SMPTE STANDARD

VC-5 Video Essence – Part 6: Sections

Table of Contents

Approved October 23, 2015

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 10E.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

This section is entirely informative and does not form an integral part of this Engineering Document.

This document defines how to subdivide a bitstream into sections.

Sections can enable more sophisticated processing by a decoder, including skipping a portion of the bitstream that does not need to be decoded, detecting and correcting errors in the bitstream by application-defined algorithms, or concurrently decoding portions of the bitstream.

Sections are backward compatible with decoders that do not support sections.

1 Scope

This standard defines extensions to SMPTE ST 2073-1, ST 2073-3, and ST 2073-4 to support sections in a VC-5 bitstream.

Sections subdivide the bitstream to enable advanced decoder features such as fast seeking within the bitstream, error detection and correction, multi-resolution decoding, and concurrent decoding.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; then formal languages; then figures; and then any other language forms.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this engineering document. At the time of publication, the editions indicated were valid. All standards are subject

to revision, and parties to agreements based on this engineering document are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE ST 2073-1:2014 VC-5 Video Essence. Part 1: Elementary Bitstream.

SMPTE ST 2073-3:2015 VC-5 Video Essence. Part 3: Image Formats.

SMPTE ST 2073-4:2015 VC-5 Video Essence. Part 4: Subsampled Color Difference Components.

SMPTE ST 2073-5:2015 VC-5 Video Essence. Part 5: Layers.

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

4.1 Image Bitstream

bitstream that is compliant with SMPTE ST 2073-1, ST 2073-3, ST 2073-4, and ST 2073-5, excluding the **StartMarkerSegment**

4.2 VC-5 Standard

SMPTE standards designated ST 2073

5 Section Syntax and Semantics

5.1 Section Syntax

A section element shall be the contiguous portion of the bitstream that comprises the section header and the section body.

The section header shall consist of one segment that comprises an 8-bit integer tag followed by three bytes that are used to compute the length of the section body. The first two bytes of the section header shall comprise a negative 16-bit two's complement integer. The two's complement negation of that 16-bit integer shall comprise:

(1) A one byte tag that identifies the type of section element as defined in Annex A,

(2) The most significant byte in the three bytes (including the last two bytes in the section header) that comprise a 24-bit unsigned integer that is the length of the section body in segments.

The section body shall consist of the contiguous sequence of segments that immediately follow the section header.

The syntax of a section element is shown in Figure 1.

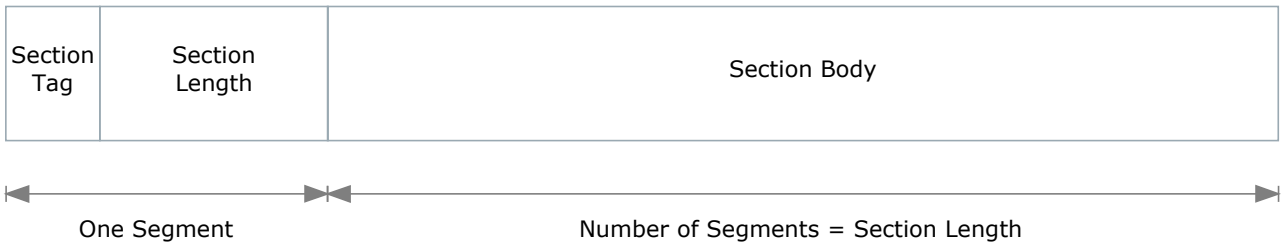

Figure 1. Diagram that illustrates the syntax of a section element.

5.2 Decoder Compatibility (Informative)

The section header mimics an optional tag-value pair so that decoders that do not support sections, or the particular section type, will skip the section header and process the section body as usual.

5.3 Section Hierarchy

Section elements listed in Table 1 shall define a nested hierarchy of bitstream structures such that the body of a section element:

1. Shall not contain syntax elements that are required to decode another section of the same type,
2. Shall comprise all syntax elements required to decode that section, and
3. May contain syntax elements that are required to decode a different type of section.

5.4 Section Independence

The portion of the bitstream after the body of a section shall be decodable using any of:

1. The codec state immediately before the section or
2. The codec state immediately after the section.

NOTE: This requirement enables concurrent decoding of sections.

NOTE: The ability to decode sections independently implies that decoders cannot rely on the stickiness of codec state parameters set by tag-value pairs, so the encoder inserts tag-value pairs to set the codec state parameters at the beginning of the section body.

5.5 Bitstream Syntax (Informative)

For the purposes of compliance with SMPTE ST 2073-1 section 8.5, the section header for image bitstream sections and bitstream header sections, are considered to be optional bitstream header parameters.

6 Section Decoding (Informative)

When a decoder encounters a section header, it processes the section header as if it is an optional tag-value pair: The decoder negates the first two bytes in the section header and checks whether the decoder supports that two-byte tag:

1. If the decoder does not support sections:

    1.1. The tag is not recognized as a valid tag.

    1.2. The tag-value pair is optional so the segment that comprises the section header is ignored.

    1.3. The bitstream after the segment that comprises the section header is processed as usual.

2. If the decoder supports sections:

    2.1. The decoder interprets the first byte in the section header as a section tag.

    2.2. If the decoder supports that type of section:

        2.2.1. The decoder computes the length of the section body (section 4.1).

        2.2.2. The decoder uses the presence of the section for whatever purpose it implements.

        2.2.3. The decoder processes the section body as usual.

    2.3. If the decoder does not support that type of section:

        2.3.1. The decoder ignores the section header.

        2.3.2. The decoder processes the section body as usual.

7 Bitstream Structure

7.1 Image Bitstream Section

The body of an image bitstream section shall contain all syntax elements that comprise the single image bitstream in that section body and no syntax elements that comprise any other image bitstream. See Figure 2 for an illustration of how to represent multiple images in a bitstream using image bitstream sections.

NOTE: Image bitstream section elements can be used to extend SMPTE ST 2073-1 to allow more than one image bitstream to be contained in a bitstream.

NOTE: An image bitstream is not a layer since layers share a common bitstream header (see section 6.3).

If more than one image bitstream is present in the bitstream, then each image bitstream shall be the body of an image bitstream section.

The **ImageCount** codec state parameter shall be the number of image bitstream sections in the bitstream.

The **ImageCount** codec state parameter can be provided as a default codec state parameter, an external codec state parameter, or explicitly in the bitstream (see ST 2073-1 section 9.1).

If present in the bitstream, the **ImageCount** codec state parameter shall be present in the bitstream header for every image bitstream section in the bitstream.

The bitstream header in each image bitstream section shall contain a tag-value pair for the **ImageNumber** codec state parameter. The image number assigned to each image bitstream section shall be unique. The tag-value pair for the section header of an image bitstream section shall use the tag **ImageSectionTag**.

The body of an image bitstream section does not contain the **StartMarkerSegment**.

NOTE: Image bitstream sections need not be present in the bitstream in order by image number.

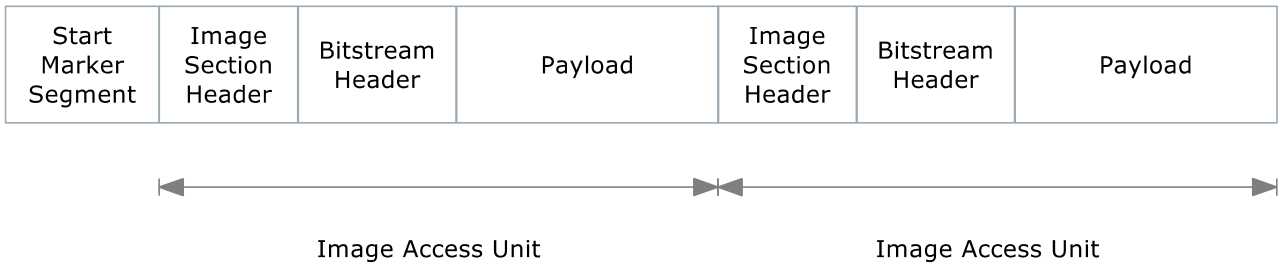

Figure 2. Diagram that illustrates how an image bitstream section delineates individual image bitsteams.

7.2 Bitstream Header Section

If a header section is used in the bitstream representation of an image, then the body of the header section shall include all tag-value pairs in the bitstream header that are present in the image bitstream.

The tag-value pair for the section header of a header section shall use the tag **HeaderSectionTag**.

7.3 Layer Section

The body of a layer section shall contain all syntax elements, except for the bitstream header, that represent the single layer in that section body and no syntax elements that represent any other layer.

The tag-value pair for the section header of a layer section shall use the **LayerSectionTag**.

7.4 Channel Section

The body of a channel section shall contain all syntax elements that represent the single channel in that section body and no syntax elements that represent any other channel.

The tag-value pair for the section header of a channel section shall use the tag **ChannelSectionTag**.

7.5 Wavelet Section

The body of a wavelet section shall contain all syntax elements that represent the single wavelet in that section body and no syntax elements that represent any other wavelet.

The tag-value pair for the section header of a wavelet section shall use the tag **WaveletSectionTag**.

7.6 Subband Section

The body of a subband section shall contain all syntax elements that represent the single subband in that section body and no syntax elements that represent any other subband.

The tag-value pair for the section header of a subband section shall use the tag **SubbandSectionTag**.

8 Section Use Cases (Informative)

8.1 Concurrent Decoding

Section elements can enable a decoder to process portions of the bitstream concurrently.

When a section element is encountered in the bitstream and the decoder implementation supports concurrent decoding of that section type, the decoder can copy the codec state and begin decoding the section body on a separate thread using the copy of the codec state. The remainder of the bitstream after the section body does not depend on any changes made to the copy of the codec state while decoding the section body (section 4.4), so the decoder can process the remainder of the bitstream on a different thread.

When a section element is encountered in the bitstream and the decoder implementation does not recognize the section element or does not support concurrent decoding of that section element, the decoder can ignore the tag-value pair for the section header and continue to process the body of the section as it would normally process the bitstream. After the section body is processed, the decoder continues to process the bitstream as usual. Any changes made to the codec state while processing the section body will not affect decoding the remainder of the bitstream after the section body.

EXAMPLE: Suppose that the segments comprising each layer are enclosed in a layer section (section 5.3). The decoding process can assign a different thread to process each layer while the decoding process running on the main thread skips over the section body and continues to process the remainder of the bitstream concurrently. As each layer section is encountered in the bitstream, the decoding process passes a copy of the current codec state to the thread assigned to decode that layer. Each thread has exclusive access to its copy of the codec state and can modify its copy of the codec state as usual. The codec state copy is discarded after the body of the layer section has been processed.

8.2 Bitstream Seeking

Section elements can enable a decoder to seek within the bitstream. Suppose that the decoder needs to seek to the second channel in the third layer in the bitstream. When the decoder encounters a section element, it can skip the body of the section. The body of a layer section cannot contain another layer section, so the decoder can skip the body of the first two layers in the bitstream. The decoder must process the third layer, but can skip the first channel section in the third layer.

Without sections, the decoder can still seek in the bitstream, but must read each tag-value pair in bitstream order. However, the decoder can skip over chunk payloads, including codeblock elements, since chunks do not change codec state parameters.

8.3 Error Detection and Correction

A codec implementation can provide a facility for error detection by using sections to delineate a portion of the bitstream used to compute a checksum.

An optional tag-value pair can be used to contain a checksum in the value of that tag-value pair. An optional chunk element can be used to contain a longer checksum or information for error detection or correction in the chunk payload. Decoders that do not recognize the syntax element containing error detection or correction information can skip that information and process the bitstream.

EXAMPLE: A section element can be used to delineate a contiguous sequence of segments in the bitstream. The body of the section can include a syntax element containing the checksum that was computed from the segments that comprise the section body. A decoder that recognizes the error detection scheme can compute the checksum as it processes the section body and compare the checksum with the value read from the bitstream.

If the syntax element that contains the checksum is the first syntax element in the body of the section, then the syntax element containing the checksum will signal the decoder to begin computing the checksum.

8.4 Multi-Resolution Decoding

The component arrays can be decoded at a smaller size than the size of the image represented in the bitstream by not decoding the portion of the bitstream associated with the larger wavelets (higher subband numbers).

The component arrays can be decoded at half, quarter, or one-eighth resolution:

Half resolution: The width (height) of the component arrays output by the decoder is half the width (height) of the encoded width (height),

Quarter resolution: The width (height) of the component arrays output by the decoder is one-quarter the width (height) of the encoded width (height),

One-eight resolution: The width (height) of the component arrays output by the decoder is one-eighth the width (height) of the encoded width (height).

Sections can allow more efficient lower-resolution decoding by delineating the portion of the bitstream that is skipped when decoding to a lower resolution.

Annex A Codec State Parameters

A.1 Bitstream Extensions

The set of codec state parameters defined in the VC-5 standard is extended by the codec state parameters defined in Annex A.2 of this standard.

The set of tag-value pairs defined in the VC-5 standard is extended by the tag-value pairs defined in Annex A.3 of this standard.

A.2 Parameter Descriptions

ImageCount

The **ImageCount** codec state parameter shall specify the number of image bitstream sections in the bitstream. The minimum value shall be 1.

If a tag-value pair for the **ImageCount** codec state parameter is present in any image bitstream section, then one tag-value pair for the **ImageCount** codec state parameter shall be present in every image bitstream section in the bitstream.

ImageNumber

The **ImageNumber** codec state parameter shall specify the number of the image bitstream section that contains the bitstream header containing the tag-value pair for the parameter.

The **ImageNumber** codec state parameter shall be in the range [0, **ImageCount**).

A.3 Tag-Value Pairs

The tag-value pairs that represent the codec state parameters defined by this standard in a VC-5 bitstream shall be as listed in Table 1, with the tag number and default value as listed in that table. The tag-value pairs defined by this standard shall be header parameters if so specified in Table 1.

Table 1. Codec state parameters defined by this standard.

Parameter Name	Tag Number	Default Value	Header Parameter?	Reference
ImageCount	130	1	Yes	A.2.1
ImageNumber	131	0	Yes	A.2.2

A.4 Section Tag Definitions

The tag numbers for the tag-value pairs that represent section headers for sections that define a hierarchy of bitstream structures (section 5) shall be as listed in Table 2.

NOTE: Section elements never have a default value and can never occur in a bitstream header.

Table 2. Tag definitions for section elements that define a hierarchy of bitstream structures.

Section Tag	Tag Number	Description
ImageSectionTag	0x27	Section element marking the beginning and extent of the portion of the bitstream containing the representation of an image
HeaderSectionTag	0x25	Section element marking the beginning and extent of the tag-value pairs that comprise the bitstream header
LayerSectionTag	0x26	Section element marking the beginning and extent of the portion of the bitstream containing all syntax elements that are unique to a single layer (includes all channels in the layer)
ChannelSectionTag	0x24	Section element marking the beginning and extent of the portion of the bitstream containing all syntax elements that are unique to a single channel (includes all wavelets in the channel)
WaveletSectionTag	0x21	Section element marking the beginning and extent of the portion of the bitstream containing syntax elements that are unique to a single wavelet (includes all subbands in the wavelet)
SubbandSectionTag	0x20	Section element marking the beginning and extent of the portion of the bitstream containing tag-value pairs and the codeblock for a single subband