|  |  |
| --- | --- |
| Text, logo  Description automatically generated | **ISO/IEC JTC 1/SC 29/WG 1 N101237**  **108th meeting – Daejon Korea, 30 June to 4 July 2025** |

|  |
| --- |
| **ISO/IEC JTC 1/SC 29/WG 1**  **(& ITU-T SG16)**  **Coding of Still Pictures**  **JBIG JPEG**  Joint Bi-level Image Joint Photographic  Experts Group Experts Group |

**TITLE:** Guideline on Placeholder Passes and Multiple HT Sets in HTJ2K codestreams – v2

**SOURCE:** WG 1

**EDITORS:**  David Taubman

**PROJECT:** ISO/IEC 15444-15

**STATUS:** Final

**REQUESTED**

**ACTION:** For review and publication on jpeg.org, replacing the existing document with the same title

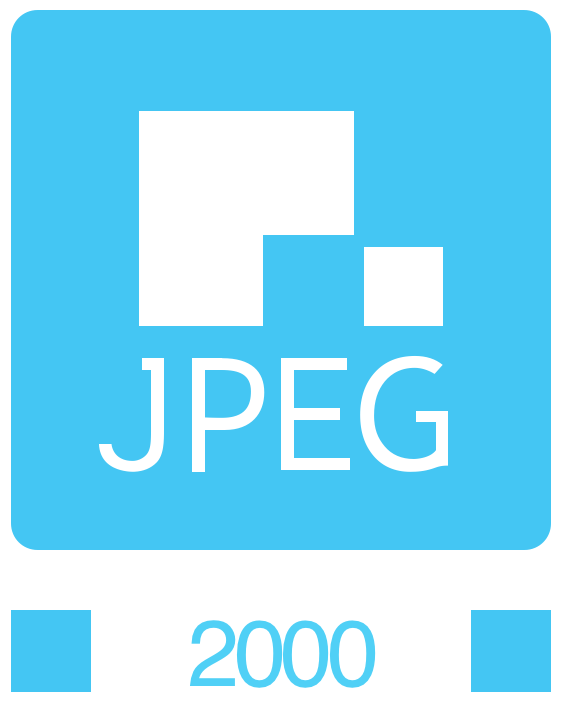
**DISTRIBUTION:** WG1

Contact:

ISO/IEC JTC 1/SC 29/WG 1 Convenor – Prof. Touradj Ebrahimi

EPFL/STI/IEL/GR-EB, Station 11, CH-1015 Lausanne, Switzerland

Tel: +41 21 693 2606, Fax: +41 21 693 7600, E-mail: [Touradj.Ebrahimi@epfl.ch](mailto:Touradj.Ebrahimi@epfl.ch)

******Placeholder Passes and Multiple HT Sets in HTJ2K codestreams – v2****Guideline**

# Introduction

The JPEG 2000 family of standards provides a large set of capabilities that can be used to address the needs of almost any image compression task. The complete set of capabilities, including resolution scalable, quality scalable and region accessible coding of images with up to 16384 components, up to 37 bits per sample and dimensions up to , is addressed by the finely embedded block coding algorithm defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, identified here as the J2K1 block coder. The high degree of quality scalability offered by the J2K1 block coder enables highly responsive remote interactive browsing of large image sources over low bandwidth lossy or lossless communication channels, which can be achieved using the technology defined in Rec. ITU-T T.808 | ISO/IEC 15444-9 (a.k.a. JPIP). For applications that require very high processing throughput, the “high throughput” (HT) block coding algorithm defined in Rec. ITU-T T.814 | ISO/IEC 15444-15 is particularly beneficial, allowing overall encoder and decoder throughput increases on the order of 10:1 on common platforms.

Codestreams that use the HT block coding algorithm are known as HTJ2K codestreams, as defined in Rec. ITU-T T.814 | ISO/IEC 15444-15. Moreover, it is possible to use either the HT or the J2K1 block coding algorithm within an HTJ2K codestream, and the choice of block coder can even change on a block-by-block basis. However, for most applications it is desirable to use a restricted class of HTJ2K codestreams, known as HTONLY codestreams, for which a decoder can be sure of encountering only HT block bit-streams.

While JPEG 2000 offers two different block coding algorithms, namely HT and J2K1, their coded representations are highly compatible. Both algorithms involve a set of coding passes that progressively refine the coded representation of a block of subband samples, at a rate of one magnitude bit-plane per three passes. In both cases, the coding passes are known as Cleanup, Significance Propagation (abbreviated as SigProp here) and Magnitude Refinement (abbreviated as MagRef here). In fact, it is always possible to reversibly transcode any J2K1 block bit-stream into a corresponding HT block bit-stream and vice-versa. Although HT bock bit-streams are not fully embedded, so that the quality scalability feature of JPEG 2000 is largely sacrificed in exchange for speed, this reversible transcoding feature allows the full quality scalability associated with a J2K1-based codestream to be recovered on demand from an HT-based codestream.

JPEG 2000 codestreams are organised as a sequence of “J2K packets,” not to be confused with network transport packets. The organization of code-blocks into precincts, precincts into a sequence of packets corresponding to global quality layers, and packets into coding pass contributions that are encoded within a packet header are all identical for both J2K1- and HT-based codestreams. Nonetheless, the construction and parsing of packet headers for HTJ2K codestreams involves some specific considerations, notably “placeholder passes” and “multiple HT Sets.” These concepts are explained further in Section 4.

The primary purpose of this document is to provide guidelines for HTJ2K decoder implementations, in regard to the parsing of packet headers and the use of placeholder passes and multiple HT Sets, to the extent that they are encountered. The second objective of this document is to explain how placeholder passes and multiple HT Sets can be used to preserve quality layer boundaries while transcoding between J2K1- and HT-based codestreams. Finally, this document provides guidance for HTJ2K encoders in regard to the use of placeholder passes and multiple HT Sets.

For simplicity, this document addresses only the case of HTONLY codestreams, for which a decoder can be sure of encountering only HT block bit-streams, as explained earlier. Another important class of HTJ2K codestreams is known as SINGLEHT, where a SINGLEHT codestream provides at most one HT Set for each code-block. The HTONLY and SINGLEHT restrictions are advertised in the CAP marker segment near the start of an HTJ2K codestream, so that decoders can assess their ability to process all content within the codestream. A decoder that supports only codestreams belonging to both the HTONLY and SINGLEHT classes still needs to address the possibility that placeholder passes are encountered within packet headers, so the guidelines in this document are highly relevant even in this simplified setting.

# Notation used in this document

* is the maximum number of bit-planes in a given code-block, as used in Rec. ITU-T T.800 | ISO/IEC 15444-1
* is the number of missing bit-planes for code-block , which is the same as from Annex B of Rec. ITU-T T.814 | ISO/IEC 15444-15 and also the same as the number of zero bit-planes from Annex B of Rec. ITU-T T.800 | ISO/IEC 15444-1
* is the same as the symbol P0 from Annex B of Rec. ITU-T T.814 | ISO/IEC 15444-15
* is the same as the set T from Annex B of Rec. ITU-T T.814 | ISO/IEC 15444-15
* is the number of quality layers in a codestream, which is also the number of packets for each precinct
* is the total number of coding passes contributed to code-block from packet
* is the number of distinguishable contributions to code-block from packet
* is the total number of distinguishable contributions to code-block from all packets
* is the index of the first contribution to code-block within packet , if there is one
* is the number of coding passes associated with contribution
* is the number of bytes associated with contribution

# J2K packet header parsing in general

A JPEG 2000 codestream consists of a sequence of J2K packets, prepended by a codestream main header and a first tile-part header, and optionally interspersed with additional tile-part headers. A codestream with quality layers has packets for each precinct, although a decoder might need to be prepared to handle truncated codestreams, and each precinct is a defined collection of code-blocks from a single tile-component-resolution – i.e., one resolution level within one component of a tile.

Conceptually at least, the packet headers of a precinct are parsed sequentially until all packets have been parsed or there are no more packets available for the precinct – this might happen if the codestream has been truncated. Following this sequence a packet header parser recovers the following quantities for each code-block in the precinct, regardless of whether the codestream is HT-based on J2K1-based.

1. A sequence of ordered pairs , for , describing a sequence of distinguishable contributions from code-block to the compressed representation, where is the number of coding passes in contriubtion and is the number of bytes (length) of the contribution.
2. If , the number of missing bit-planes for code-block .

This information is sufficient for the purpose of decoding and reconstructing the image; however, in some cases it is helpful to keep track of the quality layer index , of the J2K packet containing contribution .

In order to recover these quantities, the packet header parser performs tag-tree decoding and follows all other procedures described in Rec. ITU-T T.800 | ISO/IEC 15444-1. When parsing the header for packet , the parser first recovers the number of new coding passes that are contributed by the packet to code-block . This value is the sum of the values associated with all contributions to code-block that are found in the packet; that is,

The packet header parser uses to determine the number of contributions to code-block that are found in the packet, decoding length values . Note that the total number of contributions from all packets to code-block is

A key step here is the determination of . While the packet parsing procedure itself is generic, the method for determining depends upon a set that describes how the bit-stream for code-block is partitioned into segments.

Specifically, is a sequence of coding pass indices , such that segment consists of coding passes , for each . Here, is always implied, so that segment consists of passes to . That is, is the zero-based index of the last coding pass in segment .

It is worth clarifying the relationship between contributions and segments associated with a code-block bit-stream. Formally, the coding passes associated with a code-block are first partitioned into segments, as described by . Then as each packet contributes new coding passes to code-block these coding passes are associated with one or more segments. In this way a packet might contribute only some of the coding passes associated with a segment, so that the segment is split across multiple packets. Each contribution consists of the coding passes contributed by one packet that belong to one segment. Thus, when segments are split across multiple packets, the total number of contributions is larger than the total number of segments.

depends upon the block coder type (HT or J2K1) and other block coder mode flags found in COD and/or COC marker segments within the relevant main or tile-part headers. Moreover, for HT block bit-streams, actually depends on the point at which the first non-zero length contribution is discovered for code-block . The next section develops this point in detail.

# Placeholder passes, HT Sets and HTJ2K packet header parsing

## Placeholder passes

As mentioned earlier, HT and J2K1 block coding algorithms have the same fundamental set of coding passes, identified here as Cleanup, SigProp and MagRef. For the HT block coder, these coding passes are grouped into “HT Sets,” where each HT Set commences with an HT Cleanup pass. With the possible exception of the last HT Set, each HT Set for a code-block also consists of one SigProp pass and one MagRef pass, in that order; in general, no HT Set contains more than one coding pass of each type.

While multiple HT Sets may possibly be found within the codestream for a given code-block, a decoder only needs to decode at most one HT Set for each block. By contrast, a J2K1 block decoder generally needs to be prepared to process many coding passes. This is because a single HT Cleanup pass encodes all of the information that is encoded by both the corresponding J2K1 Cleanup pass and all earlier J2K1 coding passes. Nonetheless, all coding pass boundaries for J2K1 are also notionally available in the HT case, and an HT encoder might potentially generate encoded content for each pass before deciding which ones should be included within the final codestream.

The magnitude bit-plane associated with an HT Cleanup pass that is processed by the decoder is determined by the symbol S\_blk, as defined in Rec. ITU-T T.814 | ISO/IEC 15444-15, while the number of coding passes to be processed is denoted by the symbol Z\_blk, which is not more than 3 since only one HT Set needs to be processed. S\_blk can be understood as the number of magnitude bit-planes that are more significant than the one associated with the HT Cleanup pass, out of the total number of magnitude bit-planes that are available within the relevant sub-band. Thus, S\_blk can be as small as and as large as . The value of S\_blk for code-block is given by

Here, is the number of missing bit-planes recovered by packet header parsing, as explained earlier, is a number of “placeholder bit-planes,” that is also recovered by packet header parsing, and S\_skip denotes the number of initial available HT Sets that the decoder chooses to skip over.

S\_skip is necessarily zero unless a code-block contains multiple HT Sets; when there are multiple HT Sets, S\_skip can be chosen freely by the decoder, but specific guidance on how to make this choice is provided in Section 5.

While only the sum is needed for decoding, its partitioning into missing bit-planes and placeholder bit-planes provides a mechanism for preserving all relevant structural properties from a quality scalable J2K1-based codestream, as explained in Section 6.

The first coding passes recovered for an HT code-block by packet header parsing are known as “placeholder passes.” These placeholder passes are counted amongst the contribution pass counts .

## Discovering and the segment partition

For an HT code-block , the segment partition actually depends upon the number of placeholder bit-planes . Specifically,

This is an area of potential confusion for implementors, since depends on where the first non-zero length contribution occurs within the contribution descriptors, while is needed to recover these contribution descriptors during packet header parsing. Nonetheless, ambiguity is avoided by one very important constraint that is imposed upon an HTJ2K codestream by Rec. ITU-T T.814 | ISO/IEC 15444-15; specifically:

“The packet as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 that contains the first HT Cleanup coding pass for a code-block shall include only one HT Cleanup coding pass.”

To see how this constraint allows a decoder to unambiguously recover the value of along with all contribution descriptors , recall that while parsing the header of packet , the parser first recovers the total number of coding passes contributed by the packet to code-block . Then if , there must be at least one contribution , whose length can be recovered before deciding the number of contributions ; here is the index of the first contribution from packet . Then the following cases exist:

* If and and no earlier packet has contributed any bytes to block *b*, then packet makes only contribution, with passes. These passes are all placeholder passes.
* If and and no earlier packet has contributed any bytes to block , then must be the length of the first HT Cleanup pass; therefore must be equal to . According to the constraint quoted above, this packet may contain only one HT Cleanup pass, and so there are only 3 possibilities:
  1. – then and
  2. – then and
  3. – then and

In all three cases, the values of and are unambiguously determined and if , the length of the second contribution is parsed from the J2K packet header, as described in Rec. ITU-T T.800 | ISO/IEC 15444-1.

* If and at least one earlier packet has provided a non-zero length contribution to code-block , then the segment partition is already known, because has been recovered previously. Therefore the number of contributions from packet to code-block is readily determined, along with the values for , which allows the lengths of these contributions also to be parsed from the packet header, following the methods defined in Rec. ITU-T T.800 | ISO/IEC 15444-1.

## Understanding multiple HT Sets

After a packet that contains the first HT Cleanup pass for a code-block , there are no constraints on the number of coding passes that can be contributed to code-block within subsequent packets, apart from the constraint that cannot exceed , which is common to both J2K1 and HT block bit-streams. Each additional HT Cleanup pass that is contributed by subsequent packets belongs to an additional HT Set and each such additional HT Set, other than the last one, contains additional HT SigProp and HT MagRef passes.

As explained earlier, each HT Cleanup pass encodes all of the information associated with earlier coding passes for a block , so that blocks with multiple HT Sets (and hence multiple Cleanup passes) could introduce substantial redundancy. However, non-initial HT Cleanup passes for a block are allowed to have zero length. When this happens, all coding passes in the corresponding HT Set are required also to have zero length, so that the HT Set is empty; such an empty HT Set can be understood as a “placeholder set,” which can be used to preserve quality layer boundaries within an HT block bit-stream while reducing redundancy.

In general, placeholder passes and placeholder sets allow the mapping between each coding pass index and quality layer to be preserved for each block via the information found in packet headers, regardless of how many non-empty HT Sets a content generator chooses to include in the codestream. If at most one non-empty HT Set is included for each code-block, then there is no redundancy in the codestream, and there are also no placeholder sets for any code-block, since the first HT Set is non-empty by definition. In this case placeholder passes alone are sufficient to preserve the information associated with quality layer boundaries. What this means is that code-blocks with multiple HT Sets are only useful for preserving quality layer boundaries that lie between two non-empty HT Sets, in which case the representation necessarily contains redundancies. Nonetheless, a decoder that claims to be capable of processing HTJ2K codestreams that are not SINGLEHT codestreams needs to be prepared for the possibility that all HT Sets after the first one might be empty.

## Worked Example with Multiple HT Sets

For a certain code-block , this example tracks the parsing of J2K packet header information from four packets illustrating how placeholder passes and multiple HT Sets are recovered.

1. Packet is found to contribute coding passes from block and the first associated length value is decoded as ,
   * hence we conclude that 5 placeholder passes have been found and there are no more lengths to decode for block in this packet (i.e., );
   * we found
2. Packet is found to contribute coding passes from block and the first associated length value is decoded as ,
   * hence we conclude that 3 placeholder passes have been found and there are no more lengths to decode for block in this packet (i.e., );
   * we found
3. Packet is found to contribute coding passes from block and the first associated length value is decoded as ,
   * hence we conclude that we have found the first HT Cleanup pass, which must contain 23 bytes;
   * since we previously found 8 passes, we have now discovered a total of 14 coding passes, which is of the form , where ;
   * we thus conclude that there is one refinement pass (an HT SigProp pass) beyond the HT Cleanup pass and a total of 12 placeholder passes;
   * the number of placeholder bit-planes is ;
   * the number of contributed segments is , so we decode a second length value (for the HT SigProp pass) and find it to be 5;
   * we found and
4. Packet is found to contribute coding passes from block ,
   * the first of these passes must be the HT MagRef pass that completes the first HT Set, so it has its own length which we decode as 3;
   * the second coding pass corresponds to a second HT Cleanup pass, having its own length, which we decoded as 0, meaning that the entire HT Set must be empty (a placeholder set);
   * the 3rd and 4th coding passes correspond to HT SigProp and HT MagRef refinement passes which contribute a single segment whose length is decoded but necessarily turns out to be 0, since these belong to the placeholder set identified above;
   * the 5th coding pass is another HT Cleanup pass, having its own length, which we decode as 102;
   * we found , , and ;
   * it would be sensible for the block decoder to process the third HT Set, whose Cleanup pass has 105 bytes, rather than the first HT Set, whose Cleanup pass has only 23 bytes, but this is ultimately a decoder choice.

# HTJ2K decoder guidelines

All HTJ2K decoders must be capable of handling placeholder passes that might be encountered during packet header parsing. There are no conditions under which a decoder can assume that is zero.

In the event that a decoder encounters a code-block for which multiple non-empty HT Sets are available, it is recommended that the decoder process the last such HT Set, in preference to an earlier one. This is because later HT Sets correspond to higher precision representations of the code-block sample data, except in the case of placeholder sets of course.

There might, however, be some conditions under which a decoder chooses to process a lower precision HT Set; for example:

* If the decoder is unable to process a higher precision HT Set, due to implementation precision limitations, it might choose to process a lower precision non-empty HT Set for the same code-block, if one is available.
* If the codestream is received progressively, a decoder might first reconstruct a lower quality representation of the code-block based on any available non-empty HT Sets and later refresh the decoded result using higher precision non-empty HT Sets, if any become available.

# Reversible transcoder guidelines: J2K1 to HT

Any J2K1-based codestream can be transcoded to an HT-based codestream in a reversible way, meaning that no information is lost and the HT-based codestream can even be transcoded back to a J2K1-based codestream, preserving all original quality layer boundaries. This section describes the recommended way to do this, wherein all J2K1 coding passes up to and including the last available Cleanup pass for a code-block are collapsed into a single HT Cleanup pass, which is followed by at most one HT SigProp pass and at most one HT MagRef pass. In this way, only one HT Set is included in the transcoded codestream for each code-block.

Specifically, let be the index of the last Cleanup pass in a J2K1 code-block , so that the total number of coding passes for the block satisfies . Then is set to . Equivalently, the number of placeholder passes is set to , meaning that all coding passes prior to the last Cleanup pass in the J2K1 block bit-stream become placeholder passes in the transcoded representation.

After decoding the J2K1 block bit-stream, the transcoder encodes the one HT Cleanup pass with index , meaning that

where is the number of zero bit-planes recovered while parsing packet headers from the original codestream.

Additionally, if , the trancoder encodes an HT SigProp pass with index , while if , the transcoder encodes an HT SigProp pass and an HT MagRef pass, with indices and , respectively.

## Packet header transcoding

The transcoder outputs the same number of packets for each precinct that were found in the original J2K1-based codestream. In order to preserve all quality layer boundaries, each packet should then contribute exactly the same number of coding passes to block in both the original and transcoded codestreams.

It is possible for a transcoder to reduce the number of placeholder passes that appear in the first quality layer by a multiple of , adjusting the number of missing bit-planes accordingly, so that remains unchanged. This does not have any impact upon the preservation of quality layer boundaries within the transcoded codestream, but can have a small impact on the size of packet headers. Transcoders that choose to do this should leave at most 2 placeholder passes within the first quality layer . That is, transcoders should either reduce the number of placeholder passes in the first quality layer to the maximum possible extent, or else leave equal to the original number of missing bit-planes for the block. Following this sensible strategy allows decoders or reverse transcoders to conclude that whenever a code-block has 3 or more placeholder passes in the first quality layer, the most significant magnitude bit-planes for the block are entirely zero, which may allow the decoder to adopt a more computationally efficient block decoding or reverse transcoding implementation when processing such code-blocks, at least on a speculative basis.

## Worked example with codestream data

The electronic attachment to this document provides a very simple J2K1 codestream, as “orig.j2c,” that encodes a 12-bit greyscale image with no levels of wavelet decomposition and just one code-block into 3 quality layers. An HTJ2K transcoded version of this codestream is also provided, as “transcoded.j2c.” The three packet headers found in these codestreams have the following attributes:

1. Packet

* The original codestream identifies along with new coding passes, consisting of 4 Cleanup passes, 4 SigProp passes and 3 MagRef passes.
* The transcoded codestream identifies along with new coding passes, all of which are placeholder passes because .

1. Packet

* The original codestream identifies new coding passes, starting with a MagRef pass and finishing with a SigProp pass.
* The transcoded codestream identifies new coding passes and a first length value of . It follows that the first 16 of these new coding passes are placeholder passes, while the last two are HT coding passes. We conclude that . The HT Cleanup pass, with 5284 bytes, holds the information that was transcoded from the last J2K1 Cleanup pass. The HT SigProp pass that is also found in this packet has its own segment, with length ; this holds the information that was transcoded from the final J2K1 SigProp pass.

1. Packet

* The original codestream identifies new coding pass, which is a MagRef pass.
* The transcoded codestream identifies new coding pass and a length value of ; this is the HT MagRef pass that was transcoded from the final J2K1 MagRef pass.

Note that the transcoded codestream “transcoded.j2c” is not unique for various reasons. First, number of missing bit-planes can be restored to its original value by augmenting the number of placeholder passes in the first packet from 2 to 11, thereby adding 3 to the number of placeholder bit-planes . Secondly, the HT encoder’s strategy for terminating each of its coding passes and sub-streams can vary, so long as the termination is compatible with correct decoding.

# HTJ2K encoder guidelines

Encoders should include at most one HT Set within the codestream for each code-block, unless they have a good reason to do otherwise. One possible reason to generate multiple HT Sets is presented below, although this necessarily involves the introduction of redundant encoded content into the codestream.

Moreover, when an encoder includes at most one HT Set within the codestream, it should identify the codestream as having the SINGLEHT attribute via the CAP marker segment that appears after the SIZ marker segment at the start of the codestream.

Encoders should use for each code-block , unless they have a good reason to do otherwise. Non-zero values for are only needed to preserve quality layer boundaries within the codestream, but some applications might potentially use to trigger custom behaviour.

As explained in Section 6.1, transcoders should either reduce to the smallest value that is consistent with preserving quality layer boundaries or else adopt the value that leaves equal to the actual number of missing MSBs from an original J2K-1 based codestream, so that at least the most significant magnitude bit-planes of the code-block are certain to be entirely zero. Since there is no completely general way for a decoder to know whether an HTJ2K codestream has been generated by transcoding or not, encoders should follow the same policy. Specifically, if the number of placeholder passes included for block in the first quality layer is 3 or more, then all most significant magnitude bit-planes of the code-block should be entirely zero.

Following this guideline, the electronic attachment to this document provides two examples of HTJ2K codestreams produced by losslessly encoding the same 12-bit greyscale image that was used for the transcoding example in Section 6.2. Again, the number of decomposition levels is set to 0 so that there is only one code-block in each of these codestreams. Both codestreams have only one quality layer, with only the HT Cleanup pass. The codestream contained in file “encoded\_no\_phld.j2c” has and , while the codestream contained in file “encoded\_phld.j2c” has and . Only in the second case, where , should the most significant magnitude bit-planes be entirely 0, which is certainly satisfied in this example with .

Encoders can use placeholder passes and even multiple HT Sets to introduce quality layer boundaries into an HT block bit-stream representation. The main reason for doing this is to facilitate later transcoding operations, including transcoding operations that produce HT-based and J2K1-based codestreams.

For example, an image or video content server application that needs to offer codestreams encoded at a multitude of bit-rates could be orchestrated by encoding each image or video frame at the highest of the target bit-rates, while including quality layer boundaries within each codestream that correspond to each of the lower target bit-rates of interest. The lower quality representations can then be recovered on-demand by transcoding.

There are two ways in which an encoder can facilitate later transcoding. In the first approach, the encoder includes at most one HT Set for each code-block, using placeholder passes to record the quality boundaries associated with lower quality layers; this is no different from the way in which a J2K1 to HT transcoder preserves quality boundaries from an original J2K1-based codestream. Such codestreams contain quality layers and can be transcoded to a lower quality representation corresponding to the first quality layers, by first decoding each code-block and subsequently encoding one HT Cleanup pass, having the largest index within those coding passes that are contributed by the first packets of the original codestream; additionally, the transcoder should encode the HT SigProp pass with index and the HT MagRef pass with index , to the extent that these are contributed also by the first packets. While this approach does not avoid the need for decoding and re-encoding of the code-blocks, it does avoid the need for a transcoder to perform the non-trivial task of identifying the specific coding passes that should be generated for each code-block during re-encoding; this information has been embedded within the codestream.

The second way in which an encoder can facilitate transcoding is by including multiple non-empty HT Sets, such that the encoded content associated with multiple quality layers is already present and need not be regenerated during transcoding. This necessarily renders the original codestream redundant, which might not be desirable. As mentioned earlier, this case in which a codestream contains redundant information is the only case in which multiple HT Sets are needed for a code-block, whether empty or otherwise.