shall not contain an SOT marker segment. Inclusion of SOD markers is optional. This data bin may be formed from a legal codestream, by concatenating all marker segments except SOT and POC in all tile-part headers for tile *n*.

A.3.4 Tile data-bins

Tile data-bins shall be used only with the JPT-stream media type. For data-bins belonging to this class, the in-class identifier is the index (starting from 0) of the tile to which the data-bin belongs. Each tile data-bin corresponds to the string of bytes formed by concatenating all tile-parts belonging to the tile, in order, complete with their SOT, SOD and all other relevant marker segments.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

A.3.5 Main header data-bin

Both JPP- and JPT-stream media types use the main header data-bin. For data-bins belonging to the codestream main header class (completed or non-completed variations), the in-class identifier shall be 0. This data-bin consists of a concatenated list of all markers and marker segments in the main header, starting from the SOC marker. It contains no SOT, SOD or EOC markers.

A.3.6 Metadata-bins

A.3.6.1 Introduction to metadata-bins

Both JPP- and JPT-stream media types use metadata-bins. Metadata-bins are used to convey metadata from the logical target that contains the codestream or codestreams whose elements may be referenced by other data-bins associated with the JPP-stream or JPT-stream. For the purpose of this Recommendation | International Standard, the term "metadata" refers to any collection of "boxes" from a JPEG 2000 family file. The codestream index shall be ignored in any message which has the metadata-bin class identifier.

Unlike the numerical ID's used for other types of data-bins, metadata-bin ID's do not map algorithmically to some file format construct or byte offset. The server is free to choose any numeric ID for any particular metadata bin. The one and only one exception of this is that the metadata-bin containing the root of the logical target shall be given the ID of 0.

NOTE – The mechanism for assignment is implementation-dependent; however, it is an informative suggestion that servers assign bin-ID's using consecutive numbers.

A.3.6.2 Division of a logical target containing a JPEG 2000 file into metadata-bins

All metadata could conceivably be included in metadata-bin 0. In this case, all boxes from the logical target would belong to metadata-bin 0, appearing in their original order. Since JPEG 2000 family file formats consist of nothing but a sequence of boxes, this effectively means that metadata-bin 0 would consist of the entire logical target. More generally, however, it is useful to break the logical target into pieces that can be transmitted in a manageable fashion. This allows image servers to deliberately omit portions of the logical target that are not currently required by a client. To this end, JPIP defines a new special box type, known as the "Placeholder box." The Placeholder box serves to identify the size and type of a box from the logical target, while pointing to another data-bin that holds that box's contents. Placeholders are also able to represent codestreams from the logical target. This is particularly significant in view of the fact that the compressed data represented by any given codestream may be delivered incrementally via the other data-bin types (header data-bins and precinct data-bins or tile data-bins).

Formally, metadata-bin 0 consists of all boxes from the logical target, appearing in their original order, with the exception that a placeholder may replace any given box. The Placeholder box contains the original header of the box that has been replaced, together with the identifier of the metadata-bin that holds that box's contents, not including the header itself. Every metadata-bin, other than metadata-bin 0, shall consist of the contents of some box, whose header appears in the placeholder that references that data-bin. These box contents may themselves include sub-boxes, any of which may be replaced by further placeholders.

The following colour scheme will be used for metadata-bin example illustrations (FigureA.5):

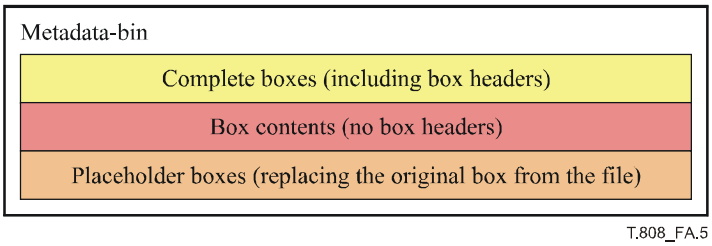

Figure A.5 – Metadata-bin example colour scheme

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

As an example, consider a simple JP2 file with the following box structure (Figure A.6):

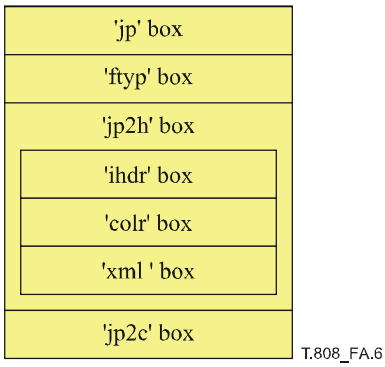

Figure A.6 – A sample JP2 file

This file may be divided up into three metadata-bins: one to represent the top-level of the original file (data-bin 0); one to represent the JP2 Header box; and one to represent the codestream. This division is shown in Figure A.7.

While the contents of any metadata-bin shall be the contents of the box or file represented by that bin, the actual data contained in those contents may conceptually vary depending on the type of box. For example, in Metadata-bin 1 in Figure A.7, representing the contents of the JP2 Header box, the contents of that box is literally a series of other complete boxes, as the JP2 Header box is a superbox. No data other than the series of those complete boxes may be found within Metadata-bin 1, as there is no other data in the JP2 Header box. In contrast, the data inside Metadata-bin 2 is the raw contents of the Contiguous Codestream box, with no box headers, because that box is not a superbox.

One point of particular interest to note from the example in Figure A.7 is that access to codestream data may be provided in two ways. The second placeholder bin is used to replace the contiguous codestream box (jp2c) in the original file. It identifies metadata-bin 2 as holding the original contents of this box, i.e., the raw codestream itself. For convenience of description in this Recommendation | International Standard, this shall be termed the "raw codestream" representation. Raw codestreams are served from metadata-bins.

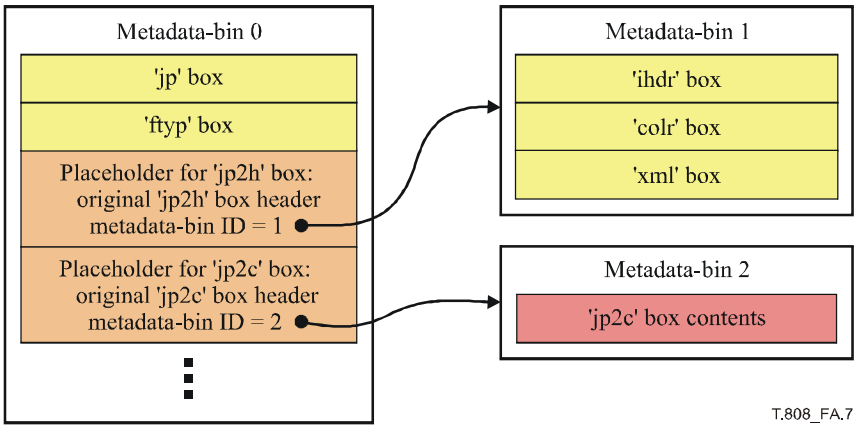

Figure A.7 – A sample JP2 file divided into three metadata-bins

The placeholder may also provide a codestream identifier. Any data-bins belonging to the main header, tile header, precinct or tile data-bin classes, having this same codestream identifier, convey compressed data associated with the same codestream as that found in metadata-bin 2. For convenience of description in this Recommendation | International Standard, this shall be termed the "incremental codestream" representation. Incremental codestreams are served from these data-bins.

In general, placeholders that reference codestream data may do so either by referencing a separate metadata-bin (raw codestream), or by providing a codestream identifier (incremental codestream), or both. Even if both methods are provided, the JPP-stream or JPT-stream data available at a client or image-rendering agent might only have the contents of the raw codestream, or only have data from the incremental codestream. Moreover, if both the raw and incremental versions of the same codestream are available, there is no guarantee that the two representations will have compatible

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

coding parameters. Only the reconstructed image samples associated with the two representations are guaranteed to be consistent.

It is also possible to use Placeholder boxes to associate multiple codestreams with a single original box. The interpretation of such an association is dependent upon the box being replaced. Further discussion of this topic appears in A.3.6.4.

In the simple example of Figure A.7, Placeholder boxes appear only at the top-level of the file, in metadata-bin 0. As already noted, however, placeholders may be used to replace any box, in any metadata-bin. This allows complex files to be decomposed in hierarchical fashion. As such, a single original file may be encapsulated in a variety of different metadata-bin structures, depending on how placeholders are used. **However, a single JPP-stream or JPT-stream shall adopt only one such encapsulation.** In client-server applications, the server will generally determine a suitable metadata-bin structure for the file, assigning a unique identifier to the resulting stream, and using the same metadata-bin structure in all communication with all clients which reference this same unique identifier.

When a placeholder relocates a box into a new metadata-bin, the header of that box (LBox, TBox and XLBox fields) is stored, unmodified, in the Placeholder box. If a client or rendering agent needs to map particular boxes to their original file offsets, it may do so using the original box headers that appear in the Placeholder boxes. This information ultimately allows any location in the original file to be mapped to a particular location in a particular metadata-bin, if the contents of that data-bin exist. This is important since some JPEG 2000 family files contain boxes that reference other boxes through their location within the file.

While considerable freedom exists in deciding how best to divide a file into metadata-bins, there is one restriction. Any Placeholder box that appears within a metadata-bin shall replace a top-level box within that data-bin. Equivalently, wherever a sub-box is to be replaced with a placeholder, its immediate containing super-box shall reside within its own metadata-bin. For example, in the sample file shown in Figure A.6, the XML data contained within the JP2 Header box may be placed in a separate data bin from the other boxes. This allows a server to deliver only those data-bins that are actually required for decoding and display of the image, unless XML data is explicitly requested. A suitable data-bin structure is shown in Figure A.8.

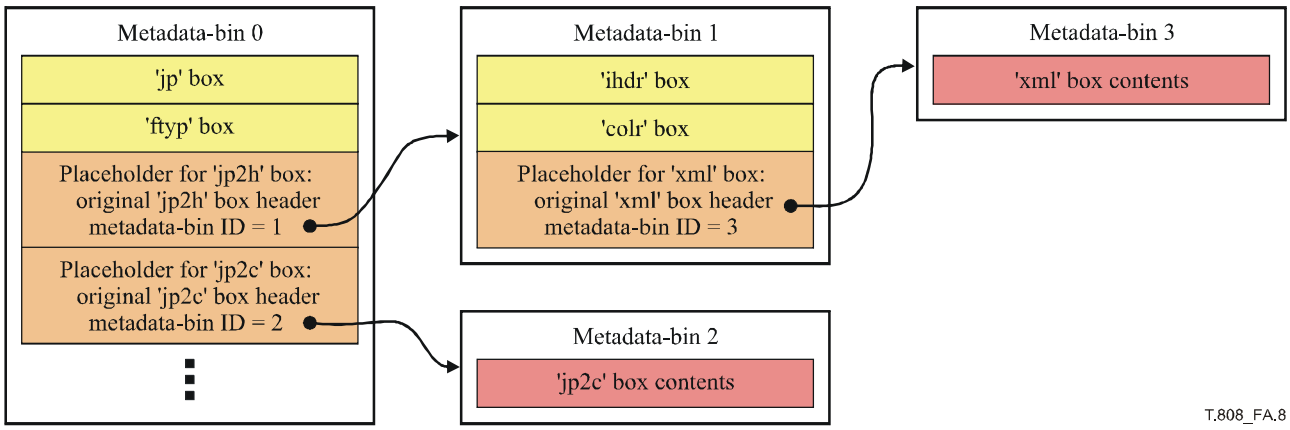

Figure A.8 – A superbox with a referenced metadata-bin

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

It would not be legal, however, for the JP2 Header box to be left in metadata-bin 0, as shown in Figure A.9:

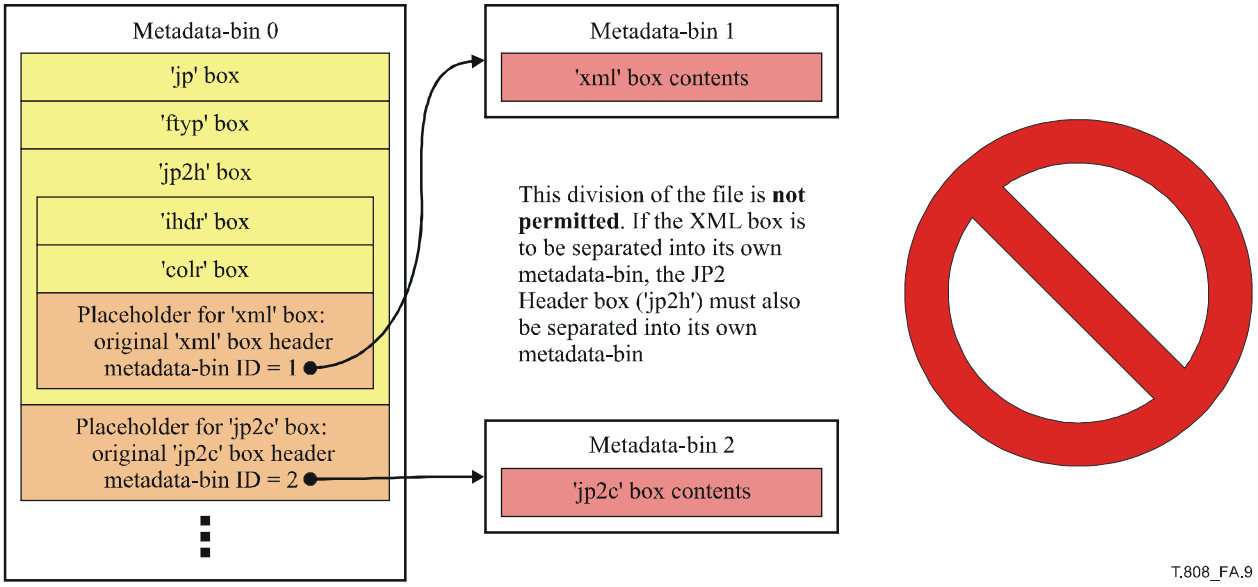

Figure A.9 – An illegal division of the file into metadata-bins

NOTE – An equivalent way to express this same restriction is as follows. Wherever a placeholder replaces a sub-box, a placeholder shall also replace its containing box. This restriction ensures that it is always possible for a client or rendering agent to recover the lengths and locations of the original boxes within the file, even if some of the boxes are not understood by the client.

In addition to providing the original contents of a box in a separate metadata-bin, JPP- and JPT-streams are also permitted to provide alternate representations of the box, which did not explicitly appear within the original file. These alternate representations are known as "stream equivalents." For example, the original file might contain a Cross-reference box whose fragment list box collects one or more fragments of the file to reconstitute a Colourspace Specification box. While a client or rendering agent should be able to follow the relevant file pointers to reconstruct the Colourspace Specification box, a more convenient JPP- or JPT-stream representation might contain a placeholder which references a data-bin containing the reconstructed Colourspace Specification box as a stream equivalent. To do this, the placeholder includes a box header for the stream equivalent, together with the identifier of the metadata-bin that holds the contents of the stream equivalent box.

The following example (shown in Figure A.10) illustrates the use of stream equivalents for Cross-reference boxes. In this case, the data-bin that holds the stream-equivalent contents is also referenced as holding the original contents of another box. While this is likely to be a common situation where the original file contained cross-reference boxes, there is no need for the stream-equivalent to point to a metadata-bin that is connected to the original file hierarchy. The stream equivalent box's contents may be created from scratch or they may refer to content which originally existed within other files. This allows Cross-reference boxes whose fragment list references other files or URLs to be fully encapsulated within a single JPP- or JPT-stream.

Stream equivalents may be used in any situation where the server can create an alternate form of the contents of a box that provide some benefit to the client; they are not just for providing access to explicitly cross-referenced data.

In addition to pointing to actual or equivalent box data, a placeholder box can point to one or more codestreams where the replaced box is equivalent to those codestreams. For example, the Contiguous Codestream box may be replaced by a placeholder box that references the ID of the incremental codestream contained within that Contiguous Codestream box. Another example would be to replace the Chunk Offset box in a Motion JPEG 2000 file with a placeholder that specifies an array of codestream ID's. Those codestream ID's refer to the codestreams that are pointed to by the Chunk Offset box.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

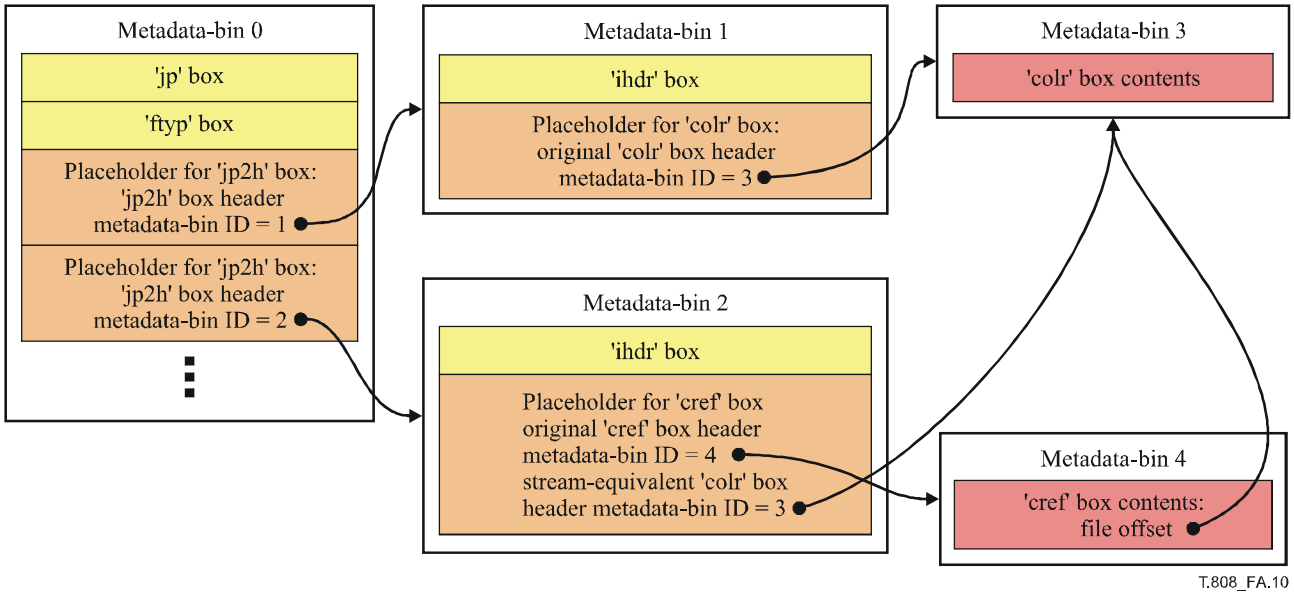

Figure A.10 – Example of the use of stream equivalents

A.3.6.3 Placeholder box format

Figure A.11 shows the format of a Placeholder box, including the box header (unlike the definition of most boxes in Annex I and other parts of this Recommendation | International Standard); it is specified this way to emphasise that the use of the length field in the box header for a Placeholder box is more restrictive than for other boxes.

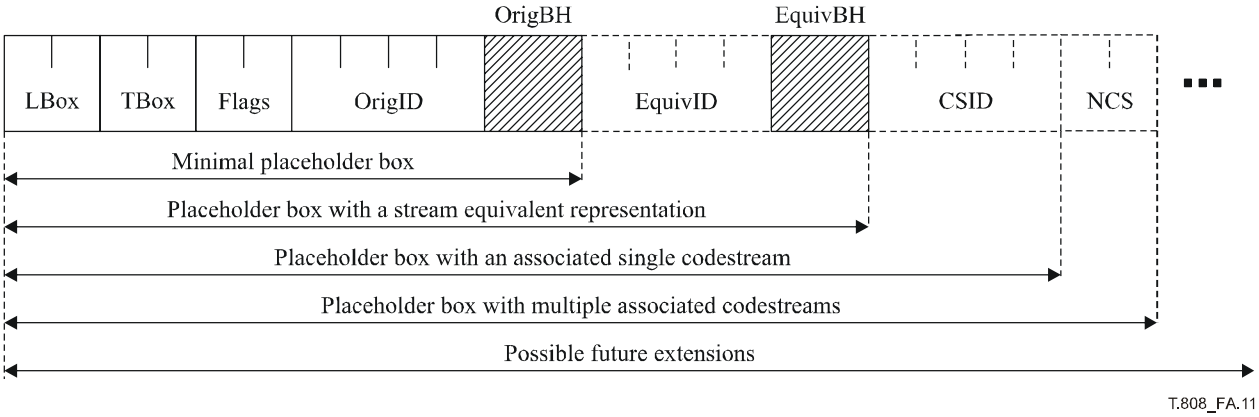

Figure A.11 – Placeholder box structure

LBox: This is the standard 4-byte big-endian length field for a box. The value shall not be 1 for a Placeholder box, meaning that the XLBox field shall not be present.

TBox: This is the standard 4-byte box type field for a box. The type value for a Placeholder box shall be 'phld' (0x7068 6c64).

Flags: This field specifies what elements of the Placeholder box contain valid data. This field is encoded as a 4-byte big-endian integer. Legal values for the Flags field are specified in Table A.3.

OrigID: This field specifies the metadata-bin ID of the bin containing the contents of the original box represented by this Placeholder box. It is encoded as an 8-byte big-endian unsigned integer.

OrigBH: This field specifies the original header (LBox, TBox and XLBox, as needed) of the original box referenced by this Placeholder box. The length of this field is 8 bytes if the original box header's LBox field is not equal to 1 and 16 bytes otherwise.

EquivID: This field specifies the metadata-bin ID of the bin that contains a stream-equivalent form of the contents of this box. This field is encoded as an 8-byte big-endian unsigned integer.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

EquivBH: This field specifies the header of the stream-equivalent box (LBox, TBox and XLBox as needed) of the box referenced by this Placeholder box. The length of this field is 8 bytes if the equivalent box header's LBox field is not equal to 1 and 16 bytes otherwise.

CSID: This field specifies the ID of the first codestream associated with the replaced box. This is the ID that is associated with all header, precinct and/or tile data-bins used to incrementally communicate the contents of the first codestream associated with the replaced box. This field is encoded as an 8-byte big-endian unsigned integer.

NCS: This field specifies the number of codestreams in the array of codestreams that is equivalent to the replaced box. The codestream ID values of these codestreams run contiguously from the value specified by the CSID field. This field is encoded as a 4-byte big-endian unsigned integer.

ExtendedBoxList: This field is not specifically shown in Figure A.11. The NCS field may be followed by a sequence of boxes containing extended information from the server. The existence of any box following the NCS field shall be specified through a bit in the Flags field. However, no extended boxes, nor any additional bit flags, are defined by this Recommendation | International Standard. Clients shall ignore any box in ExtendedBoxList that is not understood.

A bit value of "x" in Table A.3 indicates that the specified value includes cases where that bit is set to either "1" or "0". Bits indicated as "y" are unused by this standard and shall be set to 0 by servers and ignored by clients.

Not all of the fields defined for a Placeholder box need appear in every Placeholder box. As suggested by the arrows in Figure A.11, if no box equivalent or incremental codestream ID is provided, the box may be terminated at the end of the OrigBH field. Similarly, if no incremental codestream ID is provided, the box may be terminated at the end of the EquivBH field, and if no more than one incremental codestream ID is provided, the box may be terminated at the end of the CSID field.

Table A.3 – Legal values for the Flags field of a Placeholder box

Value	Meaning
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xxx1	Access is provided to the original contents of this box through the metadata-bin specified in the OrigID field
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xxx0	No access is provided to the original contents of this box, and the value of the OrigID field shall be ignored
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xx1x	A stream-equivalent box is provided, whose contents are in the metadata-bin specified by the EquivID field.
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xx0x	No stream-equivalent box is provided, and the value of any EquivID and EquivBH fields shall be ignored
yyyy yyyy yyyy yyyy yyyy yyyy yyyy 01xx	Access to the image represented by this box is provided by a single incremental codestream, which is identified by the CSID field. The value of the NCS field shall be treated as if was set to "1" regardless of the actual value of that field.
yyyy yyyy yyyy yyyy yyyy yyyy yyyy 11xx	Access to the image represented by this box is provided by one or more incremental codestreams, as specified by the CSID and NCS fields.
yyyy yyyy yyyy yyyy yyyy yyyy yyyy x0xx	This placeholder does not provide access to an image representing the original box as an incremental codestream; the CSID and NCS fields shall be ignored.
Other values	Reserved for ISO use

A.3.6.4 Referencing of incremental codestreams with placeholders

Wherever header, precinct or tile data-bins exist, their codestream ID shall appear in a Placeholder box within an appropriate metadata-bin. The only exception to this requirement is for unwrapped JPEG 2000 codestreams, which are not embedded within a JPEG 2000 family file format.

The codestream ID values that appear within the relevant Placeholder box shall conform to any requirements imposed by the containing file format. For example, JPX files formally assign a sequence number to each codestream that appears at the top level of the file, either through a Contiguous Codestream box or a Fragment Table box. The first top-level codestream in the logical target shall have a codestream ID of 0; the next shall have a codestream ID of 1; and so forth.

Placeholders that reference multiple codestream ID's may be used only where the meaning of those codestreams is well defined by the type of the box that is being replaced.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

A.3.6.5 Using Placeholder boxes with MJ2

This Recommendation | International Standard defines only two box types suitable for placeholders with Motion JPEG 2000 (MJ2) files. Specifically, either the chunk offset box ('stco') or the chunk large offset box ('co64') may be replaced by a Placeholder box which identifies multiple codestream ID's.

Each video track in an MJ2 file contains exactly one chunk offset box (either 'stco' or 'co64') that, in combination with the sample to chunk box ('stsc'), serves to identify the locations of all of the contiguous codestream boxes that belong to the video track. If the chunk offset box is replaced by a placeholder that provides one or more codestream ID's, there shall be exactly one codestream ID for each contiguous codestream box in the video track. If the visual sample entry box ('mjp2') identifies a field count of 2, there shall be 2*N* codestream ID's in the range provided by the Placeholder box, where *N* is the number of video samples (i.e., *N* is the number of frames). Otherwise, there shall be only *N* codestream ID's in the range provided by the Placeholder box. The codestream ID's shall be sequenced by sample number (frame number) and by field number within each sample.

NOTE – For MJ2 files in a JPP-stream or a JPT-stream representation, there is no need for the stream to contain the contents of the original chunk offset box, the sample to chunk box ('stsc'), or the sample size box ('stsz'). This indexing information can be regenerated if needed if the stream representation is converted to an MJ2 file.

A.4 Conventions for parsing and delivery of JPP-streams and JPT-streams (informative)

Placeholder boxes create additional flexibility and some potential ambiguity for both clients and servers in how they parse or deliver JPP- and JPT-streams. A server may choose to partition original boxes from a JPEG 2000 family file into metadata-bins using any of a wide range of strategies, by introducing Placeholder boxes at appropriate points. The server shall do this in a consistent way so that the data-bins associated with a JPP- or JPT-stream have the same nominal contents for all clients which access the same logical target (possibly qualified by a unique target ID), whenever they access it.

More significantly, however, Placeholder boxes allow servers to construct a single JPP- or JPT-stream whose data-bins provide multiple alternate representations of the same original content. This can happen when a streaming equivalent is identified within a placeholder, and/or when an incremental codestream ID is identified within a placeholder. In these cases, an original box might be made available in a metadata-bin, while also being made available as a stream equivalent in yet another metadata-bin, and/or also being made available as an incremental codestream via header, precinct or tile data-bins. While servers might distribute the contents of all data-bins that represent an original box, for efficiency reasons servers would be expected to distribute only sufficient information to convey the original content, unless explicitly asked to distribute redundant data-bins. Client-side parsers of JPP- or JPT-streams, when confronted with multiple representations of an original box, might choose to ignore all but one of the representations. The expected client convention should have a significant impact on which metadata-bins the server chooses to actually send to a client.

In view of this, this Recommendation | International Standard recommends the following conventions:

- Unless a server has reason to believe otherwise, it shall assume that the client parser will parse a stream equivalent box in preference to the original box if the presence of both box types has been signalled to the client by placeholders.
- Unless a server has reason to believe otherwise, it shall assume that the client parser will use the incremental codestream representation (header, precinct or tile data-bins) in preference to a raw codestream if the presence of both box types has been signalled to the client by placeholders.

A.5 Conventions for JPP-stream or JPT-stream interoperability (informative)

This convention describes the exchange file format for JPP-stream and JPT-stream, herein termed jpp-file and jpt-file respectively. Such a file may contain the received JPEG 2000 data from a JPIP session (the client's cache for example), or a subset thereof. It is possible for another JPIP client to read and use this file because JPP-stream and JPT-stream are self-describing media types.

These files are formed by concatenation of JPT-stream or JPP-stream messages. For example, they may be formed by the simple concatenation of all such messages received by a client in a single session or from multiple sessions. An improved situation would be where clients generated a legal JPT-stream or JPP-stream using a single Message Header and Message per data-bin.

It is recommended that the ".jpp" and ".jpt" extensions be used for these files and, if appropriate, that the file name includes a reference to a relevant JPIP `target` token or `target-id` token.

This convention does not specify the implementation or structure of the cache for a client. For example, a client may use a database to serve as its implementation of the cache function rather than a file-based cache system.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

Annex B

Sessions, channels, cache model and model-sets

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

B.1 Requests within a session vs. stateless requests

The JPIP protocol makes a clear distinction between two different types of requests: stateless requests and requests which belong to a session.

The purpose of sessions is to reduce the amount of explicit communication required between the client and server. Within a session, the server is expected to remember client capabilities and preferences supplied in previous requests so that this information need not be sent in each and every request. Even more importantly, the server would typically maintain a log of the information it has already sent to the client in response to previous requests, so that this information need not be re-transmitted in response to future requests. This log would be persistent for the duration of a session. Unless explicitly instructed otherwise, the server may assume that the client caches the responses to all requests issued within a session, and may model the client's cache, sending only those portions of the compressed image data or metadata which the client does not already have in its cache.

Stateless requests are not associated with any session and so shall be entirely self-contained. It should be noted that the term "stateless" applies only to the server, not the client. As for sessions, the client should generally cache the responses from previous requests associated with the same logical target. Clients that issue multiple stateless requests for the same target should generally include information about their cache contents with each request, so as to avoid the transmission of redundant data. Thus, the benefits of sessions are smaller, less complex requests and/or less redundant response data from the server. The benefit of stateless communication is that the server need not maintain state information between requests; this means that the same host need not ultimately serve all requests for a single target image that emanate from a single client.

B.2 Channels and sessions

Associated with each session are the following elements:

- One or more logical targets (usually image files), whose content does not change over the session.
- A single image data return type for each logical target associated with the session.
- For each logical target associated with the session, a model of the client's cache contents shall be maintained wherever the data return type is one of "jpp-stream" or "jpt-stream". Note, however, that this model need not perfectly reflect the actual state of the client's cache. Rules governing the maintenance of cache models are outlined in B.3.
- One or more JPIP channels. Clients may generally open multiple channels within the same session. Each JPIP channel may be associated with a separate underlying transport channel (e.g., a separate TCP connection), although this might not be the case. Multiple channels allow clients to issue simultaneous requests for multiple image regions, with the expectation that the server will respond to these requests concurrently. Channels also allow for intelligent bandwidth allocation amongst different types of requests either within a single target image or across multiple targets.
- Where multiple channels are associated with the same logical target, the session cache model applies across all channels. Multiple clients may open JPIP channels within the same session, although this might have undesirable side effects if the channels refer to the same logical target.

Associated with each channel are the following elements:

- A single logical target (usually an image file).
- A server-assigned identifier that shall be included with each request. JPIP does not define a separate session identifier, since the channel identifier is sufficient to associate the request with its session.
- A record of the client's capabilities and preferences, which may be adjusted through appropriate request fields.
- To the extent that the server queues requests, it should provide a separate queue for each JPIP channel.

There is a one-to-one correspondence for the client request and client response on a channel. Different JPIP channels may be on the same transport channel or on different transport channels. Requests that use different JPIP channels may arrive asynchronously at the server if separate transport channels are used to transport the requests. Responses that use different JPIP channels may arrive asynchronously at the client if separate transport channels are used to transport the

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

responses. Servicing of multiple channels is at the discretion of the server, however, the delivery rate request field and the max bandwidth and bandwidth slice preferences should guide the server.

B.3 Cache model management

As already noted, one of the principal functions of a session is that of server-side modelling of the client's cache. Unless explicitly informed otherwise, the server **may** assume that the client has cached all information sent in response to requests within the session: this information **need not** be re-transmitted. Note, however, that the server is not obliged to maintain a complete cache model or indeed any cache model at all: redundant data **may** be transmitted in response to requests.

In addition to the impact of transmitted data, explicit cache model manipulation statements in client requests may update the server's cache model. These statements are to be processed before determining the data that should be returned to the client in response to its request. There are two types of cache model manipulation statements: additive and subtractive.

Additive cache model manipulation statements serve to augment the server's cache model, adding data-bins, or portions of data-bins to the existing model. These provide a mechanism for a client to inform the server about information which it received in a previous session, or using previous stateless requests. A server **should** attempt to exploit any additive cache model manipulation statements that appear in client requests. However, servers are not obliged to maintain a complete cache model, so a server may disregard, or partially disregard, additive cache model manipulation statements.

Subtractive statements serve to remove data-bins, or portions of data-bins from the server's cache model. A client might issue subtractive cache model manipulation statements in order to inform the server that it has not cached or has discarded some data which was previously sent by the server. The server is otherwise free to assume that the client has cached all data transmitted during the session. The server **shall** remove all information identified by a subtractive cache model statement from any cache model (complete or otherwise) that it is maintaining.

Session-based JPIP requests have side effects, which may affect the response to future requests. This is true also of requests that contain cache model manipulation statements – the effects of cache model manipulation are persistent. Moreover, the side effects of a request arriving on one JPIP channel are reflected in the response to any requests that might belong to a different JPIP channel which is associated with the same logical target. This follows from the fact that there is only one cache model for each logical target in a session.

B.4 Interrogation and manipulation of model-sets

Where a logical target associated with a session contains a large number of codestreams (e.g., a video target), or a client remains connected for a long period of time, partial cache modelling becomes an increasingly likely strategy for practical server implementations. It also becomes increasingly likely that clients will be unable to cache all information sent by the server. To avoid communication inefficiencies in such circumstances, the concept of an "mset" (model-set) is introduced. The "mset" is the collection of codestreams for which client cache contents are being modelled by the server.

In any request, the client may instruct the server to limit its "mset" to a particular set of codestreams. This provides a convenient mechanism for clients to discard whole codestreams from their cache without running the risk that the server will generate incomplete responses to future requests for those codestreams.

"mset" requests also result in server responses which indicate the actual set of codestreams for which cache model information is being maintained. This allows clients to determine whether or not cache model manipulation statements which refer to a variety of codestreams will be disregarded by the server.

In the absence of any explicit "mset" manipulation or interrogation, the client may assume only that the server's "mset" includes all codestreams for which response data is generated to its request. Since servers generally have the right to limit the scope of a client's request to a smaller number of codestreams than the number which was originally specified, there is no guarantee that the server's "mset" will include all of the codestreams mentioned in a request, unless the request mentioned only one codestream. These matters are explained further in C.8.6.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

Annex C

Client request

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

C.1 Request syntax

C.1.1 Introduction

This annex describes all possible elements in a JPIP request. Each major subclause describes a group of fields and possible values for those fields. In general, a request will consist of fields from more than one group, but some groups are incompatible. Further, within each group, some request fields are incompatible. Some otherwise legal requests may not be valid for use in some situations (e.g., sessions), even though this is not indicated by the BNF syntax. Finally, even with a legal request, a server may not support all possible request fields or combinations thereof.

C.1.2 Request structure

The JPIP request consists of the following fields:

- Target identification fields;
- Session and channel management fields;
- View-window request fields;
- Metadata field;
- Data limiting request fields;
- Server control request fields;
- Cache management request fields;
- Upload request fields;
- Client capability and preference fields.

The elements in the request shall be sent in compliance with the selected transport protocol. For example, in HTTP, the requests are expressed as the characters listed in the BNF syntax, multiple parameters are joined with an "&" character, and the requests may be part of the query field of a GET request, or the body of a POST request. See Annexes F, G and H for details.

NOTE – URI reserved characters may be escaped. For example, "request=a:b" in a HTTP GET URL would result in "request=a%3Ab" where the URL-reserved character ':' being escaped to '%3A'.

```
jpip-request-field = target-field
                   / channel-field
                   / view-window-field
                   / metadata-field
                   / data-limit-field
                   / server-control-field
                   / cache-management-field
                   / upload-field
                   / client-cap-pref-field

target-field       = target                        ; C.2.2
                   / subtarget                     ; C.2.3
                   / tid                           ; C.2.4

channel-field      = cid                           ; C.3.2
                   / cnew                          ; C.3.3
                   / cclose                        ; C.3.4
                   / qid                           ; C.3.5
```


Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

```
view-window-field  = fsiz                          ; C.4.2
                   / roff                          ; C.4.3
                   / rsiz                          ; C.4.4
                   / comps                         ; C.4.5
                   / stream                        ; C.4.6
                   / context                       ; C.4.7
                   / srate                         ; C.4.8
                   / roi                           ; C.4.9
                   / layers                        ; C.4.10

metadata-field     = metareq                       ; C.5.2

data-limit-field   = len                           ; C.6.1
                   / quality                       ; C.6.2

server-control-field = align                       ; C.7.1
                   / wait                          ; C.7.2
                   / type                          ; C.7.3
                   / drate                         ; C.7.4

cache-management-field = model                     ; C.8.1
                   / tpmodel                       ; C.8.3
                   / need                          ; C.8.4
                   / tpneed                        ; C.8.5
                   / mset                          ; C.8.6

upload-field       = upload                        ; C.9.1

client-cap-pref-field = cap                        ; C.10.1
                   / pref                          ; C.10.2
                   / csf                           ; C.10.3
```

C.1.3 Restrictions on combining request fields

Each type of JPIP request field shall occur no more than once in a single request.

In general, requests for image data (view-window requests) may be combined with requests for additional metadata. However, there are restrictions on how the request fields may be combined.

The upload request field shall not be combined with `metadata-field`, `data-limit-field`, or `server-control-field`.

C.2 Target identification fields

C.2.1 Introduction to logical targets

Each JPIP request is directed to a specific representation of a specific original named resource or a specific portion of that resource. That resource may be a physically stored file or object, or may be something that is created virtually by the server upon request.

The specific representation, whether that is the original encoded form or a transcoded form, or whether that is a specific byte range or is the entire resource, is referred to as the logical target. The logical target is specified through three request fields: Target ID, Target and Sub-target.

The Target request field specifies the original named resource to which the request is directed. It is specified using a `PATH`, which could be a simple string or a URI. If the Target field is not specified and the request is carried over HTTP, then the JPIP request shall be directed to the resource specified through the path component of the JPIP request URL. This original named resource may be an actual file or other object physically stored on the server, or it may be something that the server creates in response to a JPIP request.

The Sub-target request field specifies the specific byte range of the original named resource (specified through the Target request field) to which the request is directed. If the Sub-target request field is not specified, the request is directed to the entire byte range of the original resource.

The Target ID request field can be used to further specify a particular encoding of the resource in situations where the client and server have previously exchanged data from this resource. For example, the server may have previously supplied a transcoded version of the file to the client based on information supplied and the conditions around a previous request. If that client has preserved the data previously transmitted in its cache, it will desire to continue to receive data using that same transcoding so that it can continue to use the data in the cache. The Target ID is a server

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005

defined identification string, to which the server has previously associated with that specific representation of that specific original named resource, or a byte range of some specific original named resource.

If a client specifies both the original named resource (through either the Target request field or through the path component of the JPIP Request URL) and Target-ID, the server shall verify whether or not it can respond to the request in the same manner as when it originally assigned that Target ID to that resource. If the server cannot respond in the same manner, it shall use a JPIP-tid response header to inform the client of a new Target ID, at which point the client will know that it must discard any previously cached data.

If a logical target is to be served with JPP-stream or JPT-stream messages, the associated data-bins shall remain consistent throughout all responses that are issued within the same session. Where the server, or a related server, also issues a Target ID, the data-bins shall remain consistent across all responses issued with the same Target ID, whether they are issued within the same session or not.

If this request is part of a session and a channel ID as been assigned by the server, the client may specify the channel ID through the Channel ID request field instead of specifying the Target, Sub-target and Target ID request field. If the logical target is specified both through a combination of the Target, Sub-target and Target ID fields, and through the Channel ID request field, then the server shall respond with an error.

The following examples show the specification of logical targets:

EXAMPLE 1: For JPIP request URL of

"http://one.jpeg.org/imageserver.cgi?target= http%3A%2F%2Fone.jpeg.org%2Fimages%2Fpicture.jp2&fsiz=200,200" the logical target is the entire byte range contained within the URI "http://one.jpeg.org/images/picture.jp2," relative to the server root document directory.

EXAMPLE 2: For JPIP request URL of

"http://one.jpeg.org/imageserver.cgi? target= http%3A%2F%2Fone.jpeg.org%2Fimages%2Fpicture.jp2&tid=4384-5849-af4d-3dca&fsiz=200,200" the logical target is the entire byte range contained within the URI "http://one.jpeg.org/images/picture.jp2," relative to the server root document directory, with a representation specified by the server defined Target ID 4384-5849-af4d-3dca.

EXAMPLE 3: For JPIP request URL of

"http://one.jpeg.org/imageserver.cgi?target= http%3A%2F%2Fone.jpeg.org%2Fimages%2Fpicture.jp2&subtarget=1038-13458&fsiz=200,200" the logical target is the range of bytes, starting with byte 1038, and all bytes up to and including bytes 13458, contained within the URI "http://one.jpeg.org/images/picture.jp2," relative to the server root document directory.

EXAMPLE 4: For JPIP request URL of "http://one.jpeg.org/imageserver.cgi?cid=1234-5849-af4d-3dca&fsiz=200,200" the logical target is the resource to which the server has associated with the channel with ID 1234-5849-af4d-3dca.

EXAMPLE 5: For JPIP request URL of "http://one.jpeg.org/images/picture.jp2?fsiz=200,200" the logical target is the entire byte range contained within the file "images/picture.jp2," relative to the server root document directory.

EXAMPLE 6: For JPIP request URL of "http://one.jpeg.org/images/picture.jp2?subtarget=1038-13458&fsiz=200,200" the logical target is the range of bytes, starting with byte 1038, and all bytes up to and including byte 13458, contained within the file "images/picture.jp2," relative to the server root document directory.

C.2.2 Target (target)

```
target = "target" "=" PATH
```

This field is used to specify the original named resource (often the name of a file on the server). If the Target request field is missing then the original named resource is determined by other means.

C.2.3 Sub-target (subtarget)

```
subtarget = "subtarget" "=" byte-range

byte-range = UINT-RANGE
```

This field may be used to qualify the original named resource through the specification of a byte range. The logical target is to be interpreted as the indicated byte range of the original named resource.

The lower and upper bounds of the supplied byte-range are inclusive, and 0 refers to the first byte of the target file.

Documents
Document Sharing Hub
Full document details: ISO_IEC 15444-9_2005