International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.808

Amendment 5

(03/2013)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Still-image compression – JPEG 2000

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

Amendment 5: UDP transport and additional enhancements to JPIP

Recommendation ITU-T T.808 (2005) – Amendment 5

ITU-T

International Telecommunication Union

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

Facsimile – Framework	T.0–T.19
Still-image compression – Test charts	T.20–T.29
Facsimile – Group 3 protocols	T.30–T.39
Colour representation	T.40–T.49
Character coding	T.50–T.59
Facsimile – Group 4 protocols	T.60–T.69
Telematic services – Framework	T.70–T.79
Still-image compression – JPEG-1, Bi-level and JBIG	T.80–T.89
Telematic services – ISDN Terminals and protocols	T.90–T.99
Videotext – Framework	T.100–T.109
Data protocols for multimedia conferencing	T.120–T.149
Telewriting	T.150–T.159
Multimedia and hypermedia framework	T.170–T.189
Cooperative document handling	T.190–T.199
Telematic services – Interworking	T.300–T.399
Open document architecture	T.400–T.429
Document transfer and manipulation	T.430–T.449
Document application profile	T.500–T.509
Communication application profile	T.510–T.559
Telematic services – Equipment characteristics	T.560–T.649
Still-image compression – JPEG 2000	**T.800–T.829**
Still-image compression | JPEG XR	T.830–T.849
Still-image compression – JPEG-1 extensions	T.850–T.899

For further details, please refer to the list of ITU-T Recommendations.

INTERNATIONAL STANDARD ISO/IEC 15444-9
RECOMMENDATION ITU-T T.808

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

Amendment 5

UDP transport and additional enhancements to JPIP

Summary

Amendment 5 to ITU-T T.808 | ISO/IEC 15444-9 contains a minor enhancement that provides tools and definitions to enable the user datagram protocol (UDP) as transport layer for the JPEG 2000 Interactive Protocol (JPIP). A UDP packaging of messages into chunks is described, and mechanisms for flow control and to detect and recover from package loss are introduced. UDP transport is defined in the new Annex K.

Amendment 5 also includes a minor clarification in the definition of the Placeholder box.

Since Amendment 3 of ITU-T T.801 | ISO/IEC 15444-2 adds the possibility to rotate compositing layers by multiples of 90 degrees and/or to flip them horizontally and vertically, the definition of the view window and context range of ITU-T T.808 | ISO/IEC 15444-9 had to be extended to respect this additional feature.

A further addition allows a JPIP client to identify the request fields that a server is prepared to handle. This new request addresses the need of a client to identify the capabilities of the server beforehand, and thus to adapt and optimize its strategy to retrieve data from the server.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T T.808	2005-01-08	16
1.1	ITU-T T.808 (2005) Amd. 1	2006-05-29	16
1.2	ITU-T T.808 (2005) Cor. 1	2007-01-13	16
1.3	ITU-T T.808 (2005) Amd. 2	2007-08-29	16
1.4	ITU-T T.808 (2005) Cor. 2	2008-06-13	16
1.5	ITU-T T.808 (2005) Amd. 3	2008-06-13	16
1.6	ITU-T T.808 (2005) Amd. 4	2010-05-22	16
1.7	ITU-T T.808 (2005) Cor. 3	2011-05-14	16
1.8	ITU-T T.808 (2005) Amd. 5	2013-03-16	16

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

CONTENTS

INTERNATIONAL STANDARD
RECOMMENDATION ITU-T

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

Amendment 5

UDP transport and additional enhancements to JPIP

1) Clause 6.1

Replace the third paragraph in this clause (between Figure 3 and Figure 4) with the following text:

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 the JPIP protocol over HTTP, TCP and UDP, and provides suggestions for other example implementations. The JPIP protocol itself is neutral with respect to underlying transport mechanisms for the client requests and server responses, except in regard to channel requests represented by the New Channel ("cnew") request field (see C.3.3) and the New Channel ("JPIP-cnew") response header (see D.2.3), where transport-specific details shall be communicated. This Recommendation | International Standard defines four specific transports, which are identified by the strings "http", "https", "http-tcp" and "http-udp" in the value string associated with New Channel requests.

2) Clause A.3.6.4

Replace clause A.3.6.4 with the following new clause A.3.6.4:

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 codestreams that are found in Contiguous Codestream boxes or Fragment Table boxes, either at the top level of the file, or within Multiple Codestream boxes. The first 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 IDs may be used only where the meaning of those codestreams is well defined by the type of the box that is being replaced. For JPX files, Contiguous Codestream boxes, Fragment Table boxes and Multiple Codestream boxes may be replaced by Placeholder Boxes that specify codestream IDs. Placeholders replacing Contiguous Codestream boxes and Fragment Table boxes may specify only a single codestream ID, while a placeholder replacing a Multiple Codestream box may specify multiple codestream IDs, corresponding to the number of codestreams that are found within the box.

3) Clause B.1

Replace the second paragraph of B.1 with the following:

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 every request. Even more importantly, the server may keep a log of data it knows the client to have received so that this information need not be re-transmitted in response to future requests. This log is subsequently referred to as the cache model. The cache model would typically be persistent for the duration of a session. Unless explicitly instructed otherwise, the server may assume that the client caches all data it receives 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.

4) Clause C.1.2

Replace the `server-control-field` *and* `client-cap-pref-field` *lists with the following:*

```
server-control-field = align                    ; C.7.1
                     / wait                     ; C.7.2
                     / type                     ; C.7.3
                     / drate                    ; C.7.4
                     / sendto                   ; C.7.5
                     / abandon                  ; C.7.6
                     / barrier                  ; C.7.7
                     / twait                    ; C.7.8

client-cap-pref-field = cap                     ; C.10.1
                     / pref                     ; C.10.2
                     / csf                      ; C.10.3
                     / handled                  ; C.10.4
```

5) Clause C.3.3

Replace the second and third paragraphs with the following text:

The value string identifies the names of one or more transport protocols that the client is willing to accept. This Recommendation | International Standard defines only the transport names, "http", "https", "http-tcp", and "http-udp". Details of the use of JPIP over the "http" transport appear in Annex F. Annex G describes the use of JPIP over the "http-tcp" transport and Annex K describes the use of JPIP over the "http-udp" transport.

If the server is willing to open a new channel, using one of the indicated transport protocols, it shall return the new channel identifier token using the New Channel response header (see D.2.3). In this case, the present request is the first request within the new channel.

6) Equation C-3

Modify equation C-3 with the following augmented version of the equation and subsequent explanatory text, to take account of the new rotation support in ISO/IEC 15444-2:2004/Amd.3. While making these editorial changes, note that many of the symbols from the original equation are similar.

First, define the rotated frame size, offset, width and height of the composite image as follows:

$$\begin{bmatrix} ox^F, oy^F, \\ XO_{inst}^F, YO_{inst}^F \end{bmatrix} = \begin{bmatrix} (fx - ox - sx), & (fy - oy - sy), \\ (W_{comp} - XO_{inst} - Wt_{inst}), & (H_{comp} - YO_{inst} - Ht_{inst}) \end{bmatrix}$$

$$\begin{bmatrix} fx^\theta, fy^\theta, ox^\theta, oy^\theta, W_{comp}^\theta, H_{comp}^\theta, \\ XO_{inst}^\theta, YO_{inst}^\theta, Wt_{inst}^\theta, Ht_{inst}^\theta \end{bmatrix} = \begin{cases} \begin{bmatrix} fx, & fy, & ox, & oy, & W_{comp}, & H_{comp}, \\ XO_{inst}, & YO_{inst}, & Wt_{inst}, & Ht_{inst} \end{bmatrix} & \text{if } R_{inst} = 0^\circ \mid \text{NoFlip} \\ \begin{bmatrix} fy, & fx, & oy, & ox^F, & H_{comp}, & W_{comp}, \\ YO_{inst}, & XO_{inst}^F, & Ht_{inst}, & Wt_{inst} \end{bmatrix} & \text{if } R_{inst} = 90^\circ \mid \text{NoFlip} \\ \begin{bmatrix} fx, & fy, & ox^F, & oy^F, & W_{comp}, & H_{comp}, \\ XO_{inst}^F, & YO_{inst}^F, & Wt_{inst}, & Ht_{inst} \end{bmatrix} & \text{if } R_{inst} = 180^\circ \mid \text{NoFlip} \\ \begin{bmatrix} fy, & fx, & oy^F, & ox, & H_{comp}, & W_{comp}, \\ YO_{inst}^F, & XO_{inst}, & Ht_{inst}, & Wt_{inst} \end{bmatrix} & \text{if } R_{inst} = 270^\circ \mid \text{NoFlip} \\ \begin{bmatrix} fx, & fy, & ox^F, & oy, & W_{comp}, & H_{comp}, \\ XO_{inst}^F, & YO_{inst}, & Wt_{inst}, & Ht_{inst} \end{bmatrix} & \text{if } R_{inst} = 0^\circ \mid \text{Flip} \\ \begin{bmatrix} fy, & fx, & oy, & ox, & H_{comp}, & W_{comp}, \\ YO_{inst}, & XO_{inst}, & Ht_{inst}, & Wt_{inst} \end{bmatrix} & \text{if } R_{inst} = 90^\circ \mid \text{Flip} \\ \begin{bmatrix} fx, & fy, & ox, & oy^F, & W_{comp}, & H_{comp}, \\ XO_{inst}, & YO_{inst}^F, & Wt_{inst}, & Ht_{inst} \end{bmatrix} & \text{if } R_{inst} = 180^\circ \mid \text{Flip} \\ \begin{bmatrix} fy, & fx, & oy^F, & ox^F, & H_{comp}, & W_{comp}, \\ YO_{inst}^F, & XO_{inst}^F, & Ht_{inst}, & Wt_{inst} \end{bmatrix} & \text{if } R_{inst} = 270^\circ \mid \text{Flip} \end{cases} \quad \text{(C-3a)}$$

In the above, W_{comp} and H_{comp} are the width and height of the composited image, specified in the composition box; Wt_{inst} and Ht_{inst} are the composited width and height as determined by the compositing instruction; XO_{inst} and YO_{inst} are the horizontal and vertical compositing offsets as determined by the compositing instruction; Ws_{inst} and Hs_{inst} are the width and height of the potentially cropped compositing layer as determined by the compositing instruction; XC_{inst} and YC_{inst} are the horizontal and vertical compositing layer cropping offsets as determined by the compositing instruction; and R_{inst} is derived from the ROT field of the compositing instruction, if any. If the compositing instruction contains no ROT field or the ROT field is 0, $R_{inst}=0^o$|NoFlip. Otherwise, the rotation angle for R_{inst} (expressed in degrees clockwise) is obtained from the least significant 3 bits of the ROT field using Table M-47 of Rec. ITU-T T.801 | ISO/IEC 15444-2, while the Flip|NoFlip status for R_{inst} is set to Flip if bit 4 of the ROT field is non-zero and NoFlip otherwise.

Then, define the modified frame size `fx″`, `fy″` as follows:

$$fx'' = \left\lceil fx^\theta \cdot \frac{XR_{reg}}{XS_{reg}} \cdot \frac{Wt_{inst}^\theta}{Ws_{inst}} \cdot \frac{W_{cod}}{W_{comp}^\theta} \right\rceil; \quad fy'' = \left\lceil fy^\theta \cdot \frac{YR_{reg}}{YS_{reg}} \cdot \frac{Ht_{inst}^\theta}{Hs_{inst}} \cdot \frac{H_{cod}}{H_{comp}^\theta} \right\rceil \quad \text{(C-3b)}$$

To compute the modified region, first define the clipped region edges:

$$x_{min} = \left\lfloor XO_{inst}^\theta \cdot \frac{fx^\theta}{W_{comp}^\theta} \right\rfloor; \quad y_{min} = \left\lfloor YO_{inst}^\theta \cdot \frac{fy^\theta}{H_{comp}^\theta} \right\rfloor$$

$$x_{lim} = \left\lceil (XO_{inst}^\theta + Wt_{inst}^\theta) \cdot \frac{fx^\theta}{W_{comp}^\theta} \right\rceil; \quad y_{lim} = \left\lceil (YO_{inst}^\theta + Ht_{inst}^\theta) \cdot \frac{fy^\theta}{H_{comp}^\theta} \right\rceil \quad \text{(C-3c)}$$

The modified region size `sx″` and `sy″` and region offsets `ox″` and `oy″` are then given as:

$$
\begin{aligned}
\text{sx}'' &= \min\left\{\left(\text{ox}^{\theta} + \text{sx}^{\theta}\right), x_{\text{lim}}\right\} - \max\left\{\text{ox}^{\theta}, x_{\text{min}}\right\} \\
\text{sy}'' &= \min\left\{\left(\text{oy}^{\theta} + \text{sy}^{\theta}\right), y_{\text{lim}}\right\} - \max\left\{\text{oy}^{\theta}, y_{\text{min}}\right\} \\
\text{ox}'' &= \max\left\{\text{ox}^{\theta}, x_{\text{min}}\right\} - \left\lceil\left(\text{XO}_{\text{inst}}^{\theta} - \left(\text{XC}_{\text{inst}} - \frac{\text{XO}_{\text{reg}}}{\text{XS}_{\text{reg}}}\right)\cdot\frac{\text{Wt}_{\text{inst}}^{\theta}}{\text{Ws}_{\text{inst}}}\right)\cdot\frac{\text{fx}^{\theta}}{\text{W}_{\text{comp}}^{\theta}}\right\rceil \\
\text{oy}'' &= \max\left\{\text{oy}^{\theta}, y_{\text{min}}\right\} - \left\lceil\left(\text{YO}_{\text{inst}}^{\theta} - \left(\text{YC}_{\text{inst}} - \frac{\text{YO}_{\text{reg}}}{\text{YS}_{\text{reg}}}\right)\cdot\frac{\text{Ht}_{\text{inst}}^{\theta}}{\text{Hs}_{\text{inst}}}\right)\cdot\frac{\text{fy}^{\theta}}{\text{H}_{\text{comp}}^{\theta}}\right\rceil
\end{aligned}
\qquad \text{(C-3d)}
$$

7) Clause C.4.7

Add "jpxf" context-range type; change the definition of context-range to:

```
context-range = jpxl-context-range / jpxf-context-range / mj2t-context / jpm-
context / reserved-context
```

Add the following definitions to the end of the list:

```
jpxf-context-range = "jpxf" "<" jpx-frame-indices ">" [ "[" jpx-thread "]" ]

jpx-frame-indices = sampled-range

jpx-thread = UINT
```

Replace:

"This Recommendation | International Standard defines three specific types of context-range"

with:

"This Recommendation | International Standard defines four specific types of context-range"

Append the following text at the end of the clause:

A `jpxf-context-range` may be used to compactly identify a range of compositing layers and coordinate remapping transformations which could alternately be identified via a `jpxl-context-range`. The equivalent jpx-layers and jpxl-geometry values may be obtained by expanding composited frames into their constituent JPX compositing layers and compositing instructions in the manner described below.

If the logical target does not contain a JPX Composition box, the server shall ignore any `jpxf-context-range`. Otherwise, the instructions found within the JPX Composition box together describe a sequence of composited frames, as described in Annex M of Rec. ITU-T T.801 | ISO/IEC 15444-2. These composited frames may be numbered f=0, 1, … F_{comp}–1 and are considered to belong to a base presentation thread t=0. If the logical target also contains Composition layer extensions ("jplx") boxes, these boxes may contribute additional presentation threads. As explained in Annex M of Rec. ITU-T T.801 | ISO/IEC 15444-2, a Compositing Layer Extensions box contributes Tjclx presentation threads, each of which has the same number of composited frames, Fjclx, where the values of Tjclx and Fjclx for each Compositing Layer Extensions box are specified by its Compositing Layer Extensions Info sub-box. Together, the collection of all Compositing Layer Extensions boxes in the logical target defines T global presentation threads, where T is the maximum of the associated **Tjclx** values. For each t in the range 1 through T, global presentation thread t consists of the F_{comp} composited frames from the Composition box, followed by the **Fjclx** frames defined by compositing group g = min{t, **Tjclx**} of each successive Compositing Layer Extensions box for which **Tjclx** is non-zero.

If no jpx-thread value is supplied, or jpx-thread is 0, the `jpxf-context-range` includes only those composited frames contributed by the Composition box whose indices f match jpx-frame-indices; there are at most F_{comp} of these. Otherwise, the `jpxf-context-range` includes all composited frames from global presentation thread t = min{T, jpx-thread} whose indices f match jpx-frame-indices.

8) New clause C.7.5 Sendto

Add new clause C.7.5 with the following text:

```
sendto      = "sendto" "=" host ":" port ";" mbw ";" bpc
host        = token
port        = UINT
bpc         = UINT
```

If this request field is present, the server is requested to deliver response data for this request as UDP datagrams to the supplied `host` (name or IP literal), using the supplied `port` number, with a maximum delivery bandwidth of `mbw`, and a maximum of `bpc` bytes in each data chunk, including the 8-byte chunk header. The bandwidth may be expressed in terms of bits/second, kilobits/second, megabits/second, gigabits/second or terabits/second; for a definition of `"mbw"`, see 10.2.4. The `bpc` value shall be no smaller than 32 and no larger than 4096.

This request field may only be used to direct the response data associated with an established "http-udp" transport. Servers shall ignore the request field if the transport type associated with the request is not "http-udp". Otherwise, response data is framed into chunks and delivered via UDP datagrams in the manner described in Annex K. Moreover, in this case, the client shall not send acknowledgement datagrams in response to these delivered chunks, nor should the server expect them.

The effect of this request field is non-persistent; it applies only to the response data associated with the request in which it is found.

NOTE 1 – A request is associated with the "http-udp" transport type in one of two possible circumstances: a) the request contains a "new-channel" request field and the server grants the request with a new channel that uses the "http-udp" transport, as indicated by the `JPIP-cnew` response header; or b) the request specifies a channel-id that has been issued for a channel using the "http-udp" transport and no new JPIP channel is issued by the server in response to this request.

NOTE 2 – Because response data delivered to the address specified by a Sendto request field is not explicitly acknowledged, clients should pay particular attention to the abandon and barrier request fields, which can be used to effect reliable communications. Also, because the server receives no acknowledgement information from which to estimate channel conditions, such as bandwidth and loss probability, it is the client's responsibility to perform whatever estimation may be necessary and supply an appropriate delivery bandwidth and chunk size.

9) New clause C.7.6 Abandon

Add the following new clause C.7.6

```
abandon          = "abandon" "=" 1#chunk-range
chunk-range      = chunk-qid ":" chunk-seq-range
chunk-qid        = UINT
chunk-seq-range  = UINT-RANGE
```

This request field allows the client to explicitly inform the server about the absence of one or more data chunks that may have been sent in response to previous requests. Each occurrence of `chunk-range` informs the server of one or more data chunks that should be considered not to have arrived at the client. The server shall not consider any of the data associated with JPIP messages contained within these identified data chunks received or cached by the client, for the purpose of responding to this request or any subsequent request on this or any other JPIP channel, except in the event that the server receives, or has received, explicit acknowledgement of the arrival of these data chunks via acknowledgement datagrams.

If the request does not specify a channel-id which has been issued for a channel using the HTTP-UDP transport, the client shall not include any Abandon request field and the server shall ignore any such request field that it encounters.

NOTE – The Abandon request field can be used regardless of whether the Sendto request field is present in the same request.

The `chunk-range` values identify data chunks via the 16 low-order bits of the request ID and the chunk sequence number; both of these values are found in the relevant chunk headers, as described in Annex K. The request ID component is identified by `chunk-qid` and matches the contents of the Request ID field in the chunk header the client wants to negatively acknowledge; no `chunk-range` shall have a `chunk-qid` value outside the range 0 to 65535.

The Abandon request field only applies to data chunks which have been transmitted or would be transmitted in response to previous requests within the same channel. To avoid ambiguity, servers shall ignore any Abandon request field which is part of the first request in a new JPIP channel – i.e., the request in which the channel's New Channel request field appears. Also, the Abandon request field does not apply to data chunks belonging to requests that have been excluded by means of a Barrier request field that appeared in a previous request within the channel.

NOTE 1 – It is possible that some of the data chunks affected by an Abandon request field have not been transmitted by the server by the time the request arrives. In this case, the server would typically abandon these data chunks immediately, without even transmitting them a first time. If this behaviour is not desired by the client, the client should avoid abandoning data chunks before at least one later data chunk within the same request or a data chunk from a later request have been received.

NOTE 2 – As explained in Annex B, this Recommendation | International Standard does not require the server to maintain a complete log of data which it has sent in response to client requests; nor does it require the server to exclude such data from its response to future requests. This means that a server may, at its discretion, choose to erase any log entry describing the transmitted chunks at any point. However, if the server does maintain a log of what has been sent to the client, for the purpose of avoiding redundant transmission in the future, it may need to keep track of the contents of data chunks for which it has not yet received acknowledgement information via acknowledgement datagrams or Abandon request fields, so that it can correctly respond to Abandon requests in the future. A server may choose to erase parts of its log at any time so as to reduce the burden of keeping track of unacknowledged data chunks. Alternatively, the client may use Barrier request fields to inform the server that it will never Abandon data chunks sent in response to a certain range of requests, so that the server need not keep track of unacknowledged data chunks belonging to that range.

10) New clause C.7.7 Barrier

Add the following new clause C.7.7

```
barrier          = "barrier" "=" barrier-qid
barrier-qid      = UINT
```

This request field is provided to enable clients to inform servers of the requests for which response data chunks will not be abandoned via any subsequent request. The effect of Barrier request fields persists within the associated JPIP channel. Specifically, the effect of any Abandon request field in any subsequent request is limited to data chunks whose associated request has a request-id Q that is strictly greater than Qb, where Qb is the maximum of all barrier-qid values specified in this or any preceding request within the same JPIP channel.

If the request does not specify a channel-id that has been issued for a channel using the "http-udp" transport, the client shall not include any Abandon request field and the server shall ignore any such request field that it encounters.

NOTE 1 – The Barrier request field only affects the interpretation of Abandon request fields found in subsequent requests. Thus, for example, "barrier=3&abandon=3:4-7" means that the client is abandoning data chunks 4 to 7 from the request with request-id 3, but it will not abandon any data chunks from that request in the future.

NOTE 2 – The `chunk-qid` values supplied in via a `chunk-range` in an Abandon request match any request whose request-id has the same least significant 16 bits as `chunk-qid`. On the other hand, the `barrier-qid` value supplies a full request-id, not just the least significant 16 bits.

11) New clause C.7.8 Timed wait

Add the following new clause C.7.8

```
twait = "twait" "=" max-wait-usecs
max-wait-usecs    = UINT
```

This request field allows the client to suggest the latest point at which it would like the server to start responding to the current request, pre-empting the previous incomplete request, if any, within the same JPIP channel.

If there is no previous request within the JPIP channel this request field shall be disregarded by the server and the request shall be considered not to contain `twait` for the purpose of the ensuing description. If the previous request within the JPIP channel does not contain the `twait` request field, the latest pre-empt time is obtained by adding `max-wait-usecs` microseconds to the time at which the server began to serve that previous request. If one or more immediately preceding requests within the JPIP channel contain the `twait` request field, the latest pre-empt time is obtained by adding the `max-wait-usecs` values of all such requests, as a number of microseconds, to the time at which the server began to serve the most recent request within the channel that did not contain the `twait` request field.

Clients shall not issue requests that contain both the `twait` and `wait` request fields.

NOTE – In applications where animation is involved, clients may find it useful to send a succession of timed-wait requests, so that the server is able to optimize the actual service times to devote to each outstanding request, subject to their respective latest pre-empt times.

12) New clause C.10.4 Handled

Add the following as C.10.4:

```
handled = "handled"
```

If this request field is present, the server shall include a JPIP-handled response header within its response, identifying request fields which the server is prepared to handle.

NOTE – The JPIP-handled response header is defined in D.2.26.

13) Clause D.1.2

Add the following item to the list in D.1.2:

```
/ JPIP-handled                    ; D.2.26
```

14) Clause D.2.3

Replace the last paragraph in Table D.1 ("auxport" meaning paragraph) with the following text:

This parameter is used with transports requiring a second physical channel. If the "http-tcp" or "http-udp" transports are used, the auxiliary port is used to connect the auxiliary channel. For further details, see Annexes G and K. The parameter need not be returned if the original request involved a channel that also employed an auxiliary channel, having the same auxiliary port number. Otherwise, the parameter need be returned only if the auxiliary port number differs from the default value associated with the selected transport.

15) Clause D.2.9

Replace the client request in example at the end of D.2.9 with:

stream=0&context=jpxl<2-7:2>[s0i0],jpxl<9-10>[s1i3]

16) New clause D.2.26 Handled request (handled)

Add D.2.26 containing the following:

```
JPIP-handled  = "JPIP-handled" ":" LWSP 1#handled-req
handled-req   = (request-field | partially-handled-req)
partially-handled-req = request-field "=" handled-req-option
request-field        = TOKEN
handled-req-option   = TOKEN
```

The server shall include this response header in its response to a request containing the `handled` request field. This `JPIP-handled` response header identifies the requests which the server is able to handle correctly, in accordance with this Recommendation | International Standard. Each `request-field` may be any of the request fields mentioned in C.1.2, but may also include other tokens that some clients might not recognize; clients shall ignore any request-field they do not understand.

A `partially-handled-req` may be used to indicate partial support for a request field. If the relevant request field has a finite set of possible complete parameter strings following the `"="` character (e.g., `"yes"` or `"no"`), the `handled-req-option` may be one of those values. Table D.3 describes additional values for the `handled-req-option` which are defined by this Recommendation | International Standard for use with specific request fields. Servers may include other tokens for the `handled-req-option` that some clients might not recognize. Clients shall ignore any `partially-handled-req` whose `request-field` or `handled-req-option` they do not understand.

Table D.3 – Additional `handled-req-option` values for particular request fields

request-field	handled-req-option	Meaning
Cnew	transport-name	The server correctly handles new-channel request fields that contain the indicated transport type.
Context	"jplx", "mj2t", "jpmp", "jpxf"	The server correctly handles codestream context request fields for context-range values that commence with the handled-req-option token.

17) Clause G.1

Replace the first paragraph with the following text:

The JPIP protocol itself is neutral with respect to underlying transport mechanisms for the client requests and server responses, except in regard to channel requests represented by the New Channel ("cnew") request field (see C.3.3) and the New Channel ("JPIP-cnew") response header (see D.2.3), where transport-specific details shall be communicated. This Recommendation | International Standard defines three specific transports, which are identified by the strings "http", "http-tcp" and "http-udp" in the value string associated with New Channel requests. This annex provides details of the second transport, which shall be identified in this text as HTTP-TCP. The first transport is identified in this text as HTTP and is described in Annex F. The third transport type is identified in this text as HTTP-UDP and is defined in Annex K.

18) Clause H.1

Replace the fourth paragraph and following text in this clause with the following:

Finally, it is assumed that each logical connection provides one of the following two types of services:

a) A reliable stream-oriented service, such as that offered by TCP.

b) An unreliable packet-oriented service (for example, see "http-udp" in Annex K). In this case, packets may arrive out of order or not at all.

19) Clause H.2

a) Replace the first two paragraphs with the following:

In this clause, the request connection is reliable, meaning that requests arrive at the server in order without loss, and server responses are received by the client in order and again without loss. In this case, the request fields and response headers may be communicated exactly as in the "http-tcp" protocol, and indeed HTTP is recommended for the transport of requests and response headers.

The JPIP stream messages, including the EOR message (see D.3), shall be partitioned into packets and delivered over the unreliable data connection.

b) Delete the remaining parts of this clause – no longer required.

20) New Annex K

Add the following as new Annex K and rename current Annexes K through N to Annexes L through O:

Annex K

Using JPIP with HTTP requests and UDP returns

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

K.1 Introduction

This annex provides details of the "http-udp" transport, which is identified in this text as HTTP-UDP. The HTTP-UDP transport uses the same mechanisms as the HTTP transport to send client requests to the server and receive the server's response headers and status codes. However, the server's response data are delivered as UDP datagrams over an auxiliary UDP connection. The information transported on this auxiliary UDP connection is identical to that which would have been transported as the entity body of a pure HTTP response, except that it is framed into chunks, each of which has a chunk sequence number and a record of the Request-id associated with the corresponding client request. Each chunk shall contain a whole number of JPIP messages and at most one EOR message as defined in Annex A. Message class identifiers and codestream sequence numbers shall be present in at least the first JPIP message of each data chunk.

NOTE – Since the UDP transport is not reliable (i.e., UDP packets might be dropped or out of order) clients may not receive all data chunks corresponding to a request. Clients may use the Abandon request field to explicitly inform the server of this condition, see C.7.6.

K.2 Client requests

Requests are delivered on the primary channel exactly as HTTP requests. They have exactly the same form as requests issued over a channel that uses the HTTP transport described in Annex F. In particular, HTTP "GET" and "POST" requests may both be used. Client requests that are issued within an HTTP-UDP transported JPIP channel shall include the Request-id (qid) request field.

NOTE – Clients should issue requests with consecutive Request-id values.

K.3 Response data delivery and channel establishment

A new channel may be established to a JPIP server by issuing a request that includes the New Channel request field (see C.3.3). As an example, such a request might be issued using HTTP, although it might also be issued to a JPIP-specific server using any suitable transport mechanism. If the server's response (through the New Channel response header in D.2.3) indicates that a new channel has been created to work with the HTTP-UDP transport, the request which included the New Channel request field is treated as though it had been issued within the newly created HTTP-UDP transported channel. This ensures that the response data for that request and all subsequent requests in the same channel is framed into data chunks and delivered as UDP datagrams. This response data cannot be empty, since every request issued within an HTTP-UDP transported channel shall have a response data stream that consists of at least the EOR message (see D.3).

The destination to which response datagrams are delivered depends upon whether or not the associated request contains a Sendto request field.

1) For requests that contain a Sendto request field, the datagrams is delivered to the specified address without any explicit acknowledgement by the client.
2) For requests that do not contain a Sendto request field, the response datagrams cannot be delivered until an auxiliary UDP connection has been established. To do this, the client sends one or more connection establishment datagrams to the server host identified via the New Channel response header, on the port identified by the New Channel response header. Each connection establishment datagram commences with a four byte header, which is followed by the channel-id string associated with the new HTTP-UDP channel. Once the server receives a connection establishment datagram with the correct channel-id string, it sends all subsequent response datagrams (other than those associated with Sendto request fields) to the

IP address and port from which the channel establishment datagram arrived and it expects to receive acknowledgement datagrams from the same client IP address and port.

NOTE – Since UDP is an unreliable transport, connection establishment datagrams might be lost. For this reason, clients should be prepared to send multiple connection establishment datagrams if necessary, and servers should be prepared to discard superfluous connection establishment datagrams which may arrive. Once a valid connection establishment datagram has been received, the server may choose to filter incoming datagrams as part of an overall defence strategy.

The first two bytes of a connection establishment datagram's header shall be FF, while the third and fourth bytes identify the length of the channel-id string, recorded as a 16-bit big-endian quantity. The four byte header is followed by the channel-id string, encoded as UTF-8 characters. Additional content may be provided, beyond the end of the channel-id string, but the interpretation of such content is unspecified by this Recommendation | International Standard.

K.4 Server responses

In response to each client request, the server sends an HTTP reply paragraph back to the client over the primary channel. The reply paragraph contains the status code, reason phrase and all relevant JPIP response headers and any appropriate HTTP response headers. However, no response data is returned via the primary channel. For this reason, there shall be no HTTP entity body in an HTTP-UDP response. Neither shall the "Content-length:" or the "Transfer-encoding:" HTTP response headers be used.

The response data itself are delivered via UDP, framed into chunks in the manner described in K.5. Since the HTTP-UDP transport may be used only with sessions, the image return type is constrained to JPP-stream and JPT-stream as defined in Annex A. Thus, the response data invariably consists of a sequence of JPP-stream or JPT-stream messages.

The response data resulting from each request shall consist of a whole number of chunks, meaning that no chunk may contain response data generated in response to two different requests.

The response to each and every request following the one in which the channel was requested, shall be terminated with an EOR message, even if the response data would otherwise have been empty. The EOR message is considered as part of the response data.

This means that every request issued on an HTTP-UDP transported JPIP channel results in the generation of at least one non-empty response chunk from the server and that the last chunk generated in response to each request terminates with the EOR message.

K.5 Framing of response data into chunks

All response data sent by the server via the auxiliary UDP connection shall be framed into chunks. Each chunk consists of an 8-byte chunk header, followed by the chunk body that holds the server's response data, as shown in Figure K.1. The chunk header and body are sent as a single UDP datagram, whose length shall be exactly 8 plus the length of the chunk body measured in bytes. Moreover, no UDP datagram shall have a length larger than 4096 bytes or a length smaller than 8 bytes. The first 2-byte word of the chunk header holds an unsigned big-endian integer, whose interpretation as a "control" field is provided in Table K.1.

The remaining 6 bytes of the chunk header contain the least significant 16 bits of the Request ID provided by the client in the request with which the chunk's response data is associated, together with a one-byte "repeat" field and a 24-bit chunk sequence number encoded as big-endian unsigned integer. The chunk sequence number is a number generated by the server, shall start at zero and shall be incremented by one for each subsequent chunk sent to the client in response to the same request.

New requests from the client shall cause the chunk sequence numbering to be reset starting at 0 for the first chunk sent in response to the new request. Chunk sequence numbers do not wrap around, and servers shall indicate an error using the EOR reason code 7 (Response limit reached), see D.3 in the event of running out of available chunk sequence numbers.

The one-byte "repeat" field may be used by the server in any manner desired. Typically, the "repeat" field would be used to distinguish between original and retransmitted versions of a data chunk, allowing the server to determine which instance of a chunk is being acknowledged within an acknowledgement datagram. However, the field may potentially be used in other ways. Clients should not attempt to interpret the "repeat" field but shall reproduce it within returned acknowledgement datagrams.

NOTE – The Request ID and chunk sequence number allow for chunks of data to be properly reassembled in order. They also provide a means for dropped chunk detection. Clients may detect the loss of chunks by examining the set of chunk sequence numbers for gaps or by detecting that the server has not transmitted a chunk containing an EOR message; the latter will always be included in the last chunk sent in response to a request. Clients may use the Abandon request field to explicitly inform the server of missing chunks. Clients may choose to defer the use of these mechanisms or not to use them at all, at their discretion.

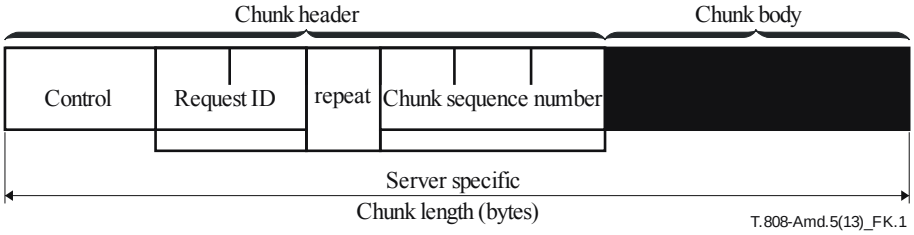

Figure K.1 – Response data structure on http-udp connection

Table K.1 – Interpretation of the "control" field in each data chunk header

"control" field value	Interpretation in response chunk headers	Interpretation in acknowledgement chunk headers
`0000 xxxx DDDD DDDD`	Maximum time that the server would prefer the client to wait from the time it receives this data chunk before acknowledging the chunk's arrival in an acknowledgement datagram for the first time is given by $2^{(D/8)}$ microseconds, where D is the unsigned integer represented by the second byte of the control field. This interpretation does not apply if the corresponding request included a Sendto request field with the value "no".	Estimated time between client receipt of the data chunk and delivery of the acknowledgement datagram, expressed as $2^{(D/8)}$ microseconds.
`0000 AAAA xxxx xxxx`	Acknowledgement repetitions A, in the range 0 to 15. The server would prefer the client to acknowledge the arrival of this data chunk at least A+1 times, in A+1 separate acknowledgement datagrams. The server would prefer all A+1 acknowledgement datagrams to be sent within the time given by the D parameter, as defined above. If the corresponding request included a Sendto request field, the value of A has no meaning, since the client sends no acknowledgement datagrams in that case.	Number of previous datagrams in which the client has already acknowledged the receipt of this data chunk.
`1111 1111 1111 1111`	Reserved for ITU/ISO use	Connection establishment datagram
`Other values`	Reserved for ITU/ISO use	Reserved for ITU/ISO use

K.6 Client acknowledgement of server responses

For response datagrams issued in response to requests containing a Sendto request field, there is no explicit client acknowledgement, although servers might deduce the arrival or non-arrival of data chunks using information provided via Barrier and/or Abandon request fields in subsequent requests.

In all other cases, clients are expected to acknowledge the successful arrival of data chunks by sending acknowledgement datagrams back to the server. UDP acknowledgement datagrams are sent to the same host address and port as the connection establishment datagram. Moreover, clients shall send acknowledgement datagrams from a socket bound to the same local address and port as that used to send the connection establishment datagram. Each acknowledgement datagram consists of one or more chunk headers from received data chunks, except that the "control" field is modified as follows. The 8-bit D value in the returned chunk header's "control" field should be modified to reflect (at least approximately) the number of microseconds between client receipt of the data chunk and delivery of the acknowledgement datagram, expressed as $2^{(D/8)}$ microseconds. The 4-bit A value in the returned chunk header's "control" field should be modified to reflect the number of previous datagrams in which the client has already acknowledged the receipt of the data chunk in question. This information is reflected in Table K.1. No acknowledgement datagram shall be more than 512 bytes. That is, no acknowledgement datagram shall contain more than 64 chunk headers. Connection establishment datagrams are distinguished from acknowledgement datagrams on the basis of the first 4 bits of the control field, as shown in Table K.1.

Even if the client has already identified a data chunk via an Abandon request field, if that chunk is subsequently received, the client may acknowledge its arrival via an acknowledgement datagram; this is generally advisable as it may reduce redundant transmission of information from the server. In this event, however, the client is expected to update its

cache accordingly. That is, the client shall not acknowledge data chunks which it discards, since then the server's log of what the client should have received may contain erroneous entries.

Clients may acknowledge the same data chunk multiple times in separate acknowledgement datagrams; in the event of multiple acknowledgements, the modified D and A values will generally be different. Clients need not send a separate acknowledgement datagram for each received data chunk, but they should endeavour to acknowledge data chunks within the period specified by the D value in the chunk header's "control" field.

NOTE 1 – Acknowledgement may be used by servers for flow-control purposes. Note further that once a data chunk has been acknowledged by a client, subsequent Abandon request fields that refer to already acknowledged chunks may be ignored by the server. In some server implementations, this means that the receipt of acknowledgement information allows the server to release temporary storage resources that may be needed to maintain cache model consistency.

NOTE 2 – Notwithstanding the guidelines presented above, it is acceptable for a client to promptly return a single acknowledgement datagram for each received data chunk, containing just the one chunk header, with the "control" field set to 0. The D values are intended primarily to allow server flow control algorithms to take into account the additional delay which may be incurred where a client chooses to aggregate data chunk acknowledgements into a smaller number of acknowledgement datagrams. The A values are intended to allow a server to loss probability for acknowledgement datagrams and feed suggested repetition counts to the client to increase the robustness of the acknowledgement mechanism.

K.7 UDP and Maximum Response Length Field (informative)

There may be little or no reason for using the Maximum Response Length field with a UDP return channel, unless the Sendto request field is also being used. Apart from this case, the server should be able to use the times at which acknowledgement datagrams arrive to regulate the flow of response data to the client, so as to maintain responsiveness. If the Sendto request field is used, however, the server does not receive continuous feedback from the client and may easily push a great deal of data over the channel. To maintain responsiveness or avoid excessive loss in these circumstances, clients should use the Maximum Response Length field (see C.6.1) to regulate the flow of traffic, much as they would with the HTTP transport.

K.8 Implementation strategies for acknowledged communication (informative)

Response data chunks delivered in response to requests that do not contain the Sendto request field are acknowledged via explicit acknowledgment datagrams. This model is quite common in network communication protocols and facilitates the implementation of flow control management within the server. Although not required by this Recommendation | International Standard, servers are recommended to adopt a retransmission strategy, in which data chunks that have not been acknowledged after an appropriate period of time are retransmitted, unless the server is able to determine that the data chunks are no longer relevant to the client – e.g., due to a change in the client's window of interest. Typically, the retransmission would stop, once a data chunk is either acknowledged or abandoned by means of an Abandon request field.

Clients cannot expect servers to retransmit data chunks that are not acknowledged. Moreover, the client cannot generally have any guarantee of the point at which a server may decide that an unacknowledged data chunk has not arrived at the client. This is important, since it may affect the interpretation of responses to future requests. For example, the response to a future request may include an EOR message with the "Window Done" reason code, yet the client cannot be sure that it has indeed received all relevant content if an earlier request with overlapping Window of Interest still has outstanding missing data chunks. To avoid the ambiguity which might be created by such situations, clients can use the Abandon request field to explicitly abandon missing data chunks from requests that have not received any content for some time. This allows the client to be sure that the server will eventually include any relevant content that was missing from those requests in its response to future requests.

If a client does not need to rely upon the reason codes supplied by EOR messages (e.g., because the client can directly compute whether its cache contents contain a complete response to a requested Window of Interest), there may be no need for it to consider use of the Abandon request field.

Clients should bear in mind that aggressive use of the Abandon request field may significantly increase server workload. For example, a typical server may generate data chunks in batches which are then scheduled for transmission. If a client automatically issues Abandon request fields referring to all previous requests whenever its Window of Interest changes in any way, the server may discard many of the data chunks it has generated without transmitting them at all. Although this does not break interoperability, the server may have to regenerate content many times over during an interactive session. To avoid this problem, it is recommended that clients wait until at least one data chunk is received from a request A before issuing an Abandon request that refers to data chunks from an earlier request B, unless no data chunks are received at all for a considerable period of time. It is generally possible for the server to make better decisions than the client regarding data chunks which should not be transmitted because the client's Window of Interest has changed.

The Barrier request field is not primarily intended for use with acknowledged communication. Nevertheless, if this request field is used by a client, it provides a supplementary mechanism to implicitly acknowledge the arrival of data chunks that have not been abandoned – it may be that acknowledgement datagrams for some of these have been lost. Ignoring a Barrier request field does not damage interoperability in JPIP but may lead to some redundant transmission, so servers are recommended to implement this feature.

K.9 Implementation strategies for unacknowledged communication (informative)

Response data chunks delivered in response to requests that contain the Sendto request field are not acknowledged via acknowledgement datagrams. As with all JPIP communications, the server may assume that data which it has sent to the client arrive and are cached by the client, unless and until it learns otherwise. Without this assumption, the server would find itself transmitting the same content over and over again in a typical interactive session. Since UDP is an unreliable transport medium, the server must be prepared for the possibility that data chunks are actually lost. In particular, it must be prepared to process Abandon request fields.

A typical server would keep a record of the data-bin byte ranges that are contained within each data chunk that has been sent to the client. If the data chunk is abandoned, the record can be erased and the server should remove the relevant data-bin byte ranges from its log of content the client has received (i.e., its client cache model); this content may be delivered again in response to future requests, if deemed relevant. Since Abandon requests provide only a mechanism for the client to declare that it has not received some data chunks, if no other steps are taken by the client, the servers record of data chunks that have been sent and not yet abandoned could grow indefinitely; as a result, a typical server would eventually need to internally abandon data chunks, so that everything gets abandoned in the long run. This would eventually cause a great deal of redundant transmission, but clients can avoid the problem using one of the following two strategies:

1) A client implementation can arrange to send additive cache model manipulation statements that directly add all received data-bin byte ranges to the server's cache model. This avoids redundant computation, so long as servers do not accidentally erase this information again while abandoning the relevant data chunks at a later point. To avoid such possibilities and reduce the burden on servers, clients are strongly recommended to use the Abandon and Model request fields together, to declare all data chunks from a given previous request abandoned but simultaneously add all the data-bin byte ranges from the arrived data chunks to the server's cache model via the Model request field. More generally, it is preferable for clients to abandon data chunks explicitly rather than leave servers to do so implicitly at an undefined point in the future; and it is preferable for clients to abandon data chunks during the same or an earlier request to that in which the arrived content is identified via the Model request field. Keeping this in mind, it is preferable for servers to process the effects of an Abandon request field prior to the processing of any Model request field found in the same request.
2) A client implementation can use the Barrier request field to assure the server that no subsequent request will contain an Abandon request field which abandons data chunks belonging to requests prior to a certain point. This effectively acknowledges the successful arrival of all data chunks not yet abandoned from requests prior to the request-id specified by Barrier. To help conserve server bookkeeping resources, clients are recommended to use Barrier regularly. A typical client implementation might abandon all non-arrived data chunks associated with a request which is sufficiently old and simultaneously use the Barrier request field to indicate to the server that there will be no further abandonment for data chunks from that request.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2013