INTERNATIONAL STANDARD ISO/IEC 15444-9:2005
TECHNICAL CORRIGENDUM 2

Published 2008-06-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

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

TECHNICAL CORRIGENDUM 2

Technologies de l'information — Système de codage d'images JPEG 2000: Outils d'interactivité, interfaces de programmes d'application et protocoles

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 15444-9:2005 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 (2005)/Cor.2.*

ICS 35.040 **Ref. No. ISO/IEC 15444-9:2005/Cor.2:2008(E)**

© ISO/IEC 2008 – All rights reserved

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD
ISO/IEC 15444-9:2005/COR 2:200X (E)
ITU-T Rec. T.808 (2005)/Cor.2 (200X E)
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system:

Interactivity tools, APIs and protocols

Technical Corrigendum 2

1) Annex C.5.2
Replace the entire body of C.5.2 by the following text:

C.5.2 Metadata Request (metareq)

C.5.2.1 Description

```
metareq = "metareq" "=" 1#("[" 1$(req-box-prop) "]" [root-bin] [max-depth])
                        [metadata-only]
req-box-prop = box-type [limit] [metareq-qualifier] [priority]
limit = ":" (UINT / "r")
metareq-qualifier = "/" 1*("w" / "s" / "g" / "a")
priority = "!"
root-bin = "R" UINT
max-depth = "D" UINT
metadata-only = "!!"
```

This field specifies what metadata is desired in response to a request, in addition to any metadata required for the client to decode or interpret the requested image data as defined by C.5.1. The purpose of this request is to allow the client to request selected parts of the contents and the layout of the meta data encoded in the JP2 and JPX file formats a server did not choose to transmit according to C.5.1.

The value string in this request field is a list of independent requests; however, the server may handle the requests as a group, and there may be overlap between the requests. It is then sufficient (but not necessary) that the server sends the requested data only once.

The way how the server decides to break up the initial stream into bins is irrelevant for defining the target of the request except that the `root-bin` field can be used to limit a request to parts of the file structure once a client has identified the layout. Once a request is confined to a specific bin, the way how that bin is broken up into more bins – or if it is broken up at all – is irrelevant for the client and the way how that data is addressed within the request.

Note, however, that data a server returns upon a request will in general depend on that layout because the division of the logical target into metadata-bins may force the server to return additional data, including the contents or headers of some other, potentially un-requested boxes. All a server has to ensure that **at least** the requested data is contained, and that **enough** data is returned to allow a client to parse it. Examples when additional data must be returned are given below in C.5.2.7. The following text uses the wording “request” to point out which data is **desired** by the client, which might be a sub-set of the data actually returned by the server due to reasons pointed out in C.5.2.7.

Example:

For better illustration, examples in the following subsections all refer to the following segment of a JPX file, see ITU-T Rec. T.801 | ISO/IEC 15444-2 for the definition of the boxes used here. The labels on the right hand side have been added for later reference:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Content	Label
association box header ('asoc')	A
number list box header ('nlst')	B
number list box content	C
association box header ('asoc')	D
ROI description box header ('roid')	E
ROI description box content	F
association box header ('asoc')	G
label box header ('lbl\040')	H
label box content	I
XML box header ('xml\040')	J
XML box content	K

The subbox structure of the above example is indicated by indention, e.g. items H to K establish the contents of the superbox at label G.

C.5.2.2 root-bin

Each request is relative to the data-bin specified by its root-bin value. If a root-bin value is not specified, the root is meta-data bin 0. The request pertains only to data within or referenced by that particular data-bin.

Example

If the server decided to place the contents of association box 'A' in the above example into a separate bin with bin id #3, the association box header 'A' would be encoded in a placeholder box, and items 'B' to 'K' would establish bin #3. In that case, a root-bin field of 3 would limit the scan to items 'B' to 'K' only. Specifically, `metareq=[roid]R3` would request items 'E' and 'F' from the server and no other data outside of this example (but see also C.5.2.3 and C.5.2.7 for additional data outside of the request potentially returned by the server along).

An alternative layout might be to include items 'B' to 'G' in bin #3 as above, but in addition place items 'H' to 'K' into the separate bin #4. Thus 'G' would be represented by a placeholder box in bin #3 and 'H' to 'K' would be part of bin #4. A root-bin field of 3 would still scan the items 'H' to 'K' because they are referenced by a placeholder in bin #3 and the way how bin #3 is broken up into sub-bins is irrelevant to the request. Thus, even though the server response would be different, the items identified by the request remain the same.

A root-bin field of 0 imposes no further restriction on the request each item, box or superbox, is somehow reachable from the meta data bin #0. Whether placeholder boxes are used or not is completely irrelevant. Thus, `metareq=[roid]` would request all ROI description boxes from the server, and thus also include items 'E' and 'F' along with all other ROI description boxes available.

C.5.2.3 max-depth

If a value for `max-depth` is specified, then only boxes contained within the root metadata-bin, and those no more than `max-depth` levels in the file hierarchy below that box are requested. If a value for `max-depth` is not specified, there is no limit on the depth of the file hierarchy for this request.

Example:

If items 'B' to 'K' establish the contents of meta data bin #3 as in the example for C.5.2.1, the root-bin field is set to 3 and max-depth is set to 0, then the request is limited to items 'B' to 'D'. If max-depth is set to 1, the request is limited to items 'B' to 'G'.

The request `metareq=[roid]R3D0` would therefore not request any data from the server because the only ROI description box within the specified bin is one level below the start of bin #3. The request `metareq=[asoc]R3D0` would request the association box starting at label 'D' and its contents, items 'E' to 'K'. This request is identical to `metareq=[asoc]R3D3` because even though the latter example also requests the association box starting at label 'G', this box is part of the box starting at label 'D' and is thus included in the former request anyhow.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.5.2.4 req-box-prop

The `req-box-prop` portion of the request specifies a list of box types that are of interest to the client. The special string "*" may be substituted for the box type, in which case all box types are implied. Thus, this field confines the request to apply only to the specific box type (or types) listed and instructs the server to deliver the box header and box contents of all matching boxes within all additional constraints.

The contents of a superbox is defined by its complete sub-box hierarchy. This implies that in case superboxes match the box type, the complete sub-box hierarchy of the matching superbox is requested, regardless of the max-depth field.

Example:

Consider again the example layout of C.5.2. Then, a `req-box-prop` of type `'asoc'` would include all items 'A' to 'K' in the request because they establish the content of the matching box defined at label 'A'. Note that once the association box at label 'A' has been identified to match the request, the depth limit does not limit the delivery of its contents. A `req-box-prop` of type `'lbl\040'` would only include items 'H' and 'I', along with all other label boxes, provided they match all other specifications of the request, e.g. are contained in the addressed root bin above the depth-limit.

The request `metareq=[*]R3D0` instructs the server to return the entire contents of all boxes it finds in the contents of bin 3, and thus requests items 'B' to 'K'. While a restriction on the desired depth has been specified, the server shall ignore that restriction because items 'E' to 'K' are part of the box starting at label 'D' and no other constraints apply.

C.5.2.5 limit

The limit attribute optionally following the `req-box-prop` field further confines the type of request, and how many bytes of the box contents identified by the `req-box-prop` field the client is interested in. The limit parameter takes the form of a colon followed by either an unsigned integer (the limit value) or the character "r". The same limit value applies to all boxes that match the `req-box-prop` of which it is an attribute. If it is not present, the client is requesting all matching boxes entirely.

If the limit field is an integer n greater than zero, then the server is requested to return the unlimited box header and only the first n bytes of the box content of the matching boxes. The byte count is here defined to count the data as it appeared in the original file before it was broken up into bins.

Furthermore, if `req-box-prop` matches any superboxes, the contents of a superbox is to be understood as the complete and unlimited sequence of box headers and box contents contained within that superbox, and the byte limit by that also counts the box headers of all boxes contained within the matching super box. It may thus happen that the limit field instructs the server to deliver only parts of a (sub-)box header even though the full header of the matching box itself is always included. However, using the limit field in this way is discouraged and should be avoided.

If the limit field is zero, only the box headers of the matching boxes are requested.

If the limit value is "r", then the server is requested to deliver the minimum data required to reconstruct the box-headers of all matching boxes as well as the minimum data to reconstruct the box-headers of all of its descendant sub-boxes up to the maximum depth specified, regardless of their box-type. Note that as an exception `max-depth` does apply for the limit value "r" to limit the contents of superboxes, which makes this type of request special as far as the interpretation of `max-depth` is concerned.

Example:

Consider again a file layout as in C.5.2 above with items 'B' to 'K' in data bin #3 and the meta data request `metareq=[asoc:8]R3D1`. By that the client requires the box header and the first eight bytes of every association box found in bin #3 not deeper than one level from bin #3. In the example at hand, this requests the item 'D', eight bytes from item 'E', namely the part of the first sub-box of 'D' that fits into the limit because it establishes the contents of 'D', the item 'G' because it is exactly one level below the first item 'B' of the bin, and eight bytes of the box content contained in the superbox starting at 'G', that is the first eight bytes of the box header 'H'. Should the box headers 'E' and 'H' not fit into eight bytes, they might get truncated. This is why usage of the numerical limit field on superboxes is discouraged.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Consider the request `metareq=[roid:1]R3D1`. This will request the box header of the ROI description box at label 'E' one level below the start of the bin, and in addition the first byte of its contents at point 'F', which happens to be the number of regions encoded in the box (see ITU-T Rec. T.801 | ISO/IEC 15444-2). If the example would contain a ROI description box at a deeper level, it would not be requested here due to the depth limit.

The request `metareq=[asoc]R3D0` does not contain a limit and thus requests the complete body of any association box found at the box level of meta data bin #3. Even though the association box at label 'G' is outside the depth-limit, it is still requested because it is contained in the association box started at point 'D', and by that items 'D' to 'K' are transmitted completely.

The "r" limit is in effect a request for a skeleton of a portion of the box hierarchy because it only supplies the minimum data, namely the box headers, to reconstruct the structure of the boxes. The request `metareq=[asoc:r]R3D1` thus requests the items at label 'D', 'E' and 'G', but not 'H' and 'J' because they are outside the depth limit. Item 'A' is not part of bin #3 in the example setup and is thus neither requested.

The difference between the limit "0" and the limit "r" is that the former only delivers the box header of all matching boxes, but not necessarily their depending subboxes. The "r" limit, however, extends the request to the box-headers of the sub-structure of the matching box up to the depth-limit.

C.5.2.6 metareq-qualifier:

The "metareq-qualifier" takes the form of a "/" followed by on or more of the flags "g", "s", "w" and "a". Each flag identifies a context from which boxes which match the request shall be drawn. Thus, the "metareq-qualifier" defines an additional constraint on the boxes besides the box-type. The interpretation for each of these contexts is supplied in Table C-2. If more than one of the flags is provided, the union of the corresponding contexts shall be taken.

The contexts "g", "s" and "w" are mutually exclusive, but their union is generally smaller than the catch-call context "a". It is at the discretion of the server to decide which box falls into which context, and no classification of box types is defined in this recommendation/international standard.

C.5.2.7 priority:

If the "priority" flag is specified, then the client is requesting that the data collected by the meta-data request has to be transmitted with higher priority (i.e. upfront) the image data described by other fields of the same request.

C.5.2.8 metadata-only:

If "metadata-only" is specified at the end of the meta-data request field, the client is requesting that the server's response consist only of meta-data, without any image data or codestream headers, regardless of whether view-window request fields such as Frame Size have been used. For JPP-stream and JPT-stream return types, this means that the returned JPIP messages will all be metadata-bin messages. This field also disables the request of the silently implied meta-data defined by C.5.1.

C.5.2.9 Implications of Layout-constraints

Regardless of the box specifications provided via the Metadata Request field, the server may send other data, either because it has determined that the other data is required for the client to decode or interpret the requested image data, or because the server had previously divided the logical target into data-bins using different criteria and additional data shall be sent in order to provide a consistent and meaningful view of the metadata-bins for this logical target.

To make the delivered data parseable to a client, all data from the start of the bin up to the last byte required to satisfy the request has to be known by the client, and thus has to be transmitted provided it is not already within the cache of the client. In addition, should any data that matches the request be relocated into additional bins by means of placeholder boxes, the complete placeholder box and the bytes of the bin referenced by the placeholder box has to be included in the request. Byte counts, as used by the limit attribute, always count bytes **as found in the original stream** and not as it was broken up by the server. This means that the number of bytes actually being transmitted back to the client might be different from the number of bytes implied by the byte-limit because not only placeholder boxes have to

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

be transmitted, but also the data in front of the requested bytes within the bin the bytes are located in might have to be included to make the resulting stream parseable.

Regardless of the box specifications provided via the metareq request field, the server may send other data, either because it has determined that the other data is required for the client to decode.

Example:

Consider again the data as found in the beginning of C.5.2. and assume the server decided to place all of the data into metadata-bin #3 without making use of any (additional) placeholder boxes. Also assume that the cache of the client is empty. Then the metadata-request `metareq=[xml\040:r]R3` is requesting only the box header of the XML box at label 'J'. However, since the bin isn't broken up into more bins, all bytes in front of item 'J' are also required by the client to parse this data successfully and to identify the transmitted data as a box header, and thus the server is required to send all data from 'B' to 'J'.

As the above example suggests, not using placeholders might be considerably inefficient for some requests. The following alternative layout at the server side provides a more efficient access to the same data:
The association boxes at 'D' and 'G' are broken up into separate bins with the bin-ids #4 and #5, respectively. Then for the very same request, the server would have to transmit the placeholder box for item 'D' in bin #3, the placeholder box for item 'G' in bin #4 and the requested box header 'J' now located at the start of bin #5. Note that the request automatically pertains to bins #4 and #5 since they are referenced by placeholders in bins #3 and #4 respectively. Depending on the size of the remaining boxes, this layout might be considerably more efficient.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale