<table>
<thead>
<tr>
<th>Status</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date of Last Update</td>
<td>2018/06/04</td>
</tr>
<tr>
<td>Person Assigned</td>
<td>David Clunie</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:dclunie@dclunie.com">mailto:dclunie@dclunie.com</a></td>
</tr>
<tr>
<td>Submitter Name</td>
<td>Rick Busbridge</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:Rick.Busbridge@philips.com">mailto:Rick.Busbridge@philips.com</a></td>
</tr>
<tr>
<td>Submission Date</td>
<td>2018/04/05</td>
</tr>
</tbody>
</table>

Correction Number CP-1818

Log Summary: Large compressed images may have more frames than fit in the Basic Offset Table

Name of Standard

PS3.3, PS3.5
Large compressed images may have more frames than fit in the Basic Offset Table.
Rationale for Correction:

The Basic Offset Table (first item of encapsulated Pixel Data) uses 32 bit unsigned offsets and these can be exceeded with a large image.

It is optional, so does not prevent encoding such an image, but it is more useful as the image grows larger, since otherwise something similar needs to be recalculated by the recipient to get access to random frames.

The work around of using Concatenations is not always desirable.

Potential solutions include:

- Using the same Transfer Syntaxes but doubling the length of the Basic Offset Table since knowing the number of frames a recipient can determine, if the table contains 32-bit or 64-bit pointers. But how would existing recipients know to check for this - do they check the length of the Item for sanity, and if not sane do they reject the image or ignore the offset table? This might break existing implementations.

- Using the same Transfer Syntaxes but using a special value in the first and second Basic Offset Table that would be invalid or unrealistic as a flag to indicate that the Basic Offset Table uses 64 bit entries subsequently, e.g., a pair of 0xFFFFFFFF entries as a flag. But how would existing recipients know to check for this - do they check the first value or the length of the Item for sanity, and if not sane do they reject the image or ignore the offset table? This might break existing implementations.

- Adding new Transfer Syntaxes that define different encapsulation rules, e.g., a Basic Offset Table with 64 bit entries, or chained Offset Tables with 32 bit entries that were interspersed in the middle of the sequence of compressed fragments. New Transfer Syntaxes might also consider fixing the number of fragments per frame to 1, since fragmentation is probably not useful in these use cases. This approach would mean that recipients would have to add support for the new Transfer Syntaxes if they want to assure receiving offset tables (assuming the sender bothered to create them).

- Adding a new Attribute to the top level data set that contains the offset information and requiring that the BOT in the first item of Pixel Data be empty; this would require addition of a new VR for 64 bit unsigned integers; it also raises the question of invalidation of the offsets in the Attribute during transfer/copying of the files that involves Transfer Syntax changes or reorganization of the fragmented frames, as well as whether or not to remove the (irrelevant and relatively large) attribute when retrieving metadata only (e.g., with WADO-RS). This approach would not break existing implementations existing systems, but would mean that recipients would have to add support for the new Attribute if they want to assure receiving offset tables (assuming the sender bothered to create them).

- Requiring support for Concatenations, since each Instance of a Concatenation can be limited in size so as not to exceed 32 bit offsets to frames. This approach would mean that recipients would have to add support for Concatenations if they want to assure receiving offset tables (assuming the sender bothered to create them).

The approach of adding a new Attribute is described here, and depends on a separate CP that adds an OV (64 bit unsigned) VR.

[Ed.Note.: OV rather than UV is chosen somewhat arbitrarily.]

[Ed.Note.: Open Issue: Should the offset information be added before the encoded pixel data, as is currently the case with the Basic Offset Table, where it is arguably more use to recipientss, or after the encoded pixel data (e.g., with a Data Element Tag greater than (7FE0.0010), which facilitates creation of the object in a streaming manner, i.e., tracking the offsets and adding them at the end. Since Number of Frames has to be encoded earlier, which precludes unbounded streaming use cases anyway, the former approach has been chosen here.]

[Ed.Note.: Open Issue: The opportunity is taken here to constrain the use of the EOT to one Pixel Data encoded as one frame per fragment, rather than permitting multiple frames per fragment as is theoretically possible currently, since in this day and age buffering an entire frame is not a practical problem and this greatly simplifies the life of the recipient, which can then use the EOT and the VL of the Item tag as the definition of the entire extent of the compressed pixel data for that frame, without worrying about parsing multiple items. The behavior in the absence of an EOT is unchanged; i.e., without an EOT, whether or not a BOT is present, frames may be fragmented if permitted by the Transfer Syntax.]

[Ed.Note.: Should the length of each encoded frame also be included in a new table? As long as there is one frame per fragment, this can be computed as the difference between successive offsets subtracting the item tag's length, but not for the last frame.
Though the explicit length of each item is already encoded in every item tag, and this would be redundant, having the information centralized would considerably simplify random access to frames without having to reindex the data.

Correction Wording:
Amend DICOM PS3.3 as follows (changes to existing text are bold and underlined for additions and struckthrough for removals):

3 Definitions
For the purposes of this Standard the following definitions apply.

3.5 DICOM Data Structures and Encoding
This Part of the Standard makes use of the following terms defined in ???:

Basic Offset Table See PS3.5.

Extended Offset Table See PS3.5.

4 Symbols and Abbreviations
The following symbols and abbreviations are used in this Part of the Standard.

BOT Basic Offset Table

EOT Extended Offset Table

C.7.6.3 Image Pixel Module
Table C.7-11a describes the Image Pixel Module.

Table C.7-11a. Image Pixel Module Attributes

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Tag</th>
<th>Type</th>
<th>Attribute Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include Table C.7-11c &quot;Image Pixel Description Macro Attributes&quot;</td>
<td></td>
<td></td>
<td>Required if the IOD is not being transferred in a STOW-RS Request and is not encoded as metadata and compressed bulk pixel data. May be present otherwise. See PS3.18 Section 6.6 “STOW-RS Request/Response”.</td>
</tr>
<tr>
<td>Pixel Data</td>
<td>(7FE0,0010)</td>
<td>1C</td>
<td>A data stream of the pixel samples that comprise the Image. See Section C.7.6.3.1.4 for further explanation. Required if Pixel Data Provider URL (0028,7FE0) is not present.</td>
</tr>
</tbody>
</table>
| Pixel Data Provider URL | (0028,7FE0) | 1C | A URL of a provider service that supplies the pixel data of the Image. Required if the image is to be transferred in one of the following presentation contexts identified by Transfer Syntax UID:  
  • 1.2.840.10008.1.2.4.94 (DICOM JPIP Referenced Transfer Syntax)  
  • 1.2.840.10008.1.2.4.95 (DICOM JPIP Referenced Deflate Transfer Syntax)  
  Note  
  The VR of this Data Element has changed from UT to UR. |
C.7.6.3.1 Image Pixel Attribute Descriptions

...  

C.7.6.3.1.X Extended Offset Table

The Extended Offset Table (0028,eee1) Value shall contain concatenated 64-bit unsigned integer values that are byte offsets to the first byte of the Item Tag of the first fragment for each frame in the Sequence of Items. These offsets are measured from the first byte of the first Item Tag following the empty (zero length) Basic Offset Table item, i.e., the Item Tag of the first fragment of the first frame.

Amend DICOM PS3.5 as follows (changes to existing text are bold and underlined for additions and struckthrough for removals):

3 Definitions

For the purposes of this standard, the following definitions apply.

3.10 DICOM Data Structures and Encoding Definitions

The following definitions are commonly used in this Standard:

Basic Offset Table: A table of 32-bit pointers to individual frames of an encapsulated multi-frame image.

Extended Offset Table: A table of 64-bit pointers to individual frames of an encapsulated multi-frame image.

4 Symbols and Abbreviations

The following symbols and abbreviations are used in this Standard.

BOT  Basic Offset Table

A.4 Transfer Syntaxes For Encapsulation of Encoded Pixel Data

These Transfer Syntaxes apply to the encoding of the entire DICOM Data Set, even though the image Pixel Data (7FE0,0010) portion of the DICOM Data Set is the only portion that is encoded by an encapsulated format. These Transfer Syntaxes shall only be used when Pixel Data (7FE0,0010) is present in the top level Data Set, and hence shall not be used when Float Pixel Data (7FE0,0008)
or Double Float Pixel Data (7FE0,0009) are present. This implies that when a DICOM Message is being encoded according to an encapsulation Transfer Syntax the following requirements shall be met:

1. ...

2. ...

3. The encoding of the Data Elements of the Data Set shall be as follows according to their Value Representations:

   • ...

   • For the Value Representations OB, OL and OW, the encoding shall meet the following specification depending on the Data Element Tag:

     • Pixel Data (7FE0,0010) may be encapsulated or native.

     It shall be encapsulated if present in the top-level Data Set (i.e., not nested within a Sequence Data Element).

     **Note**

     The distinction between fixed value length (native) and undefined value length (encapsulated) is present so that the top level Data Set Pixel Data can be compressed (and hence encapsulated), but the Pixel Data within an Icon Image Sequence may or may not be compressed.

     If native, it shall have a defined Value Length, and be encoded as follows:

     • where Bits Allocated (0028,0100) has a value greater than 8 shall have Value Representation OW and shall be encoded in Little Endian;

     • where Bits Allocated (0028,0100) has a value less than or equal to 8 shall have the Value Representation OB or OW and shall be encoded in Little Endian.

     **Note**

     a. The OL Value Representation is not used for Pixel Data, even if it has a Bits Allocated (0028,0100) of 32, since OL was added to the standard after the encoding of Pixel Data had been established

     b. That is, as if the Transfer Syntax were Explicit VR Little Endian.

     If encapsulated, it has the Value Representation OB and is an octet-stream resulting from one of the encoding processes. It contains the encoded pixel data stream fragmented into one or more Item(s). This Pixel Data Stream may represent a Single or Multi-frame Image. See Table A.4-1 and Table A.4-2.

     • ...

     • Each Data Stream Fragment encoded according to the specific encoding process shall be encapsulated as a DICOM Item with a specific Data Element Tag of Value (FFFE,E000). The Item Tag is followed by a 4 byte Item Length field encoding the explicit number of bytes of the Item.

     **Note**

     Whether more than one fragment per frame is permitted or not is defined per Transfer Syntax.

     • ...

     • The first Item in the Sequence of Items before the encoded Pixel Data Stream shall be a Basic Offset Table item. The Basic Offset Table Item Value, however, is not required to be present:

     • When the Item Value is not present, the Item Length shall be zero (00000000H) (see Table A.4-1).

     • When the Item Value is present, the Basic Offset Table Item Value shall contain concatenated 32-bit unsigned integer values that are byte offsets to the first byte of the Item Tag of the first fragment for each frame in the Sequence of Items. These offsets are measured from the first byte of the first Item Tag following the Basic Offset Table item (see Table A.4-2).
Note

1. For a Multi-Frame Image containing only one frame or a Single Frame Image, the Basic Offset Table Item Value may be present or not. If present it will contain a single 00000000H value.

2. Decoders of encapsulated pixel data, whether Single Frame or Multi-Frame, need to accept both an empty Basic Offset Table (zero length) and a Basic Offset Table filled with 32 bit offset values.

3. **A Basic Offset Table Item Value is not permitted (i.e., the Item Length of the first Item will be zero) if Extended Offset Table (0028, eee1) is present.**

   • ...

Table A.4-2. Examples of Elements for an Encoded Two-Frame Image Defined as a Sequence of Three Fragments with Basic Table Item Values

<table>
<thead>
<tr>
<th>Pixel Data Element Tag</th>
<th>Value Representation</th>
<th>Data Element Length</th>
<th>Data Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>(7FE0, 0010) with VR of OB</td>
<td>OB 0000H Reserved</td>
<td>FFFF FFFFH undefined length</td>
<td>Basic Offset Table with Item Value</td>
</tr>
<tr>
<td></td>
<td>Item Tag</td>
<td>Item Length</td>
<td>Item Value</td>
</tr>
<tr>
<td></td>
<td>(FFFE, E000)</td>
<td>0000 0008H</td>
<td>0000 0000H 0000 0646H</td>
</tr>
<tr>
<td></td>
<td>4 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Table A.4-2b. Examples of Elements for an Encoded Two-Frame Image Defined as a Sequence of Three Fragments with Basic Table Item Values (continued)

<table>
<thead>
<tr>
<th>Data Element Continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second Fragment (Frame 1) of Pixel Data</td>
</tr>
<tr>
<td>Item Tag</td>
</tr>
<tr>
<td>(FFFE, E000)</td>
</tr>
<tr>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Amend DICOM PS3.6 as follows (changes to existing text are bold and underlined for additions and struckthrough for removals):

6 Registry of DICOM Data Elements

Table 6-1. Registry of DICOM Data Elements

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Keyword</th>
<th>VR</th>
<th>VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0028, eee1)</td>
<td>Extended Offset Table</td>
<td>ExtendedOffsetTable</td>
<td>OV</td>
<td>1</td>
</tr>
</tbody>
</table>