The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

IMAGE DATA COMPRESSION

Draft Recommended Standard

CCSDS 122.0-P-1.1

Pink Sheets
July 2016

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

IMAGE DATA COMPRESSION

Draft Recommended Standard

CCSDS 122.0-P-1.1

Pink Sheets
July 2016

DOCUMENT CONTROL

Document	Title	Date	Status/Remarks
CCSDS 122.0-B-1	Image Data Compression, Recommended Standard, Issue 1	November 2005	Original issue
CCSDS 122.0-P-1.1	Image Data Compression, Draft Recommended Standard, Issue 1.1	July 2016	Current proposed draft update: – adds modifications to support proposed draft Recommended Standard for Spectral Pre-Processing Transform for Multispectral and Hyperspectral Image Compression.

NOTE – Only pages containing substantive changes are included.

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to establish a Recommended Standard for a data compression algorithm applied to two-dimensional digital spatial image data from payload instruments and to specify how this compressed data shall be formatted into coded segments to enable decompression at the receiving end.

Source coding for data compression is a method utilized in data systems to reduce the volume of digital data to achieve benefits in areas including, but not limited to,

a) reduction of transmission channel bandwidth;

b) reduction of the buffering and storage requirement;

c) reduction of data-transmission time at a given rate.

1.2 SCOPE

The characteristics of instrument data are specified only to the extent necessary to ensure multi-mission support capabilities. The specification does not attempt to quantify the relative bandwidth reduction, the merits of each approach discussed, or the design requirements for coders and associated decoders. Some performance information is included in reference [B1].

This Recommended Standard addresses image data compression, which is applicable to a wide range of space-borne digital data, where the requirement is for a scalable data reduction, including the option to use lossy compression, which allows some loss of fidelity in the process of data compression and decompression. Reference [B1] gives an outline for an implementation.

1.3 APPLICABILITY

This Recommended Standard applies to data compression applications of space missions anticipating packetized telemetry cross support. In addition, it serves as a guideline for the development of compatible CCSDS Agency standards in this field, based on good engineering practice.

1.4 RATIONALE

The concept and rationale for the Image Data Compression algorithm described herein may be found in reference [B1].

This Recommended Standard supports both frame-based input formats produced, for example, by CCD arrays (called image frames) and strip-based input formats produced by push-broom type sensors (called image stripmaps). ~~An image pixel dynamic range of up to 16 bits is supported.~~High dynamic range images are supported; the maximum allowed pixel bit depth varies from 25 to 28 bits (see 3.2.1). The algorithm specified supports a memory-effective implementation of the compression procedure which does not require large intermediate frames for buffering (see reference [B1]).

As described in reference [B8], this Recommended Standard may be used in combination with a pre-processing transform to exploit similarities among the bands of multispectral and hyperspectral images. In this case the term 'pixel', as employed through this document, is to be understood as a coefficient of a pre-processed multispectral or hyperspectral image.

2.2 DATA DELIVERY

The encoded bitstream corresponding to an image frame or stripmap, consists of a single coded segment or a sequence of coded segments. Each coded segment consists of a header followed by a coded data field. A coded segment can have either fixed length or variable length depending on the operational mode selected. Values of algorithm parameters used in encoding can be specified using the header defined in section 4.

The effects of a single bit error can propagate to corrupt reconstructed data to the end of the affected segment (see reference [B1]). Therefore measures should be taken to minimize the number of potential bit errors on the transmission link. The transport mechanism for the delivery of the encoded bitstream shall support, in the event of a bit error, the ability to relocate the header of the next coded segment.

In case the encoded bitstream is to be transmitted over a CCSDS space link, several protocols can be used to transfer the sequence of coded segments. Those protocols are specified in references:

- reference [B3], Space Packet Protocol;
- reference [B4], CCSDS File Delivery Protocol (CFDP);
- reference [B5], packet service or bitstream service as provided by the AOS Space Data Link Protocol.

Interface to those space link protocols and related issues are discussed in reference [B1].

3 DESCRIPTION OF THE DISCRETE WAVELET TRANSFORM

3.1 OVERVIEW

This Recommended Standard for the decorrelation module makes use of a three-level, two-dimensional (2-d), separable Discrete Wavelet Transform (DWT) with nine and seven taps for low- and high-pass filters, respectively. Such a transform is produced by repeated application of a one-dimensional (1-d) DWT described in 3.3, 3.5, and 3.7. Two specific 1-d wavelets are specified with this Recommended Standard: the 9/7 biorthogonal DWT, referred to as '9/7 Float DWT' or simply 'Float DWT', and a non-linear, integer approximation to this transform, referred to as '9/7 Integer DWT' or simply 'Integer DWT'.

The definition of the functional inverse of either DWT is included in 3.4, 3.6, and 3.8.

While the Float DWT generally exhibits superior compression efficiency in the lossy domain, only the Integer DWT supports strictly lossless compression.

Image data is assumed to use R-bit pixels to represent either signed or unsigned integer values~~, where $R \leq 16$~~. The maximum supported value of R depends on whether image pixels are signed or unsigned, and whether the Float or Integer DWT is used (see 3.2.1).

The values output from the 3-level 2-d DWT (see 3.7) are converted to appropriate integer values before applying the Bit Plane Encoder (BPE—see section 4). Each integer is represented using a binary word consisting of a single sign bit along with several magnitude bits. The maximum word size necessary to store ~~each~~ such integer values depends on ~~R~~the choice of DWT, the input bit depth R, and whether image pixels are signed or unsigned.

In the case of the Float DWT, the computed wavelet domain values are rounded to the respective nearest integers before applying the BPE. ~~A corresponding word length of R+5 bits is adequate to store these integer values.~~

In the case of the Integer DWT, before applying the BPE, the computed wavelet domain values are multiplied by integer weights that are uniform in each subband (see 3.9).~~ A corresponding word length of R+4 bits is adequate to store such integers before the weighting factors are applied. Following the weighting operation, longer words might be used to store wavelet coefficients in some implementations.~~

Implementation schemes and image reconstruction methods are not part of this Recommended Standard, although some approaches are outlined in reference [B1]. Compressor implementations can differ considerably depending, for example, on whether the implementation waits for an entire frame to be formed before beginning compression.

3.2 IMAGE FRAME

3.2.1 Input image pixels may be signed or unsigned integers. Pixel bit depth, *R*, shall not exceed the limit indicated in table 3-1. The stated limits for signed values are valid for integers represented either in sign-and-magnitude or in two's complement.

Table 3-1: Maximum Pixel Bit Depth

	Pixel Type	
	Unsigned	Signed
Integer DWT	25	25
Float DWT	27	28

NOTE – These limits on input pixel bit depth ensure that DWT coefficients to be encoded by the BPE do not exceed the 32-bit dynamic range supported by this Recommended Standard.

3.2.2 Calculation of both the Float DWT and Integer DWT depends on the dimensions, i.e., row width and column height, of the image frame being transformed.

NOTE – This formal consideration of frame size does not imply that an implementation has to provide buffers for complete frames.

3.2.3 Image width shall be communicated to the decoder using the segment headers defined in 4.2. The image height is inferred indirectly via a header flag indicating the last segment of an image, as described in 4.2.

NOTE – Although the compressor will operate on small images, it is designed to exploit two-dimensional correlations in the image, and consequently, compression performance may be poor when either image height or width is smaller than 32.

3.2.4 The maximum image width (number of columns) is 2^{20}; the minimum image width is 17. There is no restriction on the maximum number of rows that constitute one image frame; the minimum number of rows is 17.

3.2.5 The application of either Float or Integer DWT to an image requires that the image dimensions be integer multiples of eight. If the width or height of the original image is a not a multiple of eight, the image shall be 'padded' by appending the minimum number of extra rows and/or columns to produce an image with both width and height divisible by eight:

a) when columns are added for padding,

   1) columns shall be appended at the right edge of the image, i.e., to the edge having the highest pixel index in the horizontal direction,

   2) pixel values of appended columns shall be copies of the value of the right-most image column;

A *segment* is defined as a group of S consecutive blocks. Coding of DWT coefficients proceeds segment-by-segment and each segment is coded independently of the others. S can be assigned to any value between $16 \leq S \leq 2^{20}$, except for the last segment of an image, for which S can be assigned to any value between $1 \leq S \leq 2^{20}$. The value might be chosen based on the memory available to store the segment data. When multiple image frames are transmitted, the coding of each new frame starts with a new segment, i.e., a single coded segment must not contain coded data from two separate frames.

A segment of blocks is further partitioned into *gaggles*. Each gaggle consists of 16 blocks, except for possibly the last gaggle in a segment, which contains S mod 16 blocks when S is not a multiple of 16.

DC coefficients are represented using two's-complement representation. Let c_m denote the m^{th} DC coefficient in a segment, i.e., the DC coefficient of the m^{th} block in a segment. The number of bits needed to represent c_m in two's-complement representation is given in equation 12:

$$\begin{aligned} &1 + \lceil \log_2 |c_m| \rceil, && \text{if } c_m < 0 \\ &1 + \lceil \log_2 (1 + c_m) \rceil, && \text{if } c_m \geq 0 \end{aligned} \tag{12}$$

Within a segment, BitDepthDC is defined as the maximum of this value over all DC coefficients (i.e., all values of m) in the segment. Each DC coefficient in the segment is represented using BitDepthDC bits, in two's-complement representation. It should be noted that the minimum value of BitDepthDC is 1.

An AC coefficient is represented using the binary representation of the magnitude of the coefficient, along with a bit indicating the sign when the coefficient is nonzero. $\text{BitDepthAC\_Block}_m$ in equation 13 denotes the maximum number of bits needed to specify the magnitude of any AC coefficient in the m^{th} block.

$$\text{BitDepthAC\_Block}_m = \lceil \log_2 (1 + \max_x (|x|)) \rceil \tag{13}$$

where the maximization is over all AC coefficients x in the block.

For each segment, the BPE computes BitDepthAC, which denotes the maximum value of $\text{BitDepthAC\_Block}_m$ for the segment:

$$\text{BitDepthAC} = \max_{m=0,\ldots,S-1} \text{BitDepthAC\_Block}_m \tag{14}$$

It should be noted that the minimum value of BitDepthAC is 0.

The BPE successively encodes bit planes of coefficient magnitudes in a segment, inserting AC coefficient sign values at appropriate points in the ~~encoded~~coded segment data stream. *Bit plane b* consists of the b^{th} bit of the two's-complement integer representation of each DC coefficient, and the b^{th} bit of the binary integer representation of the magnitude of each AC coefficient. Here, bit plane index b=0 corresponds to the least significant bit. The BPE proceeds from most-significant bit to least significant bit, thus b decreases from one bit plane

to the next, beginning with b = BitDepthAC-1, and ending with b=0. DWT coefficient resolution effectively improves by a factor of 2 as encoding proceeds from one bit plane to the next. The bit plane coding process is described in 4.5.

~~Figure~~ 4-2~~(a) gives an overview of the structure of any single coded segment.~~ The structure of a *coded segment* is shown in figure 4-2(a). Within a coded segment, header information is encoded. Then quantized DC coefficients from the blocks are encoded. Then AC bit depths are encoded. Then DWT coefficient blocks are encoded, one bit plane at a time, proceeding from the most significant to the least significant bit plane. The coding of a single bit plane is performed in several stages, and the resulting order of ~~code~~encoded data is illustrated in figure 4-2(b). E.g., parent coefficients are coded in stage 1 for all blocks of the segment before encoding child coefficients in stage 2. The resulting encoded bit stream constitutes an embedded data format that provides progressive transmission.

(a) With All Bit Planes

Segment Header (see 4.2)
Initial coding of DC coefficients (see 4.3)
Coded AC coefficient bit depths (see 4.4)
Coded bit plane b = BitDepthAC-1 (see 4.5)
Coded bit plane b = BitDepthAC-2 (see 4.5)
⋮
Coded bit plane b = 0 (see 4.5)

(b) Within One Bit Plane

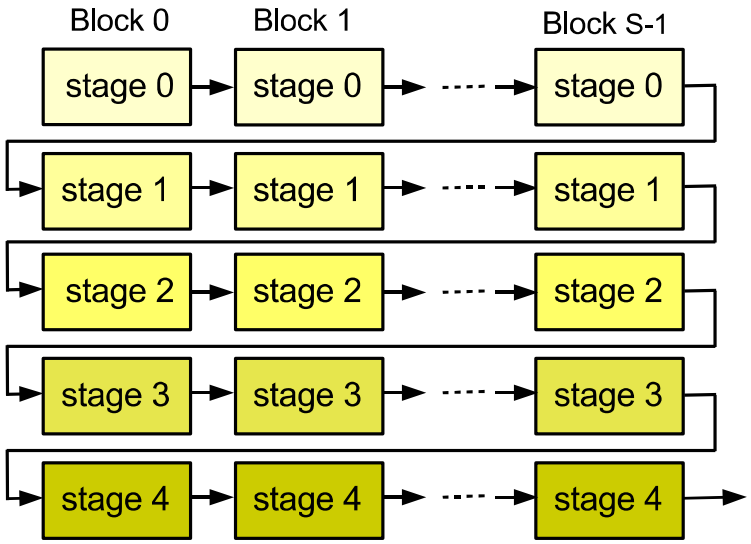

Figure 4-2: Overview of the Structure of a Coded Segment

The tradeoff between reconstructed image quality and compressed data volume for each segment is controlled by specifying the maximum number of bytes in each ~~compressed~~coded segment, SegByteLimit, and a 'quality' limit. The quality limit constrains the amount of DWT coefficient information to be encoded, and is specified as a bit plane index and a stopping point within that bit plane, as described in 4.2. ~~Compressed output for a~~Data for a coded segment is produced until the byte limit or quality limit is reached, whichever comes first.

~~The encoded bitstream for a~~A coded segment can be further truncated (or, equivalently, coding can be terminated early) at any point to further reduce the data rate, at the price of reduced image quality for the corresponding segment.

The remainder of this section describes each component of the coded segment. Subsection 4.2 describes the segment header. Subsection 4.3 describes the initial coding of DC coefficients. Subsection 4.4 describes the coded sequence of BitDepthAC_Block$_m$ values. Subsection 4.5 describes the coding process for a bit plane of the segment.

4.2 SEGMENT HEADER

4.2.1 GENERAL

4.2.1.1 Each ~~compressed~~coded segment shall begin with a segment header consisting of the following parts in the following order:

a) Part 1 (three or four bytes, mandatory—see 4.2.2);

b) Part 2 (five bytes, optional—see 4.2.3);

c) Part 3 (three bytes, optional—see 4.2.4);

d) Part 4 (eight bytes, optional—see 4.2.5).

4.2.1.2 By CCSDS convention, all reserved bits in each header part shall be set to 'zero'.

4.2.1.3 Optional header parts may be omitted when the parameters described in a part can be determined without that part, e.g., when those parameters are set to known fixed values for an entire mission.

NOTE – Figure 4-3 gives an overview of the header structure and table 4-3 provides a high-level description of the functionality of each part of the header.

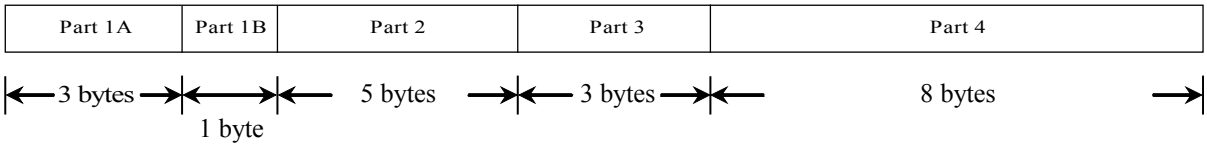

Figure 4-3: Overview of Segment Header Structure When All Parts Are Included

Table 4-3: Summary of Segment Header Functionality

Header Part	Length (Bytes)	Status	Description of Contents	Reference
Part 1A	3	Mandatory	Flags first and last segments of image; indicates which optional header parts are included; encodes information that typically changes from segment-to-segment.	4.2.2
Part 1B	1	Mandatory for the last segment of image, not included otherwise	Specifies number of 'padding' rows to be deleted after image reconstruction.	4.2.2
Part 2	5	Optional	Specifies limits on ~~compressed~~number of bytes per coded segment and image quality.	4.2.3
Part 3	3	Optional	Coding options including number of blocks per segment.	4.2.4
Part 4	8	Optional	Image and compression parameters that must be fixed for an image.	4.2.5

NOTE – The mandatory first part of the header includes values of coded segment information that change from segment-to-segment. The optional second part of the header specifies limits on the number of ~~compressed~~ bytes in a coded segment and the limits on the fidelity with which DWT coefficients are encoded. This part might be included at the start of an image or application session, or at the beginning of each coded segment for variable output rate control. The optional third part of the header specifies information that is typically fixed for each image or application session, but is allowed to change with each coded segment. In a typical application, this part might be included at the beginning of each image, but not included for each coded segment. The fourth part of the header specifies parameters that must be fixed for an entire image.

4.2.2 SEGMENT HEADER PART 1

4.2.2.1 General

Segment Header Part 1 consists of two subparts:

a) Part 1A (3 bytes, mandatory for all coded segments);

b) Part 1B (1 byte, mandatory for the last coded segment, otherwise not used).

NOTE – Table 4-4 summarizes the contents of Segment Header Part 1. Figure 4-4 illustrates the Segment Header Part 1 structure.

Table 4-4: Contents of Segment Header Part 1

Part	Field	Width (bits)	Description	Details
Part 1A	StartImgFlag	1	Flags initial segment in an image	1: first segment in image 0: continuation segment in image
	EndImgFlag	1	Flags final segment in an image	1: last segment in image 0: otherwise
	SegmentCount	8	Segment counter value	segment count value (mod 256), encoded as an unsigned binary integer
	BitDepthDC	5	Number of bits needed to represent DC coefficients in 2's complement representation	value of BitDepthDC (mod 32) encoded as an unsigned binary integer
	BitDepthAC	5	Number of bits needed to represent absolute value of AC coefficients in unsigned integer representation	value of BitDepthAC encoded as an unsigned binary integer
	Reserved	1	Reserved for future use	0
	Part2Flag	1	Indicates presence of Part 2 header	1: Part 2 of header present 0: Part 2 of header absent
	Part3Flag	1	Indicates presence of Part 3 header	1: Part 3 of header present 0: Part 3 of header absent
	Part4Flag	1	Indicates presence of Part 4 header	1: Part 4 of header present 0: Part 4 of header absent
Part 1B	PadRows	3	Number of 'padding' rows to delete after inverse DWT	Value encoded as unsigned binary integer
	Reserved	5	Reserved for future use	00000

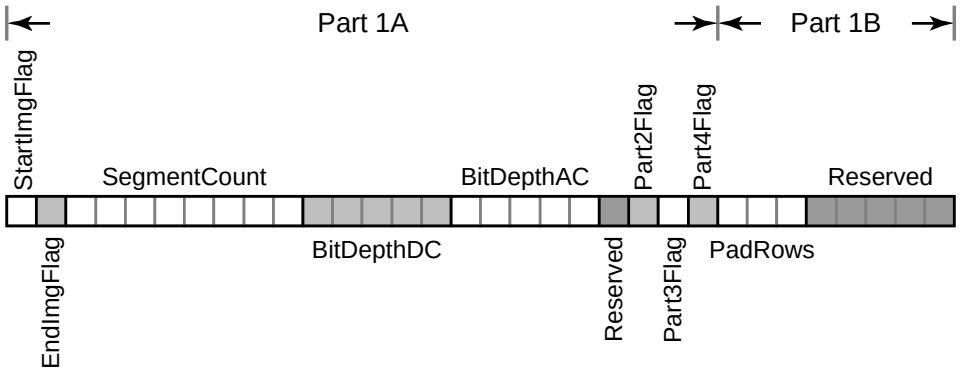

Figure 4-4: Segment Header Part 1 When Part 1B Is Included

4.2.2.2 Segment Header Part 1A

4.2.2.2.1 Bit 0 of Segment Header Part 1A shall contain the Start Image flag (StartImgFlag). The Start Image flag shall be used to indicate whether the coded segment corresponds to the start of an image:

– '1' shall indicate that the segment is the first segment in the image;

– '0' shall indicate that the segment is not the first segment.

4.2.2.2.2 Bit 1 of Segment Header Part 1A shall contain the End Image flag (EndImgFlag). The End Image flag shall be used to indicate whether the coded segment ~~is~~corresponds to the last segment in the image:

– '1' shall indicate that the segment is the last segment in the image, in which case Segment Header Part 1B must be included;

– '0' shall indicate that the segment is not the last segment, in which case Segment Header Part 1B shall not be included.

4.2.2.2.3 Bits 2-9 of Segment Header Part 1A shall provide the segment counter (SegmentCount):

a) the eight-bit segment counter shall be encoded as an unsigned binary integer;

b) the segment counter shall equal zero for the first segment of an image and shall increment by one for each subsequent segment;

c) the segment counter shall wrap back to zero and resume incrementing after reaching a value of 255.

NOTE – This counter provides an index of segments, modulo 256, within the image. Absolute segment counting can be implemented using the CCSDS Packet Sequence Count as described in reference [B3].

4.2.2.2.4 Bits 10-14 shall contain the BitDepthDC field~~. The value of the five-bit BitDepthDC field shall be encoded as an unsigned binary integer and shall equal~~, which encodes the value of BitDepthDC modulo 32 as an unsigned binary integer. BitDepthDC is equal to the number of bits needed to represent DC coefficients for the segment in 2's complement representation.

4.2.2.2.5 Bits 15-19 shall contain the BitDepthAC field. The value of the five-bit BitDepthAC variable shall be encoded as an unsigned binary integer and shall equal the number of bits needed to represent the absolute value of AC coefficients in unsigned integer representation (see 4.4).

4.2.2.2.6 Bit 20 is reserved for future use by the CCSDS and shall be set to '0'.

4.2.2.2.7 Bit 21 of Segment Header Part 1A shall contain the Part 2 flag (Part2Flag) and shall indicate the presence or absence of optional Segment Header Part 2:

– ‘1’ shall indicate that Segment Header Part 2 is present;

– ‘0’ shall indicate that Segment Header Part 2 is absent.

4.2.2.2.8 Bit 22 of Segment Header Part 1A shall contain the Part 3 flag (Part3Flag) and shall indicate the presence or absence of optional Segment Header Part 3:

– ‘1’ shall indicate that Segment Header Part 3 is present;

– ‘0’ shall indicate that Segment Header Part 3 is absent.

4.2.2.2.9 Bit 23 of Segment Header Part 1A shall contain the Part 4 flag (Part4Flag) and shall indicate the presence or absence of optional Segment Header Part 4:

– ‘1’ shall indicate that Segment Header Part 4 is present;

– ‘0’ shall indicate that Segment Header Part 4 is absent.

4.2.2.3 Segment Header Part 1B

4.2.2.3.1 When EndImgFlag (4.2.2.2.2) in Segment Header Part 1A is set to ‘1’, indicating that the coded segment ~~is~~corresponds to the last segment of the image, Segment Header Part 1B shall be included.

4.2.2.3.2 Bits 0-2 of Segment Header Part 1B shall contain the PadRows field. The value of the three-bit PadRows field shall be encoded as an unsigned binary integer and shall equal the number of ‘padding’ rows to be deleted (if any) after the inverse DWT is performed (see 3.2).

4.2.2.3.3 Bits 3-7 of Segment Header Part 1B are reserved for future use by the CCSDS and shall be set to ‘0’.

4.2.3 SEGMENT HEADER PART 2

4.2.3.1 General

If used, the optional Segment Header Part 2 shall specify output options that control the tradeoff between compressed data volume and reconstructed image quality for a segment.

NOTE – Table 4-5 summarizes the contents of Segment Header Part 2. Figure 4-5 illustrates the Segment Header Part 2 structure.

Table 4-5: Contents of Segment Header Part 2

Field	Width (bits)	Description	Details
SegByteLimit	27	Maximum number of ~~compressed~~ bytes in a coded segment.	value (mod 2^{27}) encoded as an unsigned integer
DCStop	1	Indicates whether ~~compressed output~~coded segment stops after coding of quantized DC coefficients (4.3).	1: ~~Stop coding~~Terminate coded segment after coding quantized DC coefficient information (see 4.3.2) and additional DC bit planes (see 4.3.3) 0: ~~Coding stop~~Coded segment termination determined by BitPlaneStop and StageStop values
BitPlaneStop	5	Unused when DCStop = 1. When DCStop = 0, indicates limit on coding of DWT coefficient bit planes. When BitPlaneStop = *b* and StageStop = *s*, ~~compressed output stops~~coded segment terminates once stage *s* of bit plane *b* has been completed (see 4.5), unless ~~coding stops~~coded segment terminates earlier ~~in the segment~~ because of the coded segment byte limit (SegByteLimit).	Bit plane index value BitPlaneStop encoded as an unsigned integer
StageStop	2		00: stage 1 (see 4.5.3) 01: stage 2 (see 4.5.3) 10: stage 3 (see 4.5.3) 11: stage 4 (see 4.5.4)
UseFill	1	Specifies whether fill bits will be used to produce SegByteLimit bytes in each coded segment.	1: fills bits used whenever needed to produce SegByteLimit bytes in each coded segment 0: fills bits not allowed
Reserved	4	Reserved for future use.	0000

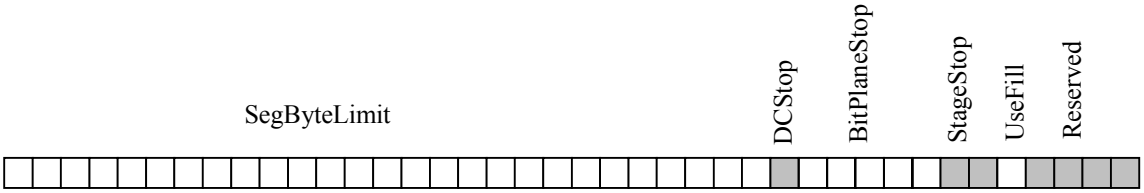

Figure 4-5: Segment Header Part 2

4.2.3.2 Contents of Segment Header Part 2

4.2.3.2.1 Bits 0-26 of Segment Header Part 2 shall contain the SegByteLimit field:

a) the value of the SegByteLimit field shall be encoded as an unsigned binary integer and shall equal the maximum number of bytes that can be used in a coded segment, including bytes used for the header;

NOTE – A SegByteLimit value less than 9 before the last segment and 10 for the last segment would not be permitted because of the combined number of bytes in the header Parts 1 and 2. The value of SegByteLimit must be an integer multiple of the word size specified by CodeWordLength in 4.2.5.2.8.

b) the value shall apply to the current coded segment and all subsequent coded segments until a new value of SegByteLimit is encoded in Segment Header Part 2 of a later segment header.

4.2.3.2.2 Bit 27 of Segment Header Part 2 shall contain the DCStop flag. The DCStop flag shall indicate the end of ~~compressed output~~the coded segment:

a) '1' shall indicate that ~~compressed output~~the coded segment terminates once the quantized DC coefficient values (see 4.3.2) and additional bit planes of DC coefficients (see 4.3.3) are encoded;

b) '0' shall indicate that end of ~~compressed output~~the coded segment shall be determined by a bit plane index (BitPlaneStop) and a coding stage within the coding of that bit plane (StageStop).

NOTE – Coded segment data stops at the indicated point in the BPE coding process, or when the coded segment byte limit is reached, whichever comes first. ~~Compressed data output~~The coded segment always includes at least the coded quantized DC coefficients values and any additional DC bit planes of DC coefficients (4.3.2 and 4.3.3) unless the value of SegByteLimit is too small to allow it.

4.2.3.2.3 Bits 28-32 of Segment Header Part 2 shall contain the BitPlaneStop field:

a) the 5-bit BitPlaneStop field shall be encoded as an unsigned binary integer and shall indicate the limit on coding of DWT coefficient bit planes;

b) when DCStop is set to '0', the value of BitPlaneStop shall equal the index value of the bit plane in which coding stops, unless coding stops earlier because SegByteLimit is reached first;

c) when DCStop is set to '1' the value of BitPlaneStop and the value of StageStop shall be ignored.

NOTE – When the user supplied BitPlaneStop value is greater than BitDepthAC-1, ~~compression~~the coded segment will terminate after ~~completing coding of~~ the quantized DC coefficient values and any additional DC bit planes (4.3.2 and 4.3.3) unless the SegByteLimit is too small to allow it.

4.2.3.2.4 Bits 33-34 of Segment Header Part 2 shall contain the StageStop field. The two-bit StageStop field shall indicate the stage at which the ~~coding stops~~coded segment terminates in the bit plane indicated by BitPlaneStop:

– '00' shall indicate stage 1 (see 4.5.3);

– '01' shall indicate stage 2 (see 4.5.3);

– '10' shall indicate stage 3 (see 4.5.3);

– '11' shall indicate stage 4 (see 4.5.4).

NOTE – ~~Compression~~Coded segment size is limited entirely by the coded segment byte limit when DCStop is 0, BitPlaneStop is 0, and StageStop is set to '11' (i.e., stage 4). In this case, lossless compression is achieved for a segment when the Integer DWT is used and the ~~compressed data for the~~coded segment requires less than SegByteLimit bytes.

4.2.3.2.5 Bit 35 of Segment Header Part 2 shall contain the UseFill field, which shall indicate whether fill bits will be used to produce SegByteLimit bytes in each coded segment:

a) if the value of UseFill field is '1', fill bits are used whenever needed to produce SegByteLimit bytes in each coded segment;

b) if the value of UseFill field is '0':

1) fill bits are not used to produce SegByteLimit bytes, and the number of ~~compressed~~ bytes in a coded segment may be less than SegByteLimit bytes;

2) fill bits are used to fill to the next word boundary, when the stopping point (according to the specified values of DCStop, BitPlaneStop, StageStop) is reached before the byte limit specified in SegByteLimit;

NOTE – A word corresponds to the unit in which the compressor produces output, which may be ~~a single byte, two bytes, or four~~from one to eight bytes. For example, a compressor that produces 4 output bytes at a time might include as many as 31 fill bits in the last word of the ~~compressed~~coded segment. The word length of the compressor is signaled by the field CodeWordLength in Segment Header Part 4 (see 4.2.5).

c) fill bits shall be set to 'all zeros'.

4.2.3.2.6 Bits 36-39 of Segment Header Part 2 are reserved for future use by the CCSDS and shall be set to 'all zeros'.

4.2.4 SEGMENT HEADER PART 3

4.2.4.1 General

4.2.4.1.1 If used, Segment Header Part 3 shall specify the segment size in terms of number of blocks, *S*, defined in 4.1 and indicate whether the heuristic parameter selection approach (4.3.2) is used in the coding of quantized DC coefficients (see 4.3) and/or AC coefficient bit depths (see 4.4).

4.2.4.1.2 The information encoded in this part of the header is permitted to change with each coded segment.

NOTE – Table 4-6 summarizes the contents of Segment Header Part 3. Figure 4-6 illustrates the Segment Header Part 3 structure.

Table 4-6: Contents of Segment Header Part 3

Parameter	Width (bits)	Description	Details
S	20	segment size in blocks	Value encoded, mod 2^{20}, as an unsigned binary integer
OptDCSelect	1	Specifies whether optimum or heuristic method is used to select value of *k* parameter for coding quantized DC coefficient values (see 4.3.2)	1: optimum selection of *k* 0: heuristic selection of *k*
OptACSelect	1	Specifies whether optimum or heuristic method is used to select value of *k* parameter for coding BitDepthAC (see 4.4)	1: optimum selection of *k* 0: heuristic selection of *k*
Reserved	2	Reserved	00

NOTE – Table 4-7 summarizes the contents of Segment Header Part 4. Figure 4-7 illustrates the Segment Header Part 4 structure.

Table 4-7: Contents of Segment Header Part 4

Parameter	Width (bits)	Description	Details
DWTtype	1	Specifies DWT type	0: Float DWT 1: Integer DWT
Reserved	~~2~~1	Reserved for future use	~~0~~0
ExtendedPixelBitDepthFlag	1	Indicates an input pixel bit depth larger than 16.	0: pixel bit depth is not larger than 16 1: pixel bit depth is larger than 16
SignedPixels	1	Specifies whether input pixel values are signed or unsigned quantities	0: unsigned 1: signed
PixelBitDepth	4	~~Specifies~~Together with ExtendedPixelBitDepthFlag, indicates the input pixel bit depth	~~Specifies values from 1 to 16 by mod(16) of 1, 2, 3, …, 16~~Input pixel bit depth value encoded, mod 16, as an unsigned binary integer
ImageWidth	20	image width in pixels	Value encoded, mod 2^{20}, as an unsigned binary integer
TransposeImg	1	Indicates whether entire image should be transposed after reconstruction	0: do not transpose image 1: transpose image
CodeWordLength	~~2~~3	Indicates the coded word length	000: 8-bit word 010: 16-bit word 100: 24-bit word 110: 32-bit word 001: 40-bit word 011: 48-bit word 101: 56-bit word 111: 64-bit word
~~Reserved~~	~~1~~	~~Reserved for future use~~	~~0~~
CustomWtFlag	1	Indicates if weights in 3.9 used or user defined	0: weights in 3.9 used 1: user-defined weights
$CustomWtHH_1$	2	Weight of HH_1 subband	00: weight = 2^0 01: weight = 2^1 10: weight = 2^2 11: weight = 2^3 (These fields are set to 00 when CustomWtFlag is 0)
$CustomWtHL_1$	2	Weight of HL_1 subband	
$CustomWtLH_1$	2	Weight of LH_1 subband	
$CustomWtHH_2$	2	Weight of HH_2 subband	
$CustomWtHL_2$	2	Weight of HL_2 subband	
$CustomWtLH_2$	2	Weight of LH_2 subband	
$CustomWtHH_3$	2	Weight of HH_3 subband	
$CustomWtHL_3$	2	Weight of HL_3 subband	
$CustomWtLH_3$	2	Weight of LH_3 subband	
$CustomWtLL_3$	2	Weight of LL_3 subband	
Reserved	11	Reserved for future	00000000000

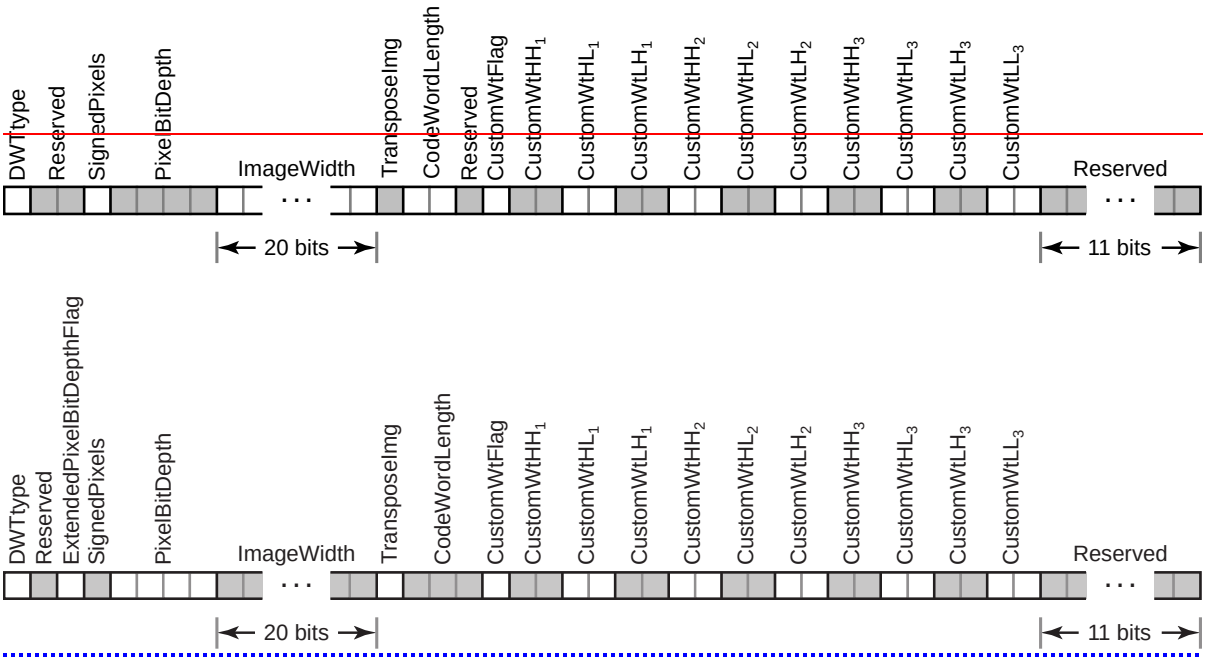

Figure 4-7: Segment Header Part 4

4.2.5.2 Segment Header Part 4 Contents

4.2.5.2.1 Bit 0 of Segment Header Part 4 shall contain the DWT Type field (DWTtype) and shall indicate the DWT type as follows:

– '0' shall indicate Float DWT;

– '1' shall indicate Integer DWT.

4.2.5.2.2 Bit~~s~~ 1~~–2~~ of Segment Header Part 4 ~~are~~is reserved for future use by the CCSDS and shall be set to ~~'all zeros'~~'0'.

4.2.5.2.3 Bit 2 of Segment Header Part 4 shall contain the Extended Pixel Bit Depth Flag field (ExtendedPixelBitDepthFlag) and shall indicate whether the input pixel bit depth is larger that 16 bits as follows:

– '0': Input pixel bit depth is not larger than 16 bits.

– '1': Input pixel bit depth is larger than 16 bits.

NOTE – The five-bit word formed by concatenating the Extended Pixel Bit Depth Flag and Pixel Bit Depth fields is equal to the unsigned binary representation of the input pixel bit depth, except when the bit depth is 16.

4.2.5.2.4 Bit 3 of Segment Header Part 4 shall contain the Signed Pixels field (SignedPixels) and shall indicate whether input pixel values are signed or unsigned quantities:

– ‘0’ shall indicate unsigned;

– ‘1’ shall indicate signed.

4.2.5.2.5 Bits 4-7 of Segment Header Part 4 shall contain the Pixel Bit Depth field (PixelBitDepth). The four-bit Pixel Bit Depth field shall be encoded as an unsigned binary integer ~~and shall~~ equal to the pixel bit depth mod 16.

4.2.5.2.6 Bits 8-27 of Segment Header Part 4 shall contain the Image Width field (ImageWidth). The 20-bit Image Width field shall be encoded as an unsigned binary integer and shall equal the width of the image in pixels mod 2^{20}.

4.2.5.2.7 Bit 28 of Segment Header Part 4 shall contain the Transpose Image field (TransposeImg) and shall indicate whether the entire image should be transposed after reconstruction:

– ‘0’ shall indicate that the image should not be transposed;

– ‘1’ shall indicate that the image should be transposed.

4.2.5.2.8 Bits 29-~~30~~31 of Segment Header Part 4 shall contain the Code Word Length field (CodeWordLength) and shall indicate the length of the code word as follows:

– ‘000’: 8-bit word;

– ‘010’: 16-bit word;

– ‘100’: 24-bit word;

– ‘110’: 32-bit word;

– ‘001’: 40-bit word;

– ‘011’: 48-bit word;

– ‘101’: 56-bit word;

– ‘111’: 64-bit word.

NOTE – The previous issue of this Recommended Standard only allowed up to 32-bit words for CodeWordLength, and thus used only 2 bits for this field.

~~**4.2.5.2.9** Bit 31 of Segment Header Part 4 is reserved for future use by the CCSDS and shall be set to ‘0’.~~

4.2.5.2.9 Bit 32 of Segment Header Part 4 shall contain the Custom Weights flag (CustomWtFlag) and shall indicate whether standard (3.9) or user-defined weights are used:

c) Otherwise, the sequence of BitDepthAC_Block$_m$ values for the segment shall be coded using the same differencing and variable-length coding procedure specified for coding quantized DC values described in 4.3.2. However, a few parameters differ for the case of coding the BitDepthAC_Block$_m$ values:

1) N shall equal the number of bits required to represent the magnitude of BitDepthAC for the segment, as in equation 21:

$$N = \lceil \log_2(1 + \text{BitDepthAC}) \rceil \quad (21)$$

For code option identification, table 4-9 shall be used with this value of N.

NOTE – For ~~a 16-bit~~all source images of supported bit depth, no more than five bits are needed to specify the value of BitDepthAC, i.e., $N \leq 5$.

2) Since the BitDepthAC_Block$_m$ values are necessarily nonnegative, $x_{\min}$ and $x_{\max}$ shall be redefined as given in equation 22 and used to map successive differences to nonnegative integers:

$$x_{\min} = 0, \quad x_{\max} = 2^N - 1 \quad (22)$$

NOTE – The coded bit string for the BitDepthAC_Block$_m$ values follows the same data format illustrated in figures 4-8 and 4-9.

4.5 BIT PLANE CODING

4.5.1 OVERVIEW

Coding of a bit plane is performed in *stages* numbered 0-4. The coded bits produced at the stages for each block are interleaved, as illustrated in figure 4-2(b) and figure 4-10. Thus, a coded bit plane first consists of all the stage 0 bits (if any) in the segment, then all of the coded stage 1 bits in the segment, and so on, finishing with all of the encoded stage 4 bits in the segment. This produces an embedded bit string with information from the highest bit plane of all S blocks in the first part of the output bit string followed by information from lower bit planes, and allows progressive decoding of the coded string. This improves image reconstruction quality when the coded bit sequence is truncated.

Stage 0 bits from each block in the segment (if any)
Coded stage 1 from block 0, 1, …, S-1
Coded stage 2 from block 0, 1, …, S-1
Coded stage 3 from block 0, 1, …, S-1
Coded stage 4 from block 0, 1, …, S-1

Figure 4-10: Coded Bit Plane Structure for a Coded Segment

Note that when the index b of the bit plane being coded is larger than or equal to the AC bit depth of the block, then there is nothing to code for the block.

The layered coding stages in figure 4-10 inherently allow a controlled amount of DWT coefficient information to be encoded within a bit plane, permitting image quality control at sub-bit plane levels. The tradeoff between reconstructed image quality and compressed data volume for each segment is controlled by specifying the first four parameters in Part 2 Header (table 4-5). Quality level and data volume are both specified for each segment, and ~~compressed output~~data for a coded segment ~~is~~are produced until either the quality limit or the volume limit is reached. Users may choose to achieve fixed output rate by using fill bits. These features are all described in 4.2.3.

Annex C contains a list of symbols used in the various coding stages specified in 4.5.2, 4.5.3, and 4.5.4.

4.5.2 OVERVIEW OF CODING STAGES

The coding stages for a block at bit plane b are described in the following paragraphs.

Stage 0 for a block consists of at most a single bit, which is simply the b^{th} most significant bit of the two's-complement representation of the DC coefficient. Note that whenever the bit plane index b satisfies $b \geq q$, this bit value is already known from the DC coefficient information already encoded, and in this case, stage 0 is empty, i.e., no bits are coded in stage 0. Stage 0 is also empty when scaling of the DC coefficient assures that the bit must be zero, i.e., when $b <$ BitShift(LL_3).

The remaining stages (1-4) encode AC coefficient bits. The stage in which bits from AC coefficients in a bit plane are coded depends on the *type* of the AC coefficient at the bit plane, which we now define. At bit plane b, the type of an AC coefficient x, denoted $t_b(x)$, has one of the following values:

- $t_b(x) = 0$ if $|x| < 2^b$, (x is not due for selection at this bit plane);
- $t_b(x) = 1$ if $2^b \leq |x| < 2^{b+1}$, (x is due for selection at this bit plane);
- $t_b(x) = 2$ if $2^{b+1} \leq |x|$, (x has already been selected at a previous bit plane);
- $t_b(x) = -1$ if $b <$ BitShift(Γ), (x must be zero at this bit plane due to subband scaling).

Here, Γ denotes the subband containing x. Thus, during bit-plane encoding, each AC coefficient typically proceeds from type 0 to 1, to 2, to -1. For a set of coefficients Ψ, we define $t_{\max}(\Psi)$ as the maximum of the coefficient types in Ψ.

An AC coefficient x is said to be *selected* at bit plane b if $t_b(x) = 1$. I.e., the 'selection' of a coefficient marks the first bit plane where a non-zero magnitude bit is encoded for the

b) **Stage 2 (children)**:

1) $tran_B$;

2) $tran_D$, if $tran_B \neq 0$ and $t_{max}(B) \neq -1$;

3) ~~$types_b[C_i]$ and $signs_b(C_i)$ for each i such that $t_{max}(D_i) \neq 0, -1$~~ $types_b[C_i]$ and $signs_b[C_i]$ for each i such that $t_{max}(D_i) > 0$ in current or any more significant bit planes.

c) **Stage 3 (grandchildren)**:

If $tran_B = 0$ or $t_{max}(B) = -1$, then stage 3 is unnecessary and shall be omitted. Otherwise stage 3 consists of:

1) $tran_G$;

2) $tran_{Hi}$, for each i such that $t_{max}(G_i) \neq 0, -1$;

3) $types_b[H_{ij}]$ and $signs_b[H_{ij}]$ for each i such that $t_{max}(G_i) \neq 0, -1$ and each j such that $t_{max}(H_{ij}) \neq 0, -1$.

NOTE – All of the words generated in the above stages are variable length (including the null word).

4.5.3.1.9 Words $types_b[P]$, $types_b[C_i]$, $types_b[H_{ij}]$, $tran_D$, $tran_G$, $tran_{Hi}$ shall be entropy coded, i.e., each shall be replaced with a corresponding variable-length codeword, whenever such a word has a length of at least 2 bits.

NOTE – The sign bit words are not coded further, because AC coefficients are generally positive and negative with about equal probability. The $tran_B$ word is always, at most, one bit in length and is never entropy coded.

4.5.3.2 Mapping Words to Symbols

4.5.3.2.1 The entropy coding procedure used to encode the words $types_b[P]$, $types_b[Ci]$, $types_b[Hij]$, $tran_D$, $tran_G$, $tran_{Hi}$ shall be accomplished through the use of variable-length codes given in 4.5.3.3. Words having a length of one bit, and sign-bit words, shall be included in the compressed data stream without further coding. Words of length greater than one bit shall be coded in the sequence in which they occur within each stage with the entropy coding method described in 4.5.3.3.

NOTE – Certain bit sequences cannot appear as values for certain words and this fact is taken into account in the entropy coding process. For example, $tran_D$ can never equal 000, because this condition would be inferred from the fact that $tran_B$=0. Table 4-11 summarizes the maximum word lengths and impossible values for each word that is entropy coded.

4.5.4.2 For each block, the output bit string shall consist of the b^{th} magnitude bit of type 2 coefficients, in the following order:

- p_i, for each $i = 0,1,2$;
- members of C_i, for each $i = 0,1,2$;
- members of H_{ij}, for each $i = 0,1,2$, and each $j = 0,1,2,3$.

4.5.4.3 Members of the sets C_i and H_{ij} shall be processed in the order listed in table 4-1. No bits shall be coded in stage 4 for AC coefficients x not of type 2 ($t_b(x) \neq 2$).

4.5.4.4 The resulting strings for all blocks in the segment shall be concatenated to produce the entire stage 4 output string for the coded segment.

HL_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal, and low-pass 1-d DWT filter to vertical lines of image; for n>1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal, and low-pass 1-d DWT filter to vertical lines of subband LL_{n-1}.
HH_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal and vertical lines of image; for n>1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal and vertical lines of subband LL_{n-1}.
weight factors	after transforming image data by means of the 9/7 integer DWT, the obtained wavelet coefficients need to be multiplied by these numbers before encoding with the BPE. One weight factor is defined for each subband.
DC coefficient	any wavelet coefficient from the lowest frequency subband LL_3.
AC coefficient	any wavelet coefficient from any subband except LL_3.
block	Collection of 64 wavelet coefficients, consisting of one DC coefficient from the LL_3 subband and 63 uniquely associated AC coefficients from the remaining nine subbands (see figure 4-1 and table 4-1). For encoding, the complete wavelet domain is divided into blocks that are pairwise disjoint. There is one block for each DC coefficient.
~~segment~~	~~Bitstream of compressed code consisting of a data field headed by a segment header. The segment header is defined in 4.2. The data field contains the encoded bits from S consecutive blocks. S is a user-selected parameter such that $16 \leq S \leq 2^{20}$.~~
segment	A group of S consecutive blocks. S is a user-selected parameter in the range $16 \leq S \leq 2^{20}$, except for the last segment of an image, when $1 \leq S \leq 2^{20}$.
coded segment	Bitstream of compressed code consisting of a data field preceded by a segment header. The segment header is defined in 4.2. The data field contains the encoded bits corresponding to a segment of the image.
gaggle	A gaggle consists of a set of consecutive blocks within a segment. Specifically, a segment is partitioned into gaggles, with each gaggle consisting of 16 blocks, except possibly the last gaggle, which contains (S mod 16) blocks when S is not a multiple of 16 (see 4.3.2.5). Certain code parameters are independently selected for each gaggle. Specifically, all of the quantized DC coefficient values in a gaggle are coded using the same code option (see 4.3.2.6). Similarly,

ANNEX B

INFORMATIVE REFERENCES

(Informative)

[B1] *Image Data Compression*. Report Concerning Space Data System Standards, CCSDS 120.1-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, June 2007.

[B2] *Lossless Data Compression*. Recommendation for Space Data Systems Standards. CCSDS 121.0-B-1. Blue Book. Issue 1. Washington D.C.: CCSDS, May 1997.

[B3] *Space Packet Protocol*. Recommendation for Space Data Systems Standards. CCSDS 133.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

[B4] *CCSDS File Delivery Protocol (CFDP)*. Recommendation for Space Data System Standards, CCSDS 727.0-B-4. Blue Book. Issue 4. Washington, D.C.: CCSDS, January 2007.

[B5] *AOS Space Data Link Protocol*. Recommendation for Space Data System Standards, CCSDS 732.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, July 2006.

[B6] *Information Technology—JPEG 2000 Image Coding System: Core Coding System*. International Standard, ISO/IEC 15444-1:2004. 2nd ed. Geneva: ISO, 2004.

[B7] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

[B8] *Spectral Pre-Processing Transform for Multispectral and Hyperspectral Image Compression*. Draft Recommendation for Space Data System Standards, CCSDS 122.1-R. Forthcoming.