SMPTE RECOMMENDED PRACTICE

VC-2 Conformance Specification

Table of Contents

Approved July 21, 2010

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 Part XIII of its Administrative Practices.

SMPTE Recommended Practice RP 2042-3 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 Standard. 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 document.

VC-2 is a wavelet-based, intra-frame video compression system aimed at professional applications that provides efficient coding at many resolutions including various flavors of CIF, SDTV and HDTV. The VC-2 standard defines a bit-stream syntax and a reference decoder specification. To aid the creation of new VC-2 streams, this recommended practice also provides details for the operation of a sample VC-2 stream generator.

This recommended practice includes a list of VC-2 compliant streams that can be used to test various aspects of VC-2 decoder compliance. This recommended practice also includes a reference decoder for comparison-testing of decoder implementations. The reference decoder may also be used to establish the compliance of arbitrary streams.

1 Scope

The VC-2 video compression standard is an intra-frame video compression system defined through the VC-2 stream syntax, entropy coding, a coefficient unpacking process and a picture decoding process. The decoder operations are defined by means of a mixture of pseudo-code and mathematical operations.

This recommended practice defines the materials, procedures, and criteria for verifying conformance of SMPTE VC-2 video decoders. The documents that form this specification include this document, a register of test streams, a sample VC-2 stream generator and a reference decoder.

The reference decoder and the sample stream generator are both software written in the C++ programming language and capable of being compiled to executable code on a variety of computer platforms. The reference decoder can be used to verify conformance of VC-2 decoder implementations against a register of test streams, and the conformance of arbitrary streams to the VC-2 specification.

This conformance practice defines both the format for the register of test streams and the lists of test streams.

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.

3 Normative Reference

Normative references are external documents referenced in normative text that are indispensable to the user. Bibliographic references are references made in informative text or are those otherwise not indispensable to the user.

The following standards contain provisions which, through reference in this text, constitute provisions of this recommended practice. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this recommended practice are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE ST 2042-1:2009, VC2 Video Compression

ISO/IEC 646:1991, Information Technology — ISO 7-Bit Coded Character Set for Information Interchange

4 Description of Materials

The VC-2 conformance specification shall comprise the following components:

1. VC-2 encoded video Files,
2. Reference decoder software,
3. Sample VC-2 stream generator software, and
4. Procedures for testing VC-2 streams and the reference decoder.

These components of the conformance specification shall be as defined in the following sections.

5 File Formats

This section defines the file formats used by this conformance specification. The specification defines two file formats:

- the file format for the storage of the coding parameters, and
- the file format for the storage of uncompressed video data.

The data pertaining to each frame in a VC-2 stream shall be stored as separate files to simplify data management.

A file belonging to a VC-2 stream `basename` shall have a filename of the form:

```
basename-frame_number.extension
```

where `frame_number` shall be the frame number in the VC-2 stream, and `extension` shall be a defined extension name that identifies the file storage format. The base name may be null, in which case the file name would be of the form frame_number.extension.

5.1 Coding Parameters File Format

All coding parameters for a given frame number shall be stored in a file whose format is defined as per Annex A. The `extension` part of the filename is recommended to end with "`.cparams`".

Note: The extension name is not required by the reference software.

5.2 Video Data File Format

The video data file format shall be used to store uncompressed video data.

Interlaced frames shall be stored in a pseudo-progressive manner, i.e., fields shall be interleaved such that a frame comprising two fields with no intermediate motion shall appear indistinguishable from a progressive frame.

The color difference format used for each frame shall be either 4:4:4, 4:2:2 or 4:2:0.

Each frame of Y, C1, C2 (as per the color model of Section 6.1 of the VC-2 specification) image data shall be stored as raw video data in a headerless planar format.

Data shall be serialized by raster scanning each component separately for each frame (i.e. a frame of Y, followed by a frame of C1, then a frame of C2).

Each sample of video data shall be stored as a 16 bit Big-Endian word, MSB justified and zero-padded. For example, an 8 bit sample of value 0x35, would be stored in a 16 bit word as 0x3500.

The `extension` part of the filename shall be used by the reference software to determine the desired input/output format. The `extension` value shall be the FOURCCode value that correctly represents the color difference format as defined in the table below:

Color difference Format	FOURCCode
4:4:4	`16P4`
4:2:2	`16P2`
4:2:0	`16P0`

Annex C illustrates the storage format for FOURCCode values `16P4`, `16P2` and `16P0`.

In this practice, required components of the command line application wrap the component with the angle bracket symbols < > and optional components of the command line application wrap the component with the square bracket symbols [ ].

6 Reference Decoder Software

The reference decoder shall be used for testing VC-2 stream conformance and shall implement the decoding processes defined in SMPTE ST 2042-1 [VC-2].

The reference decoder, in the form of platform independent C++ source code and associated documentation (instructions for building, installing and running the software), is available at http://store.smpte.org.

The decoder is a command line application that shall have the following invocation format:

```
vc2decoder <inputstream> <outputvideo> [codingparams]
```

where:

- ‘`inputstream`’ is the VC-2 stream filename to decode.
- '`outputvideo`' is the base of the filename to be used for storing decoded output video frames.

  Note: The actual name of the output file will comprise the base name concatenated with a frame number plus an appropriate file extension

- '`codingparams`' is an optional base file specification, as per annex A, to be used for storage of the coding parameters associated with each video frame as determined from the input stream. If this argument is not specified, no coding parameters files shall be produced.

  Note: The actual name of the file generated, if any, will comprise the base file name concatenated with a frame number plus “cparams” as a file extension

The decoder decoding the input VC-2 stream, shall produce one video frame and coding parameters file for each frame decoded.

The frame number for each output file shall start with frame number of the first file. The arguments '`outputvideo`' and '`codingparams`' shall use identical frame numbers.

Note: There is no implicit relationship between the output frame numbers and the stream parameter, `picture_number`.

The reference decoder shall handle multiple VC-2 sequences within a VC-2 stream by storing all frames from each sequence as presented in the VC-2 stream. The output frame numbers shall not be reset between sequences within a decoded VC-2 stream.

7 Sample VC-2 Stream Generator

The sample VC-2 stream generator may be used for the creation of VC-2 streams. The sample VC-2 stream generator provided with this conformance specification provides compliant VC-2 streams that may be used by VC-2 decoders for testing purposes but this generator does not represent a full implementation of a VC-2 encoder.

The sample VC-2 stream generator, in the form of platform independent C++ source code and associated documentation (instructions for installing, building and running the software), is available at http://store.smpte.org.

The sample VC-2 stream generator is a command line application that shall have the following invocation format:

`vc2streamgen <inputvideo> <codingparams> <outputstream>`, where:

- `<inputvideo>` shall be the base of the filename, as per Section 4, to be used for input video frames,
- `<codingparams>` shall be the base of the filename, as per Section 8, providing the coding parameters associated with each input video frame and
- `<outputstream>` shall be the required filename of the output VC-2 stream.

The sample stream generator encodes the input video and coding parameters using one video and one coding parameters file per frame to produce a VC-2 output stream.

The frame number shall start from zero for both types of input file.

Note: There is no implicit relation between the output frame numbers and the stream parameter `picture_number` since the `picture_number` value is determined from the coding parameters file.

Multiple VC-2 sequences may be encoded with a VC-2 stream by varying any value in the sequence parameter section in the relevant coding parameters file where the new sequence is desired. It is not permitted to change the sequence parameter section and not set "`signalled: true`" on the encoded frame with the first change.

8 Test VC-2 Streams

The register of VC-2 streams to be used for testing target decoder conformance is identified in Annex C. The set of test VC-2 streams is comprehensive but not intended to represent every possible combination of encoding parameters and tools specified in SMPTE ST 2042-1.

9 Conformance Testing Procedure

The following sub-sections define the normative tests for verifying conformance of a candidate VC-2 stream and decoder to SMPTE ST 2042-1, and provide an informative description of how an encoder implementation can be tested with the conformance tools provided.

9.1 Conformance Testing of A Candidate VC-2 Stream

Bit stream testing is a process intended to verify that a candidate bit stream is a valid VC-2 stream in accordance to SMPTE ST 2042-1.

A candidate bit stream shall be deemed a valid VC-2 stream if no fatal errors occur in the decoding of the bit stream with the reference decoder (Section 6).

Note: The video files output by the reference decoder do not form part of this test, however they can be used to check that the stream decompresses as intended. If the video output is different from that intended, the encoder implementation can be tested according to the procedure described in Section 9.3.

9.2 Conformance Testing of A Candidate Decoder

Decoder testing is a process intended to verify that a candidate decoder implementation conforms to the decoding process as specified in SMPTE ST 2042-1. Decoder testing uses the register of test VC-2 streams (Section 8).

It shall be noted that:

- A candidate decoder need not support all possible profiles and levels.
- Failure of any test is sufficient to identify the candidate decoder as non-conformant to a particular level and profile.
- Successful completion of all tests only represents a high confidence of conformance to a particular level and profile and cannot guarantee 100% conformance.

Decoder implementations shall specify which profiles and levels are supported and shall declare conformance only when all the test VC-2 streams for those specified profiles and levels have been successfully decoded.

Decoder conformance testing shall be conducted as follows:

For each VC-2 stream in the register:

1) Verify the VC-2 stream is valid as per Section 9.1. The video files output by this step shall be used in step 3. The coding parameter file output by this step shall be used in step 4, if applicable.

2) Decode the VC-2 stream with the decoder under test. The video files output by this step shall be used in step 3.

3) Compare the two sets of video files. If, after taking into account differences in output file formatting, the two sets are not identical, the test shall be failed.

4) Compare the output coding parameters (if available). The decoder under test may be able to produce metadata containing some or all of the coding parameters extracted from the stream (but is not required to do so). This metadata must be identical to the coding parameters produced by the reference decoder, after taking into account differences in output and any omissions by the decoder under test. If not, the test shall be considered failed.

9.3 Encoder Testing (Informative)

A candidate stream can be verified by the procedure set out in Section 9.1 as a valid VC-2 stream, yet the decoded video output might not match what was intended. This will arise from an error in the encoder which generated the stream.

Verification that an encoder is producing a correct VC-2 stream, given the coding parameters it has selected or been given, is possible using the stream generator. Developers ought to use the following procedure to verify that an encoder is producing the intended results:

1) Use the sample stream generator (Section 7) to produce a new VC-2 stream using identical input video and coding parameters to those used to generate the candidate VC-2 stream.

2) Decode the candidate VC-2 stream using the reference decoder (Section 6).

3) Decode the new VC-2 stream using the reference decoder.

4) Compare the two sets of coding parameters (steps 2 and 3). If the two sets differ, there is a fault in the candidate VC-2 stream produced by the encoder under test.

5) Compare the two sets of video files (steps 2 and 3). If the two sets differ, there is a fault in the candidate stream produced by the encoder under test.

Annex A Coding Parameters File (Normative)

The file as a whole shall indicate the metadata values sent in the VC-2 stream. The file is intended to be human readable with clear interpretation. The following rules shall apply:

- Indentation shall use the ASCII space character only. Indentation shall be considered significant.
- Numeric values shall be represented as ISO/IEC 646 numerals.
- String values shall be represented as ISO/IEC 646 printing characters encapsulated in double quotes; e.g., "string literal".
- Boolean values shall be represented using the words "**`True`**" and "**`False`**" as defined by SMPTE ST 2042-1.
- A single file shall correspond to the parameters used to decode/encode either a single video frame or a pair of video fields.
- Each coding parameters file shall contain the following two sections for each picture in the frame:
    1. a sequence parameter section, and
    2. a picture parameter section.

  Note: If the frame contains two pictures, then these two sections will define respectively the parameters of the first picture, followed immediately by a second pair of sequence and picture parameter sections defining the parameters of the second picture.

The individual sections shall be as defined below.

A.1 Sequence Parameter Section

The Sequence parameter section shall start with the text string: "`--- !SequenceParameters`".

This section shall comprise the parameters extracted from the sequence parameters in force at the point of encoding or decoding the frame (see `signalled` parameter in Table A.1).

The parameters in this section shall be as defined below. Unless otherwise specified, all references to section numbers in the tables below shall refer to SMPTE ST 2042-1.

Table A.1 – Sequence Parameters Structure

Key	Meaning
`signaled`	This parameter does not correspond to an individual flag in the stream. When **`True`**, this sequence parameters section was present in the VC-2 stream prior to the following picture. When **`False`**, this sequence parameters section was inferred from a previously signaled section. This parameter shall be **`True`** at the start point of a new sequence.
`version`	In the form "`major aa, minor bb`", representing the version number of the VC-2 stream as per Section 11.1.1.
`profile`	Numeric values as per 11.1 and Annex D.1.
`level`	Numeric values as per 11.1 and Annex D.2.
`picture_coding_mode`	Numeric values as per 11.4
`video_format`	A structure containing the parameters defined in Table A.2.

Table A.2 – Video Format Structure

Key	Meaning
base_video_format	Numeric values as per 11.2.
color difference_format[1]	Numeric values as per 11.3.3.
source_sampling[1]	Boolean value as per 11.3.4.
top_field_first	As defined by the base_video_format.
frame_rate[1]	In the form {numerator: aa, denominator: bb}, represents a custom frame rate as per 11.3.5. In the form nn, numeric values represent frame_rate_index as per 11.3.5.
pixel_aspect_ratio[1]	In the form {numerator: aa, denominator: bb}, represents a custom pixel aspect ratio as per 11.3.6. In the form nn, numeric values represent pixel_aspect_ratio_index as per 11.3.6.
clean_area[1]	In the form {width: aa, height: bb, left_offset: cc, top_offset: dd}, represents a clean area as per 11.3.7.
signal_range[1]	In the form nn, numeric values represent signal_range_index as per 11.3.8. In the form of a structure, represents a custom signal range as per 11.3.8: luma: {offset: aa, excursion: bb} color difference: {offset: aa, excursion: bb}
color_spec[1]	In the form nn, numeric values represents custom_color_spec_index as per 11.3.9. In the form of a structure:

color_primaries[2]	Numeric values as per index in 11.3.9.1. Textual values map to numeric values: "ITURec709"=0, "ITURec601_SD525"=1, "ITURec601_SD625"=2, "SMPTE428p1"=3.
color_matrix[2]	Numeric values as per index in 11.3.9.2. Textual values map to numeric values: "ITURec709"=0, "ITURec601"=1, "YCoCg"=2.
transfer_function[2]	Numeric values as per index in 11.3.9.3. Textual values map to numeric values: "ITURec709"=0, "ITURec1361_1998Annex1"=1, "Linear"=2, "SMPTE428p1"=3.

Notes:

1. If present, it signals a custom value that overrides the default value set by `base_video_format`.

2. If present, it signals a custom value that overrides the default values set by `base_video_format` and `preset_color_spec`.

A.2 Picture Parameter Section

This section shall start with the text string: "`--- !PictureParameters`".

This section shall comprise the parameters extracted from a picture data unit excluding the coefficient values.

The parameters in this section shall be as defined below. Unless otherwise specified, all references to section numbers in the tables below shall refer to SMPTE 2042-1.

Depending upon the picture type, either a `low_delay_syntax` or `core_syntax` set of parameters are specified.

Table A.3 – Picture Structure (common parameters)

Key	Meaning
`picture_number`	Numeric values as per 12.1.
`wavelet_index`	Numeric values as per 12.3.1. Textual values map to numeric values: `"DDubuc97"`=0, `"LeGall53"`=1, `"DDubuc137"`=2, `"Haar0"`=3, `"Haar1"`=4, `"Fidelity"`=5, `"Approx97"`=6.
`dwt_depth`	Numeric values as per 12.3.2.
`low_delay_syntax`	Presence indicates the picture is coded in `low_delay_syntax` mode. Shall not be present if `core_syntax` is present. A structure containing the parameters defined in table A.4 below.
`core_syntax`	Presence indicates the picture is coded in `core_syntax` mode. Shall not be present if `low_delay_syntax` is present. A structure containing the parameters defined in table A.5 below.

Table A.4 – Low Delay Syntax Structure

<table>
<tr><th>Key</th><th>Meaning</th></tr>
<tr><td>slice_bytes</td><td>In the form {numerator: aa, denominator: bb}, represents as per 12.3.4.1.</td></tr>
<tr><td>slices_x</td><td>Numeric values as per 12.3.4.1.</td></tr>
<tr><td>slices_y</td><td>Numeric values as per 12.3.4.1.</td></tr>
<tr><td>quant_matrix</td><td>In the form [aa, bb, cc, ..., zz] is a list of numeric values in subband order forming the quantization matrix as per 12.3.4.2.
If present, it signals a custom value that overrides the default set by dwt_depth and wavelet_index.</td></tr>
<tr><td>slices</td><td>A list of structures indicating the parameters for each slice.
Note: there will typically be many slice structures in a picture.
Each structure contains:
<table>
<tr><td>sx</td><td>position of slice in slice co-ordinates (as per 13.5.1).</td></tr>
<tr><td>sy</td><td>position of slice in slice co-ordinates (as per 13.5.1).</td></tr>
<tr><td>qindex</td><td>Numeric values as per 13.5.2.</td></tr>
<tr><td>slice_y_length</td><td>Numeric values as per 13.5.2.</td></tr>
</table></td></tr>
</table>

Table A.5 – Core Syntax Structure

<table>
<tr><th>Key</th><th>Meaning</th></tr>
<tr><td>using_ac</td><td>Boolean value as per 10.4.1.</td></tr>
<tr><td>codeblock_mode</td><td>Conditional numeric value as per 12.3.3.</td></tr>
<tr><td>levels</td><td>A list of structures, one for each level. Each structure contains:
<table>
<tr><td>level</td><td>Numeric values as per 12.3.3, 13.4.1.</td></tr>
<tr><td>codeblocks_x</td><td>Numeric values as per 12.3.3.</td></tr>
<tr><td>codeblocks_y</td><td>Numeric values as per 12.3.3.</td></tr>
<tr><td>subbands</td><td>A list of structures, one for each subband within the level. Each structure contains:
<table>
<tr><td>component</td><td>String values ("Y", "C1" or "C2") as per 13. Indicates the video component this structure applies to.</td></tr>
<tr><td>orient</td><td>String values ("LL", "HL", "LH" or "HH") as per 13.4.1.</td></tr>
<tr><td>length</td><td>Numeric values as per subband_length in 13.4.2.</td></tr>
<tr><td>quant_index</td><td>Numeric values as per 13.4.2.</td></tr>
<tr><td>codeblocks</td><td>A list of structures, one for each codeblock within the subband.
Only present if codeblocks_x*codeblocks_y > 1.
Each structure contains:
<table>
<tr><td>cx</td><td>Numeric values indicating position of the codeblock in codeblock co-ordinates.</td></tr>
<tr><td>cy</td><td>Numeric values indicating position of the codeblock in codeblock co-ordinates.</td></tr>
<tr><td>quant_index_offset</td><td>Numeric values as per 13.4.3.4. Only present if codeblock_mode == 1 and skipped_codeblock == false.</td></tr>
<tr><td>skipped_codeblock</td><td>Boolean values as per 13.4.3.3.</td></tr>
</table></td></tr>
</table></td></tr>
</table></td></tr>
</table>

A.3 Example File (Informative)

Further examples are provided in the register.

The following example shows a single VC-2 progressive frame that uses parameters from the HD1080P-50 video format, except:

- the frame rate has been overridden to 24fps,
- the signal ranges have been setup for 8 bit video and are to be signaled with `custom_signal_range_index` = 0
- and the `color_spec` uses the defaults as for `custom_color_spec_index` = 0 except the color primaries which are further overridden to be SD625 primaries.

```
--- !SequenceParameters

signalled: true
version:    major: 1, minor: 0
profile: 1
level: 4
picture_coding_mode: 0
video_format:
  base_video_format: 14
  frame_rate: 2
  signal_range:
    luma: {offset: 16, excursion: 219}
    color difference: {offset: 128, excursion: 224}
  color_spec:
    color_primaries: "ITURec601_SD625"
--- !PictureParameters
picture_number: 1234
wavelet_index: 0
dwt_depth: 2
low_delay_syntax:
  slice_bytes: {numerator: 61, denominator: 1}
  slices_x: 2
  slices_y: 2
  quant_matrix: [6, 3, 3, 0, 5, 5, 0]
  slices:
    - {sx: 0, sy: 0, qindex: 5, slice_y_length: 15673}
    - {sx: 1, sy: 0, qindex: 6, slice_y_length: 21423}
    - {sx: 0, sy: 1, qindex: 7, slice_y_length: 87340}
    - {sx: 1, sy: 1, qindex: 3, slice_y_length: 68413}
```

Annex B Image File Format Specifications (Normative)

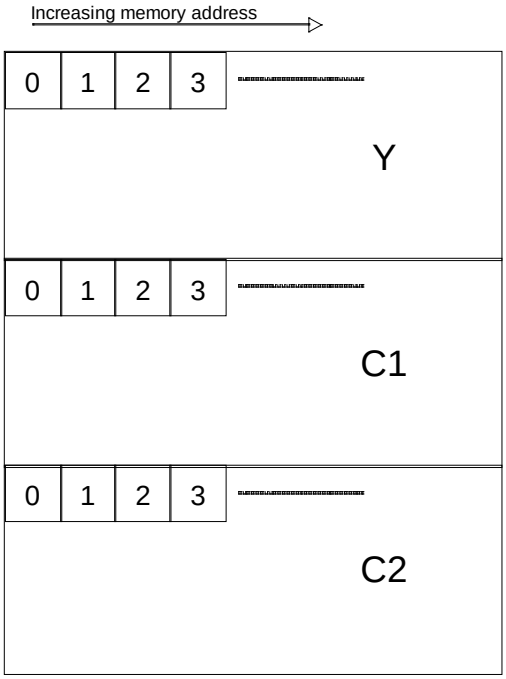

Figure B.1 – Memory layout of the image file format for 16P4

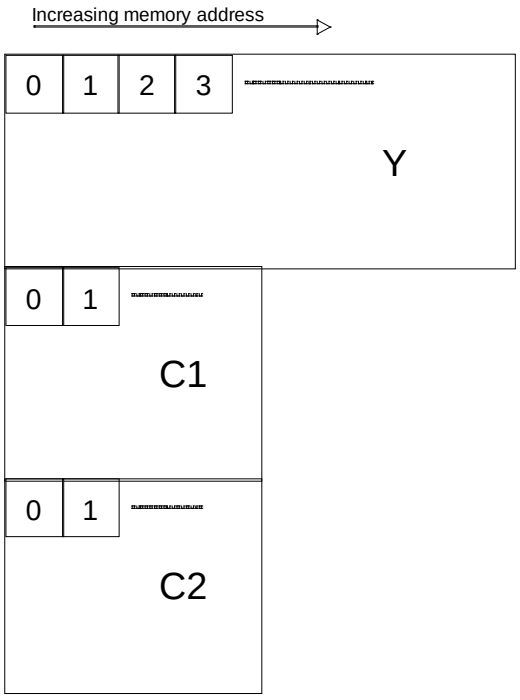

Figure B.2 – Memory layout of the image file format for 16P2

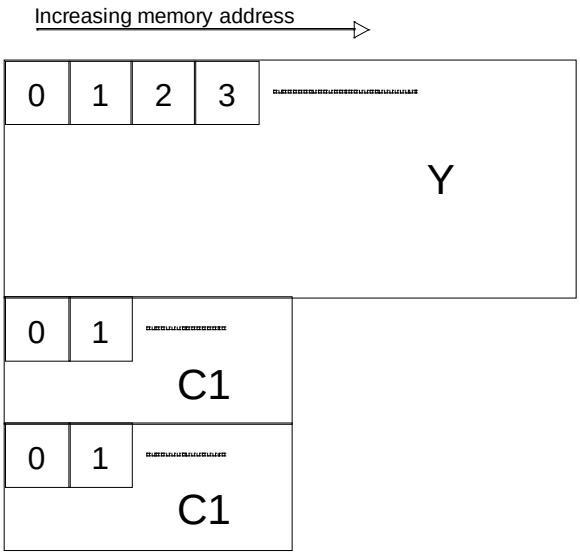

Figure B.3 – Memory layout of the image file format for 16P0

Annex C VC-2 Test Streams (Informative)

C.1 Overall Description of Test Streams

The test directory contains a large number of test streams. These streams are intended for testing and verifying SMPTE 2042-1 (VC-2) decoders. The test streams are contained in a hierarchy of directories that are designed to simplify finding the streams necessary to test a specific decoder implementation. It is important to note that since these test sequences are intended for decoder test purposes it is inappropriate to use them to assess picture quality. The picture coding decisions are designed to provide good test coverage and are not the coding decisions that would be used in producing high quality compressed pictures.

The test directory contains subdirectories called “real-pictures” and “syntax+semantics” which contain the conformance test streams. It also contains a directory called “software”, which contains source code and executable code for a reference decoder, called “vc2decoder”, and for a program, called “vc2streamgen” to encode SMPTE ST 2042-1 (VC-2) streams. These programs are described in more detail in sections 6 and 7 of this specification.

The test directory also contains an, optional, directory called “decoded”, which contains some decoded test streams and their associated decoded metadata. Clearly the “decoded” subdirectory is much larger than the actual test streams because it contains uncompressed video sequences. For this reason not all the test sequences are provided in decoded form. To do would require an unfeasibly large amount of extra storage (111 Gbytes). The decode files are “optional” because they can be regenerated by decoding the test streams using the supplied reference decoder software.

A decoder can be tested by comparing the decoded frames produced by the decoder under test with reference decoded frames. The frames produced by a conformant decoder will be bit identical to the reference decoded frames.

The set of test streams goes further than simply ensuring that video is correctly decoded. Some tests intentionally contravene restrictions defined in the specification. For example a test might try to access an undefined member of a table. Such tests can cause a decoder to fail. These tests allow the behaviour of a decoder to be assessed when presented with a non-conformant stream. It would be appropriate for a decoder to indicate, at least for test purposes, that it has encountered a non-conformant stream. Other streams are designed to test correct decoding of the metadata contained within the stream. Correct decoding of this metadata can be determined by comparing it to the correctly decoded metadata that accompanies the decoded stream. The decoded metadata is provided as human readable test files with a “.cparams” extension. Like the decoded pictures a separate “.cparams” file is provided for each frame of the decoded stream. The details of the formats of decoded frames and associated metadata are as defined in the two preceding Annexes A and B.

The conformance test streams are divided between the “real-pictures” and “syntax+semantics” directories. The former directory contains test streams using coded real pictures. The latter directory contains streams that test additional specific aspects of the stream syntax and semantics. Each of these sets of test streams are described in more detail below.

The real picture streams test a wide range of parameters. Multiple tests are included in many of the streams in order to minimize the total number of streams. A particular stream, for a given profile, level and video format, might contain concatenated sequences that test multiple wavelet depths and wavelet filter kernels. For example the stream called “rp-bvfi7-cs-lvl2-d13-noac-nocb-pcmi-cr422-10b.drc” (stream naming is described below) contains 10 concatenated sequences. The first 5 frames form the first sequence in the stream and are coded with a wavelet depth of 1 using wavelet kernel index zero (i.e. the Deslauriers-Dubuc (9,7) kernel). The next five frames comprise a second sequence and are coded with a wavelet depth of 3 using the same wavelet kernel. Eight further, 5 frame, sequences are concatenated, which test the other permutations of wavelet depths 1 and 3 and wavelet kernel indices 0 to 4. Concatenating multiple sequences not only reduces the number of streams but also tests the decoder’s response to starting and ending sequences within a stream. The description of Real Picture Test Streams is provided in Annex C.2.

The syntax and semantics test streams are intended to verify all the remaining aspects of the SMPTE 2042-1 specification that are not directly tested in the real picture streams. They ensure, for example, that all the base video formats, all color difference resolutions and 8, 10 and 12 bit video bit depths are tested. There are thousands of these test streams, but each stream is small and only a subset of these streams may be required to test a specific coder implementation. The description of the syntax and semantics test streams is provided in Annex C.3.

The VC-2 test streams and other VC-2 resources can be found at http://store.smpte.org.

C.2 Description of Real Picture Test Streams

There are 62 real picture test streams (using pictured with a camera), many of which contain 10 individual, concatenated, sequences. The streams are either 25 or 50 frames long. They cover all three profiles, a selection of levels, a range of base video formats, wavelet depths from 1 to 4 and a range of wavelet filter kernels. These streams test both progressive and interlaced video formats. Any other parameters, not covered in these streams, are tested in the syntax and semantic test streams described below.

The stream names are intentionally long to encapsulate information about the stream. The names comprise a number of fields concatenated by hyphens. Some of the fields are followed (or preceded) by one or more numerical values. The meaning of these fields is indicated in the table below.

Name Field	Description	Values	Comment
rp	real pictures		
bvfi	base video format index	0 to 20	
cs	core syntax		simple and main profiles
ld	low delay		low delay profile
lv	level	1 to 7 & 64	multiple levels possible
d	wavelet depth	1 to 4	multiple depths possible
ac	arithmetic coding		core syntax only
noac	no arithmetic coding		core syntax only
cb	using code blocks		
nocb	not using code blocks		
cbmq	code blocks with multiple quantization indices		
pcm	picture coding mode	i or p	
cr	color difference resolution	420, 422, 444	
b	bit depth	8, 10, 12	value precedes “b”
znpo	zero next parse offset		for live streaming
cqm	custom quantisation matrix		

For example the test stream called “rp-bvfi7-cs-lvl2-d13-noac-nocb-pcmi-cr422-10b.drc” is a real picture test stream, with a base video format index of 7 (i.e. 480 line standard definition video at 60/1.001 fields/second), coded with core syntax and no arithmetic coding (i.e. simple profile), coded without using code blocks and coded using interlaced coding mode for 4:2:2, 10 bit video.

The table below indicates the main parameters tested by the real picture streams. The wavelet filter depth and the wavelet filter (kernel) may have multiple entries. If so, it indicates that the stream contains multiple concatenated streams.

Sequence Name	Profile	Level	Base Video Format	Coding Mode	Code blocks	Wavelet Depth	Wavelet Filter
rp-bvfi7-ld-lvl2-d234-znpo-cqm-pcmi-cr422-10b.drc	low delay	2	7	interlaced	n/a	2,3,4	0,1,2,3,4
rp-bvfi7-cs-lvl2-d13-noac-nocb-pcmi-cr422-10b.drc	simple	2	7	interlaced	none	1,3	0,1,2,3,4
rp-bvfi7-cs-lvl2-d2-noac-cb-pcmi-cr422-10b.drc	simple	2	7	interlaced	codeblocks	2	0,1,2,3,4
rp-bvfi7-cs-lvl2-d4-noac-cbmq-pcmi-cr422-10b.drc	simple	2	7	interlaced	multiquant	4	0,1,2,3,4
rp-bvfi7-cs-lvl2-d24-ac-nocb-pcmi-cr422-10b.drc	main	2	7	interlaced	none	2,4	0,1,2,3,4
rp-bvfi7-cs-lvl2-d1-ac-cb-pcmi-cr422-10b.drc	main	2	7	interlaced	codeblocks	1	0,1,2,3,4
rp-bvfi7-cs-lvl2-d3-ac-cbmq-pcmi-cr422-10b.drc	main	2	7	interlaced	multiquant	3	0,1,2,3,4
rp-bvfi8-ld-lvl2-d234-znpo-cqm-pcmi-cr422-10b.drc	low delay	2	8	interlaced	n/a	2,3,4	0,1,2,3,4
rp-bvfi8-cs-lvl2-d13-noac-nocb-pcmi-cr422-10b.drc	simple	2	8	interlaced	none	1,3	0,1,2,3,4
rp-bvfi8-cs-lvl2-d2-noac-cb-pcmi-cr422-10b.drc	simple	2	8	interlaced	codeblocks	2	0,1,2,3,4
rp-bvfi8-cs-lvl2-d4-noac-cbmq-pcmi-cr422-10b.drc	simple	2	8	interlaced	multiquant	4	0,1,2,3,4
rp-bvfi8-cs-lvl2-d24-ac-nocb-pcmi-cr422-10b.drc	main	2	8	interlaced	none	2,4	0,1,2,3,4
rp-bvfi8-cs-lvl2-d1-ac-cb-pcmi-cr422-10b.drc	main	2	8	interlaced	codeblocks	1	0,1,2,3,4
rp-bvfi8-cs-lvl2-d3-ac-cbmq-pcmi-cr422-10b.drc	main	2	8	interlaced	multiquant	3	0,1,2,3,4

rp-bvfi9-ld-lvl3-d3-pcmp-cr422-10b.drc	low delay	3	9	progressive	n/a	3	1
rp-bvfi9-ld-lvl3-d234-znpo-cqm-pcmp-cr422-10b.drc	low delay	3	9	progressive	n/a	2,3,4	0,1,2,3,4
rp-bvfi9-cs-lvl3-d13-noac-nocb-pcmp-cr422-10b.drc	simple	3	9	progressive	none	1,3	0,1,2,3,4
rp-bvfi9-cs-lvl3-d2-noac-cb-pcmp-cr422-10b.drc	simple	3	9	progressive	codeblocks	2	0,1,2,3,4
rp-bvfi9-cs-lvl3-d4-noac-cbmq-pcmp-cr422-10b.drc	simple	3	9	progressive	multiquant	4	0,1,2,3,4
rp-bvfi9-cs-lvl3-d24-ac-nocb-pcmp-cr422-10b.drc	main	3	9	progressive	none	2,4	0,1,2,3,4
rp-bvfi9-cs-lvl3-d1-ac-cb-pcmp-cr422-10b.drc	main	3	9	progressive	codeblocks	1	0,1,2,3,4
rp-bvfi9-cs-lvl3-d3-ac-cbmq-pcmp-cr422-10b.drc	main	3	9	progressive	multiquant	3	0,1,2,3,4
rp-bvfi10-ld-lvl3-d3-pcmp-cr422-10b.drc	low delay	3	10	progressive	n/a	3	1
rp-bvfi10-ld-lvl3-d234-znpo-cqm-pcmp-cr422-10b.drc	low delay	3	10	progressive	n/a	2,3,4	0,1,2,3,4
rp-bvfi10-cs-lvl3-d13-noac-nocb-pcmp-cr422-10b.drc	simple	3	10	progressive	none	1,3	0,1,2,3,4
rp-bvfi10-cs-lvl3-d2-noac-cb-pcmp-cr422-10b.drc	simple	3	10	progressive	codeblocks	2	0,1,2,3,4
rp-bvfi10-cs-lvl3-d4-noac-cbmq-pcmp-cr422-10b.drc	simple	3	10	progressive	multiquant	4	0,1,2,3,4
rp-bvfi10-cs-lvl3-d24-ac-nocb-pcmp-cr422-10b.drc	main	3	10	progressive	none	2,4	0,1,2,3,4
rp-bvfi10-cs-lvl3-d1-ac-cb-pcmp-cr422-10b.drc	main	3	10	progressive	codeblocks	1	0,1,2,3,4
rp-bvfi10-cs-lvl3-d3-ac-cbmq-pcmp-cr422-10b.drc	main	3	10	progressive	multiquant	3	0,1,2,3,4
rp-bvfi11-ld-lvl3-d3-pcmi-cr422-10b.drc	low delay	3	11	interlaced	n/a	3	0,1,2,3,4
rp-bvfi11-ld-lvl3-d234-znpo-cqm-pcmi-cr422-10b.drc	low delay	3	11	interlaced	n/a	2,3,4	1
rp-bvfi11-cs-lvl3-d13-noac-nocb-pcmi-cr422-10b.drc	simple	3	11	interlaced	none	1,3	0,1,2,3,4

rp-bvfi11-cs-lvl3-d2-noac-cb-pcmi-cr422-10b.drc	simple	3	11	interlaced	codeblocks	2	0,1,2,3,4
rp-bvfi11-cs-lvl3-d4-noac-cbmq-pcmi-cr422-10b.drc	simple	3	11	interlaced	multiquant	4	0,1,2,3,4
rp-bvfi11-cs-lvl3-d24-ac-nocb-pcmi-cr422-10b.drc	main	3	11	interlaced	none	2,4	0,1,2,3,4
rp-bvfi11-cs-lvl3-d1-ac-cb-pcmi-cr422-10b.drc	main	3	11	interlaced	codeblocks	1	0,1,2,3,4
rp-bvfi11-cs-lvl3-d3-ac-cbmq-pcmi-cr422-10b.drc	main	3	11	interlaced	multiquant	3	0,1,2,3,4
rp-bvfi12-ld-lvl3-d234-znpo-cqm-pcmi-cr422-10b.drc	low delay	3	12	interlaced	n/a	2,3,4	0,1,2,3,4
rp-bvfi12-ld-lvl3-d3-pcmi-cr422-10b.drc	low delay	3	12	interlaced	n/a	3	1
rp-bvfi12-cs-lvl3-d13-noac-nocb-pcmi-cr422-10b.drc	simple	3	12	interlaced	none	1,3	0,1,2,3,4
rp-bvfi12-cs-lvl3-d2-noac-cb-pcmi-cr422-10b.drc	simple	3	12	interlaced	codeblocks	2	0,1,2,3,4
rp-bvfi12-cs-lvl3-d4-noac-cbmq-pcmi-cr422-10b.drc	simple	3	12	interlaced	multiquant	4	0,1,2,3,4
rp-bvfi12-cs-lvl3-d24-ac-nocb-pcmi-cr422-10b.drc	main	3	12	interlaced	none	2,4	0,1,2,3,4
rp-bvfi12-cs-lvl3-d1-ac-cb-pcmi-cr422-10b.drc	main	3	12	interlaced	codeblocks	1	0,1,2,3,4
rp-bvfi12-cs-lvl3-d3-ac-cbmq-pcmi-cr422-10b.drc	main	3	12	interlaced	multiquant	3	0,1,2,3,4
rp-bvfi13-ld-lvl3-d234-znpo-cqm-pcmp-cr422-10b.drc	low delay	3	13	progressive	n/a	2,3,4	0,1,2,3,4
rp-bvfi13-ld-lvl64-d2-pcmp-cr422-10b.drc	low delay	64	13	progressive	n/a	2	4
rp-bvfi13-cs-lvl3-d13-noac-nocb-pcmp-cr422-10b.drc	simple	3	13	progressive	none	1,3	0,1,2,3,4
rp-bvfi13-cs-lvl3-d2-noac-cb-pcmp-cr422-10b.drc	simple	3	13	progressive	codeblocks	2	0,1,2,3,4
rp-bvfi13-cs-lvl3-d4-noac-cbmq-pcmp-cr422-10b.drc	simple	3	13	progressive	multiquant	4	0,1,2,3,4
rp-bvfi13-cs-lvl3-d24-ac-nocb-pcmp-cr422-10b.drc	main	3	13	progressive	none	2,4	0,1,2,3,4

rp-bvfi13-cs-lvl3-d1-ac-cb-pcmp-cr422-10b.drc	main	3	13	progressive	codeblocks	1	0,1,2,3,4
rp-bvfi13-cs-lvl3-d3-ac-cbmq-pcmp-cr422-10b.drc	main	3	13	progressive	multiquant	3	0,1,2,3,4
rp-bvfi14-ld-lvl3-d234-znpo-cqm-pcmp-cr422-10b.drc	low delay	3	14	progressive	n/a	2,3,4	0,1,2,3,4
rp-bvfi14-ld-lvl64-d2-pcmp-cr422-10b.drc	low delay	64	14	progressive	n/a	2	4
rp-bvfi14-cs-lvl3-d13-noac-nocb-pcmp-cr422-10b.drc	simple	3	14	progressive	none	1,3	0,1,2,3,4
rp-bvfi14-cs-lvl3-d2-noac-cb-pcmp-cr422-10b.drc	simple	3	14	progressive	codeblocks	2	0,1,2,3,4
rp-bvfi14-cs-lvl3-d4-noac-cbmq-pcmp-cr422-10b.drc	simple	3	14	progressive	multiquant	4	0,1,2,3,4
rp-bvfi14-cs-lvl3-d24-ac-nocb-pcmp-cr422-10b.drc	main	3	14	progressive	none	2,4	0,1,2,3,4
rp-bvfi14-cs-lvl3-d1-ac-cb-pcmp-cr422-10b.drc	main	3	14	progressive	codeblocks	1	0,1,2,3,4
rp-bvfi14-cs-lvl3-d3-ac-cbmq-pcmp-cr422-10b.drc	main	3	14	progressive	multiquant	3	0,1,2,3,4

C.3 Description of Syntax and Semantic Test Streams

These test streams test the stream syntax and semantics and are contained in the “syntax and semantics” directory. There are approximately one hundred basic tests and these tests are repeated for each of the three profiles and each of the 21 base video formats. The result is a suite of approximately six thousand test sequences, although not all of the basic tests apply to every profile and video format.

The structure of the “syntax+semantics” directory that contains these streams is structured into three subdirectories “low-delay”, “simple” and “main”, corresponding to the 3 profiles. Each of these subdirectories is further structured into 21 subdirectories, with names such as “HD1080I-60”, corresponding to the 21 base video formats. These directory names correspond to the informative names for the base video formats defined in Annex C of SMPTE ST 2042-1. The test sequences within each sub directory contain the test number as a field in the name (see also below).

The set of basic tests are numbered from 1 to 114. These tests are subdivided into a number of sets as shown in the table below.

Test Numbers	Description
0 to 15	Sequence Header syntax
16 to 60	Sequence Header semantics
61 to 73	Sequence semantics
74 to 84	Picture semantics
85 to 114	Picture syntax

The syntactic tests are designed to exercise the branches in flow of decoding specified in SMPTE ST 2042-1. The semantic tests are designed to test restrictions embodied in the specification. Note that not all test are applicable for each profile and base video format, so “missing” tests do not indicate that the conformance test suite is incomplete.

Some sequences are intentionally designed as non-conformant streams. Such streams are intended to test the behaviour of a decoder in the presence of a stream which is non-conformant in a specific defined way. Non-conformant test streams are flagged by the presence of the field “fail” in the stream name. That is, decoding is expected to fail in some manner for such streams. All other streams should decode correctly and may be flagged by the presence of the field “pass” in the stream name.

Most of these tests decode to yield a single mid-grey frame (i.e., all zero pictures). Such pictures are referred to as “minimal pictures” because they represent the default picture when minimum information is encoded in the sequence. Rather than provide many identical versions of these minimal decoded pictures a single example minimal picture, for each base video format, is provided in the directory “/decoded/minimal-pictures”. To ensure that the decoder is compliant the decoded metadata can be compared with that from the decoder, in addition to confirming that the decoded picture is identical to the appropriate minimal picture.

Note: For core syntax (simple and main profiles), a minimal picture is one in which all sub-bands are “skipped”. For the low delay profile, a minimal picture is where the whole picture is a single slice and 2 bytes are allocated to code the slice. The slice contains a 7-bit quantisation index (arbitrarily set to 42d, 0101010b), 4 bits to indicate the UV partition and the remaining bits are set to zero.

All syntactic and semantic test streams decode to yield a minimal picture, EXCEPT the picture syntactic tests (test numbers 85 to 114) which use random data. Random data is used because minimal pictures would not provide an adequate test for these streams. The decoded “random” pictures are provided in the appropriate subdirectory under the “decoded” directory.

The stream names are intentionally long to encapsulate information about the stream. The names comprise a number of fields concatenated by hyphens. Some of the fields are followed (or preceded) by one or more numerical values. The meaning of these fields is indicated in the table below. Note in particular that each stream name includes the test number.

Name Field	Description	Values	Comment
ss	Syntax and semantics		
	Test Number	1 to 114	Corresponds to test numbers in tables below
bvfi	base video format index	0 to 20	
cs	lore syntax		simple and main profiles
ld	low delay		low delay profile
lv	level	1 to 7 & 64	multiple levels possible
d	wavelet depth	1 to 4	multiple depths possible
ac	arithmetic coding		core syntax only
noac	no arithmetic coding		core syntax only

A description of the syntactic and semantic test sequences is provided in the tables below.

Sequence Header syntax		
Test Number	**Description**	**Pass/Fail**
0	`base sequence header`	pass
1	`custom_dimensions_flag (with default dimensions)`	pass
2	`custom_color difference_format_flag (with default color difference format)`	pass
3	`custom_scan_format_flag (with default scan format)`	pass
4	`custom_frame_rate_index=0 (with default frame rate)`	pass
5	`custom_frame_rate_index = default frame rate index`	pass
6	`custom_pixel_aspect_ratio_index=0 (with default pixel aspect ratio)`	pass
7	`custom_pixel_aspect_ratio_index = default pixel aspect ratio index`	pass
8	`custom_clean_area_flag (with default clean area)`	pass
9	`custom_signal_range_index=0 (with default signal range)`	pass
10	`custom_signal_range_index = default signal range index`	pass

11	custom_color_spec_index = default color spec	pass
12	custom_color_spec_flag (with default index)	pass
13	custom_color_primaries_flag (with default color primaries)	pass
14	custom_color_matrix_flag (with default color matrix)	pass
15	custom_transfer_function_flag (with default transfer function)	pass

Sequence Header Semantics		
Test Number	**Description**	**Pass/Fail**
16	legal future version_minor	pass
17	invalid version_major	fail
18	invalid profile	fail
19	invalid level	fail
20	first invalid base_video_format_index	fail
21	last valid base_video_format_index	pass
22	first invalid color difference_format	fail
23	last valid color difference_format	pass
24	first invalid source_sampling	fail
25	last valid source_sampling	pass
26	first invalid custom_frame_rate_index	fail
27	last valid custom_frame_rate_index	pass
28	first invalid custom_pixel_aspect_ratio_index	fail
29	last valid custom_pixel_aspect_ratio_index	pass
30	first invalid custom_signal_range_index	fail
31	last valid custom_signal_range_index	pass
32	first invalid custom_color_spec_index	fail
33	last valid custom_color_spec_index	pass
34	first invalid custom_color_primaries_index	fail

35	last valid custom_color_primaries_index	pass
36	first invalid custom_color_matrix_index	fail
37	last valid custom_color_matrix_index	pass
38	first invalid custom_transfer_function_index	fail
39	last valid custom_transfer_function_index	pass
40	first invalid picture_coding_mode	fail
41	picture_coding_mode = opposite of source sampling	pass
42	invalid: clean area exceeds the bounds of frame_width	fail
43	valid: clean area of same dimensions as frame	pass
44	invalid: clean area exceeds the bounds of frame_height	fail
45	custom frame_height mod 4 = 0 (picture_coding_mode = frame coding) (4:4:4)	pass
46	custom frame_height mod 4 = 3 (picture_coding_mode = frame coding) (4:4:4)	pass
47	custom frame_height mod 4 = 0 (picture_coding_mode = field coding) (4:4:4)	pass
48	custom frame_height mod 4 = 1 (picture_coding_mode = field coding) (4:4:4)	fail
49	custom frame_height mod 4 = 0 (picture_coding_mode = frame) (4:2:2)	pass
50	custom frame_height mod 4 = 1 (picture_coding_mode = frame) (4:2:2)	pass
51	custom frame_height mod 4 = 2 (picture_coding_mode = field) (4:2:2)	pass
52	custom frame_height mod 4 = 3 (picture_coding_mode = field) (4:2:2)	fail
53	custom frame_height mod 4 = 0 (picture_coding_mode = frame) (4:2:0)	pass
54	custom frame_height mod 4 = 1 (picture_coding_mode = frame) (4:2:0)	fail
55	custom frame_height mod 4 = 2 (picture_coding_mode = frame) (4:2:0)	pass
56	custom frame_height mod 4 = 3 (picture_coding_mode = frame) (4:2:0)	fail

57	custom frame_height mod 4 = 0 (picture_coding_mode = field) (4:2:0)	pass
58	custom frame_height mod 4 = 1 (picture_coding_mode = field) (4:2:0)	fail
59	custom frame_height mod 4 = 2 (picture_coding_mode = field) (4:2:0)	fail
60	custom frame_height mod 4 = 3 (picture_coding_mode = field) (4:2:0)	fail

Sequence semantics		
Test Number	**Description**	**Pass/Fail**
61	First frame is odd numbered	pass
62	Sequence contains no pictures	pass
63	Sequence starts with dangling field	fail
64	Sequence ends with dangling field	fail
65	Sequence header not constant within sequence	fail
66	Zero Next Parse Offset on all data units	pass
67	No End of Sequence data unit	fail
68	Duplicate Picture Number	fail
69	Picture Number discontinuity within sequence	fail
70	Picture Number wraps within sequence	pass
71	MultiSeq with Picture Number discontinuity at sequence boundary	pass
72	Profile mismatch	fail
73	Level mismatch (too low)	fail

Picture semantics		
Test Number	**Description**	**Pass/Fail**
74	first invalid wavelet_index	fail
75	last valid wavelet_index	pass
76	first invalid codeblock_mode	fail
77	slice_bytes too small for slice	fail
78	division by zero in slice_bytes	fail
79	custom_quant_matrix=false with too high dwt_depth	fail
80	too many slices vertically (some slices with missing subband)	fail
81	too many slices horizontally (some slices with missing subband)	fail
82	maximum number of slices	pass
83	maximum number of codeblocks	pass
84	slice_length_exceeds avaliable slice storage	fail

Picture syntax		
Test Number	**Description**	**Pass/Fail**
85	spatial_partitioning=false	pass
86	spatial_partitioning=codeblocks, num_blocks=1	pass
87	spatial_partitioning=codeblocks, num_blocks>1	pass
88	spatial_partitioning=multiquant, num_blocks=1	pass
89	spatial_partitioning=multiquant, num_blocks>1, skipping	pass
90	Custom quant matrix = false, quantizer index=0	pass
91	spatial_partitioning=false	pass
92	spatial_partitioning=codeblocks, num_blocks=1	pass

93	spatial_partitioning=codeblocks, num_blocks>1, skipping	pass
94	spatial_partitioning=multiquant, num_blocks=1	pass
95	spatial_partitioning=multiquant, num_blocks>1, skipping	pass
96	custom_quant_matrix=false, quant=0	pass
97	spatial_partitioning=false	pass
98	spatial_partitioning=codeblocks, num_blocks=1	pass
99	spatial_partitioning=codeblocks, num_blocks>1, skipping	pass
100	spatial_partitioning=multiquant, num_blocks=1	pass
101	spatial_partitioning=multiquant, num_blocks>1, skipping	pass
102	custom_quant_matrix=false, quant=0	pass
103	spatial_partitioning=false	pass
104	spatial_partitioning=codeblocks, num_blocks=1	pass
105	spatial_partitioning=codeblocks, num_blocks>1, skipping	pass
106	spatial_partitioning=multiquant, num_blocks=1	pass
107	spatial_partitioning=multiquant, num_blocks>1, skipping	pass
108	custom_quant_matrix=false, quant=0	pass
109	spatial_partitioning=false	pass
110	spatial_partitioning=codeblocks, num_blocks=1	pass
111	spatial_partitioning=codeblocks, num_blocks>1, skipping	pass
112	spatial_partitioning=multiquant, num_blocks=1	pass
113	spatial_partitioning=multiquant, num_blocks>1, skipping	pass
114	custom_quant_matrix=false, quant=0	pass

Annex D Bibliography (Informative)

YAML, see http://www.yaml.org/ for the home page and http://yaml.org/spec/current.pdf for the specification

SMPTE ST 2042-2:2009, VC-2 Level Definitions