INTERNATIONAL STANDARD

ISO/IEC 23008-8

First edition
2015-04-15

Information technology — High efficiency coding and media delivery in heterogeneous environments —

Part 8: Conformance Specification for HEVC

Technologies de l'information — Codage à haute efficacité et livraison des médias dans des environnements hétérogènes —

Partie 8: Spécification de conformité du codage video à haute efficacité

Reference number
ISO/IEC 23008-8:2015(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23008-8 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with ITU-T. The identical text is published as ITU-T H.265 (08/2014).

HEVC Conformance Testing

1 Scope

This Recommendation | International Standard[1] specifies a set of tests and procedures designed to indicate whether encoders or decoders meet the normative requirements specified in Rec. ITU-T H.265 | ISO/IEC 23008-2.

2 Normative references

2.1 General

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.2 Identical Recommendations | International Standards

– None.

2.3 Paired Recommendations | International Standards equivalent in technical content

– Recommendation ITU-T H.265 (in force), *High efficiency video coding.*

– ISO/IEC 23008-2: in force, *Information technology – High efficiency video coding and media delivery in heterogeneous environment – Part 2: High Efficiency Video Coding.*

– Recommendation ITU-T H.265.2 (in force), *High efficiency coding reference software*.

– ISO/IEC 23008-5: in force, *Information technology – High efficiency video coding and media delivery in heterogeneous environment – Part 2: High Efficiency Video Coding Reference Software*.

2.4 Additional references

– None.

3 Definitions

For the purposes of this Recommendation | International Standard, the terms, definitions, abbreviations and symbols specified in Rec. ITU-T H.265 | ISO/IEC 23008-2 (particularly in clauses 3) apply. The following terms are further clarified for purposes herein as follows.

3.1 bitstream: A Rec. ITU-T H.265 | ISO/IEC 23008-2 video bitstream.

3.2 decoder: A Rec. ITU-T H.265 | ISO/IEC 23008-2 video decoder, i.e., an embodiment of the decoding process specified by Rec. ITU-T H.265 | ISO/IEC 23008-2. The decoder does not include the display process, which is outside the scope of this Recommendation | International Standard.

[1] This Recommendation | International Standard includes an electronic attachment containing the conformance bitstreams identified within the text.

3.3 encoder: An embodiment of a process, not specified in this Recommendation | International Standard (except in regard to identification of the reference software encoder), that produces a bitstream.

3.4 reference software decoder: The software decoder provided in Rec. ITU-T H.265.2 | ISO/IEC 23008-5.

3.5 reference software encoder: The software encoder provided in Rec. ITU-T H.265.2 | ISO/IEC 23008-5.

4 Abbreviations and acronyms

For the purposes of this Recommendation | International Standard, relevant abbreviations and acronyms are specified in clause 4 of Rec. ITU-T H.265 | ISO/IEC 23008-2.

5 Conventions

For the purposes of this Recommendation | International Standard, relevant conventions are specified in clause 5 of Rec. ITU-T H.265 | ISO/IEC 23008-2.

6 Conformance testing for ITU-T H.265 | ISO/IEC 23008-2

6.1 Introduction

The following clauses specify normative tests for verifying conformance of video bitstreams as well as decoders. Those normative tests make use of test data (bitstream test suites) provided as an electronic annex to this Recommendation | International Standard and the reference software decoder specified in Rec. ITU-T H.265.2 | ISO/IEC 23008-5.

6.2 Bitstream conformance

Bitstream conformance for Rec. ITU-T H.265 | ISO/IEC 23008-2 is specified by clause C.4 of Rec. ITU-T H.265 | ISO/IEC 23008-2.

6.3 Decoder conformance

Decoder conformance for Rec. ITU-T H.265 | ISO/IEC 23008-2 is specified by clause C.5 of Rec. ITU-T H.265 | ISO/IEC 23008-2.

6.4 Procedure to test bitstreams

A bitstream that claims conformance with Rec. ITU-T H.265 | ISO/IEC 23008-2 shall pass the following normative test.

The bitstream shall be decoded by processing it with the reference software decoder. When processed by the reference software decoder, the bitstream shall not cause any error or non-conformance messages to be reported by the reference software decoder. This test should not be applied to bitstreams that are known to contain errors introduced by transmission, as such errors are highly likely to result in bitstreams that lack conformance to Rec. ITU-T H.265 | ISO/IEC 23008-2.

Successfully passing the reference software decoder test provides only a strong presumption that the bitstream under test is conforming to the video layer, i.e., that it does indeed meet all the requirements for the video layer (except Annexes C, D and E) specified in Rec. ITU-T H.265 | ISO/IEC 23008-2 that are tested by the reference software decoder.

Additional tests may be necessary to more thoroughly check that the bitstream properly meets all the requirements specified in Rec. ITU-T H.265 | ISO/IEC 23008-2 including the hypothetical reference decoder (HRD) conformance (based on Annexes C, D and E). These complementary tests may be performed using other video bitstream verifiers that perform more complete tests than those implemented by the reference software decoder.

Rec. ITU-T H.265 | ISO/IEC 23008-2 contains several informative recommendations that are not an integral part of that Recommendation | International Standard. When testing a bitstream for conformance, it may also be useful to test whether or not the bitstream follows those recommendations.

To check correctness of a bitstream, it is necessary to parse the entire bitstream and to extract all the syntax elements and other values derived from those syntactic elements and used by the decoding process specified in Rec. ITU-T H.265 | ISO/IEC 23008-2.

A verifier may not necessarily perform all stages of the decoding process specified in Rec. ITU-T H.265 | ISO/IEC 23008-2 in order to verify bitstream correctness. Many tests can be performed on syntax elements in a state prior to their use in some processing stages.

6.5 Procedure to test decoder conformance

6.5.1 Conformance bitstreams

A bitstream has values of general_profile_idc, general_tier_flag, and general_level_idc corresponding to a set of specified constraints on a bitstream for which a decoder conforming to a specified profile, tier, and level is required in Annex A of Rec. ITU-T H.265 | ISO/IEC 23008-2 to properly perform the decoding process.

6.5.2 Contents of the bitstream file

The conformance bitstreams are included in this Recommendation | International Standard as an electronic attachment. The following information is included in a single zipped file for each such bitstream.

- bitstream;
- decoded pictures or hashes of decoded pictures (may not be present);
- short description of the bitstream;
- trace file (results while decoding the bitstream, in ASCII format).

In cases where the decoded pictures or hashes of decoded pictures are not available, the reference software decoder shall be used to generate the necessary reference decoded pictures from the bitstream.

6.5.3 Requirements on output of the decoding process and timing

Two classes of decoder conformance are specified:

- output order conformance; and
- output timing conformance.

The output of the decoding process is specified in clause 8 and Annex C of Rec. ITU-T H.265 | ISO/IEC 23008-2.

For output order conformance, it is a requirement that all of the decoded pictures specified for output in Annex C of Rec. ITU-T H.265 | ISO/IEC 23008-2 shall be output by a conforming decoder in the specified order and that the values of the decoded samples in all of the pictures that are output shall be (exactly equal to) the values specified in clause 8 of Rec. ITU-T H.265 | ISO/IEC 23008-2.

For output timing conformance, it is a requirement that a conforming decoder shall also output the decoded samples at the rates and times specified in Annex C of Rec. ITU-T H.265 | ISO/IEC 23008-2.

The display process, which ordinarily follows the output of the decoding process, is outside the scope of this Recommendation | International Standard.

6.5.4 Recommendations (informative)

This clause does not form an integral part of this Recommendation | International Standard.

In addition to the requirements, it is desirable that conforming decoders implement various informative recommendations specified in Rec. ITU-T H.265 | ISO/IEC 23008-2 that are not an integral part of that Recommendation | International Standard. This clause discusses some of these recommendations.

It is recommended that a conforming decoder be able to resume the decoding process as soon as possible after the loss or corruption of part of a bitstream. In most cases it is possible to resume decoding at the next start code or slice header. It is recommended that a conforming decoder be able to perform concealment for the coding tree blocks or video packets for which all the coded data has not been received.

6.5.5 Static tests for output order conformance

Static tests of a video decoder require testing of the decoded samples. This clause will explain how this test can be accomplished when the decoded samples at the output of the decoding process are available. It may not be possible to perform this type of test with a production decoder (due to the lack of an appropriate accessible interface in the design at which to perform the test). In that case this test should be performed by the manufacturer during the design and development phase. Static tests are used for testing the decoding process. The test will check that the values of the samples decoded by the decoder under test shall be identical to the values of the samples decoded by the reference decoder. When a hash of the values of the samples of the decoded pictures is attached to the bitstream file, a corresponding hash operation performed on the values of the samples of the decoded pictures produced by the decoder under test shall produce the same results.

6.5.6 Dynamic tests for output timing conformance

Dynamic tests are applied to check that all the decoded samples are output and that the timing of the output of the decoder's decoded samples conforms to the specification of clause 8 and Annex C of Rec. ITU-T H.265 | ISO/IEC 23008-2, and to verify that the HRD models (as specified by the CPB and DPB specification in Annex C of Rec. ITU-T H.265 | ISO/IEC 23008-2) are not violated when the bits of the bitstream are delivered at the proper rate.

The dynamic test is often easier to perform on a complete decoding system, which may include a systems decoder, a video decoder and a display process. It may be possible to record the output of the display process and to check that display order and timing of decoded pictures are correct at the output of the display process. However, since the display process is not within the normative scope of Rec. ITU-T H.265 | ISO/IEC 23008-2, there may be cases where the output of the display process differs in timing or value even though the video decoder is conforming. In this case, the output of the video decoder itself (before the display process) would need to be captured in order to perform the dynamic tests on the video decoder. In particular the output order and timing of the decoded pictures shall be correct.

If buffering period and picture timing SEI messages are included in the test bitstream, HRD conformance shall be verified using the values of initial_cpb_removal_delay, initial_cpb_removal_delay_offset, cpb_removal_delay and dpb_removal_delay that are included in the bitstream.

If buffering period and picture timing SEI messages are not included in the bitstream, the following inferences shall be made to generate the missing parameters:

- fixed_pic_rate_flag shall be inferred to be equal to 1.
- low_delay_hrd_flag shall be inferred to be equal to 0.
- cbr_flag shall be inferred to be equal to 0.
- The frame rate of the bitstream shall be inferred to be equal to the frame rate value specified in the corresponding table of clause 6.7, where the bitstream is listed. If this is missing, then a frame rate of either 25 or 30000 ÷ 1001 can be inferred.

- time_scale shall be set equal to 90 000 and the value of num_units_in_tick shall be computed based on field rate (twice the frame rate).
- The bit rate of the bitstream shall be inferred to be equal to the maximum value for the level specified in Table A-1 in Rec. ITU-T H.265 | ISO/IEC 23008-2.
- CPB and DPB sizes shall be inferred to be equal to the maximum value for the level specified in Table A-1 in Rec. ITU-T H.265 | ISO/IEC 23008-2.

With the above inferences, the HRD shall be operated as follows.

- The CPB is filled starting at time t = 0, until it is full, before removal of the first access unit. This means that the initial_cpb_removal_delay shall be inferred to be equal to the total CPB buffer size divided by the bit rate divided by 90000 (rounded downwards) and initial_cpb_removal_delay_offset shall be inferred to be equal to zero.
- The first access unit is removed at time t = initial_cpb_removal_delay ÷ 90000 and subsequent access units are removed at intervals based on the frame distance, i.e., 2 * (90000 ÷ num_units_in_tick) or the field distance, i.e., (90000 ÷ num_units_in_tick), depending on whether the pictures in the bitstream are indicated to represent complete frames or individual fields of such frames.
- Using these inferences, the CPB will not overflow or underflow and the DPB will not overflow.

6.5.7 Decoder conformance test of a particular profile, tier, and level

In order for a decoder of a particular profile, tier, and level to claim output order conformance to Rec. ITU-T H.265 | ISO/IEC 23008-2 as specified by this Recommendation | International Standard, the decoder shall successfully pass the static test specified in clause 6.5.5 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile, tier, and level combination.

In order for a decoder of a particular profile, tier, and level to claim output timing conformance to Rec. ITU-T H.265 | ISO/IEC 23008-2 as specified by this Recommendation | International Standard, the decoder shall successfully pass both the static test specified in clause 6.5.5 and the dynamic test specified in clause 6.5.6 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile, tier, and level. Table 1 specifies the normative test suites for each profile, tier, and level combination. The test suite for a particular profile, tier, and level combination is the list of bitstreams that are marked with an 'X' in the column corresponding to that profile, tier, and level combination. In the column 'Main tier', 'X' indicate the bitstream is for Main tier. A decoder conformed to Main tier shall be capable of decoding the specified bitstreams, among the testing profile-level combination, indicated by 'X' at 'Main tier' column in Table 1. A decoder conformed to High tier shall be capable of decoding all the specified bitstreams, among the testing profile-level combination, in Table 1.

'X' indicates that the bitstream is designed to test both the dynamic and static conformance of the decoder.

The bitstream column specifies the bitstream used for each test.

A decoder that conforms to the Main profile, Main Still Picture profile, or Main 10 profile at a specific level shall be capable of decoding the specified bitstreams in Table 1.

6.6 Specification of the test bitstreams

6.6.1 General

Some characteristics of each bitstream listed in Table 1 are specified in this clause. In Table 1, the value "29.97" shall be interpreted as an approximation of an exact value of 30000 ÷ 1001 and the value "59.94" shall be interpreted as an approximation of an exact value of 60000 ÷ 1001.

6.6.2 Test bitstreams – Block structure

6.6.2.1 Test bitstreams #STRUCT_A

Specification: All slices are coded as I, P or B slices. Each picture contains one slice. Various CTU and maximum CU sizes are used.

Functional stage: Test the reconstruction process of slices.

Purpose: Check that the decoder can properly decode I, P and B slices with various CTU and maximum CU sizes.

6.6.2.2 Test bitstreams #STRUCT_B

Specification: All slices are coded as I, P or B slices. Each picture contains one slice. Various CTU and minimum CU sizes are used.

Functional stage: Test the reconstruction process of slices.

Purpose: Check that the decoder can properly decode I, P and B slices with various CTU and minimum CU sizes.

6.6.3 Test bitstreams – Intra coding

6.6.3.1 Test bitstreams #IPRED_A, #IPRED_B, and #IPRED_C

Specification: All slices are coded as I slices. Each picture contains one slice. All intra prediction modes (35 modes for each of luma 32x32, luma 16x16, luma 8x8, luma 4x4, chroma 16x16, chroma 8x8 and chroma 4x4, for a total 245 modes) are used. The IPRED_B bitstream contains only one picture, and conforms to the Main Still Picture profile.

Functional stage: Test the reconstruction process of I slices.

Purpose: Check that the decoder can properly decode I slices with all intra prediction modes.

6.6.3.2 Test bitstreams #CIP_A

Specification: The bitstream contains one I slice and one B slice, using one slice per picture. Both SAO and the deblocking filter are disabled.

Functional stage: Test the reference sample substitution process for intra sample prediction.

Purpose: Check that the decoder can properly decode slices of coded pictures containing intra TUs with unavailable samples for intra prediction.

6.6.3.3 Test bitstreams #CIP_B

Specification: The bitstream contains an I-picture and 4 P-pictures. Each picture contains only one slice. constrained_intra_pred_flag is equal to 1.

Functional stage: Test the reference sample substitution process for intra sample prediction.

Purpose: Check that the decoder can properly decode slices of coded pictures containing intra TUs with unavailable samples for intra prediction.

6.6.3.4 Test bitstreams #CIP_C

Specification: The bitstream contains one I slice and one B slice, using more than one slice per picture. Both SAO and the deblocking filter are disabled.

Functional stage: Test the reference sample substitution process for intra sample prediction.

Purpose: Check that the decoder can properly decode slices of coded pictures containing intra TUs with unavailable samples for intra prediction.

6.6.4 Test bitstreams – Inter frame coding

6.6.4.1 Test bitstreams #MERGE_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 4.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with the maximum number of merging candidates equal to any value permitted by the standard (i.e. 1, 2, 3, 4, 5).

6.6.4.2 Test bitstreams #MERGE_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 3.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with the maximum number of merging candidates equal to any value permitted by the standard (i.e. 1, 2, 3, 4, 5).

6.6.4.3 Test bitstreams #MERGE_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 2.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with the maximum number of merging candidates equal to any value permitted by the standard (i.e. 1, 2, 3, 4, 5).

6.6.4.4 Test bitstreams #MERGE_D

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 1.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with the maximum number of merging candidates equal to any value permitted by the standard (i.e. 1, 2, 3, 4, 5).

6.6.4.5 Test bitstreams #MERGE_E

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with the maximum number of merging candidates equal to any value permitted by the standard (i.e. 1, 2, 3, 4, 5).

6.6.4.6 Test bitstreams #MERGE_F

Specification: All slices are coded as I or B slices. Each picture contains only one slice. sps_temporal_mvp_enable_flag is equal to 0 and five_minus_max_num_merge_cand is equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode when the temporal merging candidate is not included in the merge candidate set.

6.6.4.7 Test bitstreams #MERGE_G

Specification: All slices are coded as I or B slices. Each picture contains only one slice. five_minus_max_num_merge_cand is set equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode with merge index ranging from 0 to 4.

6.6.4.8 Test bitstreams #PMERGE_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. log2_parallel_merge_level_minus2 is set equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode parallel merge level values permitted by the standard (i.e. 2, 3, 4, 5, 6 for luma CTB size 64x64).

6.6.4.9 Test bitstreams #PMERGE_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. log2_parallel_merge_level_minus2 is set equal to 1.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode parallel merge level values permitted by the standard (i.e. 2, 3, 4, 5, 6 for luma CTB size 64x64).

6.6.4.10 Test bitstreams #PMERGE_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. log2_parallel_merge_level_minus2 is set equal to 2.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode the parallel merge level values permitted by the standard (i.e. 2, 3, 4, 5, 6 for luma CTB size 64x64).

6.6.4.11 Test bitstreams #PMERGE_D

Specification: All slices are coded as I or B slices. Each picture contains only one slice. log2_parallel_merge_level_minus2 is set equal to 3.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode the parallel merge level values permitted by the standard (i.e. 2, 3, 4, 5, 6 for luma CTB size 64x64).

6.6.4.12 Test bitstreams #PMERGE_E

Specification: All slices are coded as I or B slices. Each picture contains only one slice. log2_parallel_merge_level_minus2 is set equal to 4.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode the parallel merge level values permitted by the standard (i.e. 2, 3, 4, 5, 6 for luma CTB size 64x64).

6.6.4.13 Test bitstreams #AMVP_A

Specification: All slices are coded as I or P slices. Each picture contains only one slice. num_ref_idx_l0_default_active_minus1 is equal to 0, num_ref_idx_l1_default_active_minus1 is equal to 0 and num_ref_idx_active_override_flag is equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode when motion vector scaling is not needed for spatial motion vector prediction candidate generation (all inter-coded PUs within the same slice have the same inter_pred_idc and ref_idx_l0).

6.6.4.14 Test bitstreams #AMVP_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. Multiple reference pictures are used. For some slices, num_ref_idx_l0_default_active_minus1 is equal to 3 and num_ref_idx_active_override_flag is equal to 0. For other B slices, num_ref_idx_l0_default_active_minus1 is equal to 1, num_ref_idx_l1_default_active_minus1 is equal to 1 and num_ref_idx_active_override_flag is equal to 0.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode when motion vector scaling is not needed for spatial motion vector prediction candidate generation.

6.6.4.15 Test bitstreams #AMVP_C

Specification: All slices are coded as I or P slices. Each picture contains only one slice.

Functional stage: Test the reconstruction process of motion vector prediction, specifically, motion vector prediction during the low delay condition.

Purpose: Check that the decoder can properly decode when motion vector scaling is not needed for spatial motion vector prediction candidate generation.

6.6.4.16 Test bitstreams #TMVP_A

Specification: Each picture contains only one slice. slice_temporal_mvp_enable_flag is set equal to 0 for pictures 0 to 8 and 1 for pictures 9 to 16.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode for different slice_temporal_mvp_enable_flag values.

6.6.4.17 Test bitstreams #MVDL1ZERO_A

Specification: The bitstream contains multiple B slices per picture. Mvd_l0_zero_flag is set equal to 1. Randomized on and off switching of the mvd_l1_zero_flag for multiple B slices.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode when the parsing of list 1 motion vector difference for bi-prediction varies according to values of mvd_l1_zero_flag.

6.6.4.18 Test bitstreams #MVCLIP_A

Specification: Each picture contains only one slice. Motion vector prediction and merge candidate motion vectors are clipped to 16-bit values. Clipped motion vector prediction and merge candidates are selected.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode when clipping of motion vector prediction and merge candidate motion vectors to 16-bit values occurs.

6.6.4.19 Test bitstreams #MVEDGE_A

Specification: Each picture contains only one slice. The bitstream includes motion vectors pointing to the padded edge regions in a picture.

Functional stage: Test the reconstruction process of motion vector prediction.

Purpose: Check that the decoder can properly decode motion vectors pointing to the padded edge regions of a picture.

6.6.4.20 Test bitstreams #WP_A

Specification: All slices are coded as I or P slices. Each picture contains only one slice. weighted_pred_flag is equal to 1. Plural reference indices are assigned to each reference picture.

Functional stage: Weighted sample prediction process for P slices with plural reference indices.

Purpose: Check that the decoder can properly decode weighted sample prediction for P slices with plural reference indices.

6.6.4.21 Test bitstreams #WP_B

Specification: All slices are coded as I, P or B slices. Each picture contains only one slice. weighted_pred_flag is equal to 1 and weighted_bipred_flag is equal to 1. Plural reference indices are assigned to each reference picture.

Functional stage: Weighted sample prediction process for P and B slices with plural reference indices.

Purpose: Check that the decoder can properly decode weighted sample prediction for P and B slices with plural reference indices.

6.6.5 Test bitstreams – Transform and quantization

6.6.5.1 Test bitstreams #RQT_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter and max_transform_hierarchy_depth_intra are both set equal to 0.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder can properly decode slices with residual quadtree with intra and inter depth equal to 0.

6.6.5.2 Test bitstreams #RQT_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter and max_transform_hierarchy_depth_intra are both set equal to 1.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with intra and inter depth equal to 1.

6.6.5.3 Test bitstreams #RQT_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter and max_transform_hierarchy_depth_intra are both set equal to 2.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with intra and inter depth equal to 2.

6.6.5.4 Test bitstreams #RQT_D

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter and max_transform_hierarchy_depth_intra are both set equal to 3.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with intra and inter depth equal to 3.

6.6.5.5 Test bitstreams #RQT_E

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter and max_transform_hierarchy_depth_intra are both set equal to 4.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with intra and inter depth equal to 4.

6.6.5.6 Test bitstreams #RQT_F

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter is set equal to 2 and max_transform_hierarchy_depth_intra is set equal to 0.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with different intra and inter depths.

6.6.5.7 Test bitstreams #RQT_G

Specification: All slices are coded as I or B slices. Each picture contains only one slice. max_transform_hierarchy_depth_inter is set equal to 0 and max_transform_hierarchy_depth_intra is set equal to 2.

Functional stage: Test the reconstruction process of slices with residual quadtree.

Purpose: Check that the decoder properly decodes slices with residual quadtree with different intra and inter depths.

6.6.5.8 Test bitstreams #TUSIZE_A

Specification: All slices are coded as I or P slices. Each picture contains only one slice. log2_min_transform_block_size_minus2 is set equal to 2. The maximum luma CB size is 64x64, the minimum luma CB size is 32x32, the minimum transform size for luma is 16x16 and for chroma is 8x8.

Functional stage: Test the reconstruction process of slices with limited minimum transform size.

Purpose: Check that the decoder properly decodes slices with residual quadtree with minimum transform size that are not the default 4x4.

6.6.5.9 Test bitstreams #DELTAQP_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The maximum luma CB size is equal to 64x64 and the minimum luma CB size is equal to 8x8. diff_cu_qp_delta_depth is set randomly to values in the range of 0 to 3. CuQpDeltaVal is set randomly from −26 to 25.

Functional stage: Test the reconstruction process of slices with nonzero values of CuQpDeltaVal.

Purpose: Check that the decoder properly decodes slices with different values of CuQpDeltaVal.

6.6.5.10 Test bitstreams #DELTAQP_B

Specification: All slices are coded as I, P or B slices. Each picture contains more than one slice. The maximum luma CB size is equal to 64x64 and the minimum luma CB size is equal to 8x8. CuQpDeltaVal is set randomly from −26 to 25. slice_cb_qp_offset and slice_cr_qp_offset are set randomly from −4 to 4.

Functional stage: Test the reconstruction process of slices with nonzero values of CuQpDeltaVal.

Purpose: Check that the decoder properly handles various combination of chroma QP offset.

6.6.5.11 Test bitstreams #DELTAQP_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The maximum luma CB size is equal to 64x64 and the minimum luma CB size is equal to 8x8. diff_cu_qp_delta_depth is set randomly to values in the range of 0 to 3. CuQpDeltaVal is set randomly from −26 to 25. In some TUs, the cbfLuma or cbfChroma is equal to 0.

Functional stage: Test the reconstruction process of slices with nonzero values of CuQpDeltaVal.

Purpose: Check that the decoder properly decodes slices with different values of CuQpDeltaVal.

6.6.5.12 Test bitstreams #INITQP_A

Specification: All slices are coded as I or B slices. The value of init_qp_minus26 is set from −26 to 25.

Functional stage: Test QP initialization based on init_qp_minus26.

Purpose: Check that the decoder properly decodes different init_qp_minus26 values.

6.6.5.13 Test bitstreams #SLIST_A

Specification: All slices are coded as I or B slices. Each picture contains one slice. One SPS and more than one PPS are included. The SPS includes scaling list data. One of the PPSs does not include scaling list data. In other PPSs, different scaling lists data is included. In each picture, the PPS is overriden. scaling_list_enabled_flag is set equal to 1.

Functional stage: Test the reconstruction process of scaling list. Tests switching of scaling list data in SPS and PPS.

Purpose: Check that the decoder properly decodes slices of coded frames with scaling list, with different coding modes of the scaling list, when no scaling list is included in the PPS and when scaling list data is included in the PPS.

6.6.5.14 Test bitstreams #SLIST_B

Specification: All slices are coded as I or B slices. Each picture contains one slice. More than one SPS and more than one PPS are included. One of the SPSs does not includes scaling list data. One of the PPSs does not include scaling list data. In other SPSs and PPSs, different scaling lists data is included. In each picture, the PPS is overriden. scaling_list_enabled_flag is set equal to 0 or 1.

Functional stage: Test the reconstruction process of scaling list. Tests switching of scaling list off, default scaling list and scaling list in parameter sets.

Purpose: Check that the decoder can properly decode slices of coded frames with scaling list, different coding modes of the scaling list and when there are multiple SPSs and PPSs.

6.6.5.15 Test bitstreams #SLIST_C

Specification: All slices are coded as I or B slices. Each picture contains one slice. One SPS and more than one PPS are included. The SPS does not include scaling list data. One of the PPSs does not include scaling list data. In other PPSs, different scaling lists data is included. In each picture, the PPS is overriden. scaling_list_enabled_flag is set equal to 1.

Functional stage: Test the reconstruction process of scaling list. Tests switching of default scaling list and scaling list in PPS.

Purpose: Check that the decoder can properly decode slices of coded frames with scaling list, different coding modes of the scaling list, when no scaling list data is present and switching of default scaling list and scaling list data in PPS occurs.

6.6.5.16 Test bitstreams #SLIST_D

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. More than one SPS and more than one PPS are included. One of the SPSs does not includes scaling list data. One of the PPSs does not include scaling list data. In other SPSs and PPSs, different scaling lists data is included. In each picture, the PPS is override. scaling_list_enabled_flag is set equal to 0 or 1.

Functional stage: Test the reconstruction process of scaling list. Tests switching of scaling list off, default scaling list and scaling list in parameter sets.

Purpose: Check that the decoder can properly decode slices of coded frames with scaling list, different coding modes of scaling list and when there are multiple SPSs and PPSs.

6.6.6 Test bitstreams – Deblocking filter

6.6.6.1 Test bitstreams #DBLK_A

Specification: All slices are coded as I, P or B slices. Each picture contains more than one slice. More than one PPS is used. QP is set randomly to values in the range of 22 to 51. pps_beta_offset_div2 is randomly set in each picture from −6 to 6. slice_beta_offset_div2 and slice_tc_offset_div2 are randomly set in each slice from −6 to 6.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode slices with various combinations of deblocking filter control parameters.

6.6.6.2 Test bitstreams #DBLK_B

Specification: All slices are coded as I, P or B slices. Each picture contains more than one slice. More than one PPS is used. pps_cb_qp_offset and pps_cr_qp_offset are randomly set to values in the range from −12 to 12. slice_cb_qp_offset and slice_cr_qp_offset are randomly set from −4 to 4.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode when the deblocking filter varies according to various combinations of QP.

6.6.6.3 Test bitstreams #DBLK_C

Specification: All slices are coded as I, P or B slices. Each picture contains more than one slice. pps_disable_deblocking_filter_flag is set equal to 0. slice_disable_deblocking_filter_flag is randomly set equal to 0 or 1.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode with the deblocking filter being enabled and disabled across slices.

6.6.6.4 Test bitstreams #DBLK_D

Specification: All slices are coded as I or B slices. Each picture contains more than one slice and tile. loop_filter_across_slices_enabled_flag is set equal to 0 and loop_filter_across_tiles_enabled_flag is set equal to 1.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode with the deblocking filter being enabled and disabled at slice and tile boundaries.

6.6.6.5 Test bitstreams #DBLK_E

Specification: All slices are coded as I or B slices. Each picture contains more than one slice and tile. loop_filter_across_slices_enabled_flag is set equal to 1 and loop_filter_across_tiles_enabled_flag is set equal to 0.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode with the deblocking filter being enabled and disabled at slice and tile boundaries.

6.6.6.6 Test bitstreams #DBLK_F

Specification: All slices are coded as I or B slices. Each picture contains more than one slice and tile. loop_filter_across_slices_enabled_flag is set equal to 0 and loop_filter_across_tiles_enabled_flag is set equal to 1.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode with the deblocking filter being enabled and disabled at slice and tile boundaries.

6.6.6.7 Test bitstreams #DBLK_G

Specification: All slices are coded as I or B slices. Each picture contains more than one slice and tile. loop_filter_across_slices_enabled_flag is set equal to 1 and loop_filter_across_tiles_enabled_flag is set equal to 0.

Functional stage: Test the deblocking filter process.

Purpose: Check that the decoder can properly decode with the deblocking filter being enabled and disabled at slice and tile boundaries.

6.6.7 Test bitstreams – Sample adaptive offset

6.6.7.1 Test bitstreams #SAO_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. sao_merge_left_flag and sao_merge_up_flag are randomly set equal to 0 or 1.

Functional stage: Test the reconstruction process of sample adaptive offset.

Purpose: Check that the decoder can properly decode withrandom SAO merge left/up flag values.

6.6.7.2 Test bitstreams #SAO_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice and contains more than one tile. slice_sao_luma_flag and slice_sao_chroma_flag are randomly set equal to 0 or 1.

Functional stage: Test the reconstruction process of sample adaptive offset.

Purpose: Check that the decoder can properly decode with tiles and randomly enabled SAO for luma and/or SAO for chroma per slice.

6.6.7.3 Test bitstreams #SAO_C

Specification: All slices are coded as I or P slices. Each picture contains only one slice. All SAO offset values in this bitstream have maximum allowed magnitude 7 and random sign.

Functional stage: Test the reconstruction process of sample adaptive offset.

Purpose: Check that the decoder can properly decode with maximum SAO offset values.

6.6.7.4 Test bitstreams #SAO_D

Specification: All slices are coded as I or P slices. Each picture contains only one slice. SAO offset values in this bitstream have random values in the range −7..7.

Functional stage: Test the reconstruction process of sample adaptive offset.

Purpose: Check that the decoder can properly decode with random SAO offset values.

6.6.7.5 Test bitstreams #SAO_E

Specification: All slices are coded as I or B slices. Each picture contains only one slice. A set of SAO parameters is associated with each CTB for all frames, therefore no SAO merge flags (up or left) are used. Only the band offset SAO type is used and the four SAO offset values are set equal to −7 or 7 in a random way. The luma CTB size is set equal to 16x16.

Functional stage: Tests loading of maximum SAO information at CTB level and frame.

Purpose: Check that the decoder can properly decode with the maximum possible SAO information.

6.6.7.6 Test bitstreams #SAO_F

Specification: All slices are coded as I or B slices. Each picture contains only one slice. A set of SAO parameters is associated to each CTB for all frames, therefore, no SAO merge flags (up or left) are used. Only the band offset SAO type is used and the four SAO offset values are set equal to −7 or 7 in a random way. The luma CTB size is set equal to 32x32.

Functional stage: Tests loading of maximum SAO information at CTB level and frame.

Purpose: Check that the decoder can properly decode with the maximum possible SAO information.

6.6.7.7 Test bitstreams #SAO_G

Specification: All slices are coded as I or B slices. Each picture contains only one slice. A set of SAO parameters is associated to each CTB for all frames, therefore, no SAO merge flags (up or left) are used. Only the band offset SAO type is used and the four SAO offset values are set equal to −7 or 7 in a random way. The luma CTB size is set equal to 64x64.

Functional stage: Tests loading of maximum SAO information at CTB level and frame.

Purpose: Check that the decoder can properly decode with the maximum possible SAO information.

6.6.8 Test bitstreams – Entropy coding

6.6.8.1 Test bitstreams #MAXBINS_A

Specification: All slices are coded as I slices. Each picture contains only one slice. The number of bins per CTU is constructed to be within 95% of the maximum number which is 4096 bits per CTU with luma CTB size 16x16. pcm_enabled_flag is set equal to 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder can properly decode slices with the maximum number of bins per CTU.

6.6.8.2 Test bitstreams #MAXBINS_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The number of bins per CTU is constructed to be within 95% of the maximum number which is 4096 bits per CTU with luma CTB size 16x16. pcm_enabled_flag is set equal to 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder can properly decode slices with the maximum number of bins per CTU.

6.6.8.3 Test bitstreams #MAXBINS_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The number of bins per CTU is constructed to be within 95% of the maximum number which is 4096 bits per CTU with luma CTB size 16x16. pcm_enabled_flag is set equal to 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder can properly decode slices with the maximum number of bins per CTU.

6.6.8.4 Test bitstreams #CAINIT_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. There is one PPS. cabac_init_present_flag is equal to 0 in PPS.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes when cabac_init_flag is not signalled in the slice header of P or B slices.

6.6.8.5 Test bitstreams #CAINIT_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for B slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes with different cabac_init_flag values in B slices.

6.6.8.6 Test bitstreams #CAINIT_C

Specification: All slices are coded as I or P slices. Each picture contains only one slice. There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for P slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes with different cabac_init_flag values in P slices.

6.6.8.7 Test bitstreams #CAINIT_D

Specification: All slices are coded as I or B slices which are uni-directionally predicted. Each picture contains only one slice. There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for B slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes with different cabac_init_flag values in P slices.

6.6.8.8 Test bitstreams #CAINIT_E

Specification: All slices are coded as I or P slices. Each picture contains only one slice. Each slice contains four tiles (two columns of tiles and two rows of tiles with uniform spacing). There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for P slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes when cabac_init_flag is switched in P slices with the use of tiles.

6.6.8.9 Test bitstreams #CAINIT_F

Specification: All slices are coded as I or uni-directionally predicted B slices. Each picture contains only one slice. Each slice contains four tiles (two columns of tiles and two rows of tiles with uniform spacing). There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for uni-directionally predicted B slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes when cabac_init_flag is switched in B slices with the use of tiles.

6.6.8.10 Test bitstreams #CAINIT_G

Specification: All slices are coded as I or P slices. Each picture contains only one slice. Each slice contains multiple dependent slice segments. Each dependent slice contains three CTUs or less. There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for P slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes when cabac_init_flag is switched in P slices with dependent slice segments.

6.6.8.11 Test bitstreams #CAINIT_H

Specification: All slices are coded as I or uni-directionally predicted B slices. Each picture contains only one slice. Each slice contains multiple dependent slice segments. Each dependent slice contains three CTUs or less. There is one PPS. cabac_init_present_flag is equal to 1 in PPS. cabac_init_flag is signalled for uni-directionally predicted B slices in the slice header referring the PPS. cabac_init_flag can take on values 0 or 1.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes when cabac_init_flag is switched in B slices with dependent slice segments.

6.6.8.12 Test bitstreams #SDH_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. sign_data_hiding_enabled_flag is set equal to 1. The bitstream includes various configurations of sign data hiding.

Functional stage: Test the parsing process.

Purpose: Check that the decoder properly decodes with sign data hiding.

6.6.9 Test bitstreams – Temporal scalability

6.6.9.1 Test bitstreams #TSCL_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The bitstream includes four temporal layers.

Functional stage: Test temporal scalability.

Purpose: Check that the decoder properly decodes temporal layers.

6.6.9.2 Test bitstreams #TSCL_B

Specification: All slices are coded as I or P slices. Each picture contains only one slice. The bitstream includes four temporal layers.

Functional stage: Test temporal scalability.

Purpose: Check that the decoder properly decodes temporal layers.

6.6.10 Test bitstreams – Parallel processing tools

6.6.10.1 Test bitstreams #TILES_A

Specification: All slices are coded as I or P slices. Each picture contains only one slice. num_tiles_columns_minus1 and num_tiles_rows_minus1 are set equal to 4, which is the maximum value for level 4.1. uniform_spacing_flag is set equal to 0. The values of column_width_minus1[i] and row_height_minus1[i] are set randomly for each picture. loop_filter_across_tiles_enabled_flag is set randomly for each picture.

Functional stage: Test dependency breaks at tile boundaries.

Purpose: Check that the decoder properly decodes when there is random non-uniform tile spacing with a maximum number of tiles and the deblocking filter is enabled and disabled at tile boundaries.

6.6.10.2 Test bitstreams #TILES_B

Specification: All slices are coded as I or P slices. Each picture contains random number of slices. All slice boundaries aligned with tile boundaries. num_tiles_columns_minus1 and num_tiles_rows_minus1 are set equal to 4, which is the maximum value for level 4.1. uniform_spacing_flag is set equal to 0. The values of column_width_minus1[i] and row_height_minus1[i] are set randomly for each picture. loop_filter_across_tiles_enabled_flag is set randomly for each picture. pps_loop_filter_across_slices_enabled_flag is set randomly for each frame. slice_loop_filter_across_slices_enabled_flag is set randomly for each slice.

Functional stage: Test dependency breaks at tile boundaries and enabling/disabling the deblocking filter at tile/slice boundaries.

Purpose: Check that the decoder properly decodes when there is random non-uniform tile spacing with a maximum number of tiles and the deblocking filter is enabled and disabled at tile and slice boundaries.

6.6.10.3 Test bitstreams #WPP_A

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.
- One slice in the frame, the QP of each CU is set equal to a value such that $\text{Abs}(QP - SliceQp_Y) > 2$.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\text{Abs}(QP - SliceQp_Y) > 2$.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\text{Abs}(QP - SliceQp_Y) > 2$.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that $\text{Abs}(QP - SliceQp_Y) > 2$.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.4 Test bitstreams #WPP_B

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 32x32 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.
- One slice in the frame, the QP of each CU is set equal to a value such that $\text{Abs}(QP - SliceQp_Y) > 2$.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.

- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $\mathrm{SliceQp}_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.5 Test bitstreams #WPP_C

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 16x16 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.
- One slice in the frame, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that $\mathrm{Abs}(\mathrm{QP} - \mathrm{SliceQp}_Y) > 2$.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.6 Test bitstreams #WPP_D

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The picture is one CTU wide. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. These are encoded with varying numbers of slices and slice segments. Even frames have fixed QP, while the QP established at the CU level in odd frames is set such that $\mathrm{Abs}(QP - SliceQp_Y) > 2$. The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized when a picture is one CTU wide. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when a picture is one CTU wide and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.7 Test bitstreams #WPP_E

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The picture is two CTUs wide. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. These are encoded with varying numbers of slices and slice segments. Even frames have fixed QP, while the QP established at the CU level in odd frames is set such that $\mathrm{Abs}(QP - SliceQp_Y) > 2$. The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized when a picture is two CTUs wide. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when a picture is two CTUs wide and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.8 Test bitstreams #WPP_F

Specification: entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The picture is three CTUs wide. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. These are encoded with varying numbers of slices and slice segments. Even frames have fixed QP, while the QP established at the CU level in odd frames is set such that Abs(QP − $SliceQp_Y$) > 2. The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized when a picture is three CTUs wide. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when a picture is three CTUs wide and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main profile or Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.10.9 Test bitstreams #ENTP_A

Specification: All slices are coded as I slices. Each picture contains only one slice. Four tiles are included in a picture. num_tiles_columns_minus1 and num_tiles_rows_minus1 are set equal to 1. uniform_spacing_flag is set equal to 1. There are some tiles in PicOrderCntVal equal to 4 that contains emulation prevention bytes.

Functional stage: Test entry point signalling.

Purpose: Check that the decoder properly decodes when entry point signalling in a slice header is used with tiles and emulation prevention bytes occur in the substream(s).

6.6.10.10 Test bitstreams #ENTP_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. Six tiles are included in a picture. num_tiles_columns_minus1 is set equal to 1 and num_tiles_rows_minus1 are set equal to 2. uniform_spacing_flag is set equal to 1. There are some pictures (e.g., PicOrderCntVal equal to 4, 6, 10, 12, 18, and 20) contains a tile in which emulation prevention bytes occur.

Functional stage: Test entry point signalling.

Purpose: Check that the decoder properly decodes when entry point signalling in a slice header is used with tiles and emulation prevention bytes occur in the substream(s).

6.6.10.11 Test bitstreams #ENTP_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. entropy_coding_sync_enabled_flag is set equal to 1.

Functional stage: Test entry point signalling.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is used.

6.6.11 Test bitstreams – Other coding tools

6.6.11.1 Test bitstreams #IPCM_A

Specification: All slices are coded as I slices. Each picture contains only one slice. pcm_enabled_flag is equal to 1. Both pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 are equal to 7. log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, and pcm_loop_filter_disable_flag are equal to 0, 2 and 0, respectively.

Functional stage: Test parsing of pcm_flag in the coding unit syntax.

Purpose: Check that the decoder properly decodes pcm_flags.

6.6.11.2 Test bitstreams #IPCM_B

Specification: All slices are coded as I slices. Each picture contains only one slice. pcm_enabled_flag is equal to 1. Both pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 are equal to 5. log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, and pcm_loop_filter_disable_flag are equal to 0, 1 and 0, respectively.

Functional stage: Test parsing of pcm_flags in the coding unit syntax. Test parsing of pcm_sample_luma and pcm_sample_chroma data in PCM sample syntax.

Purpose: Check that the decoder properly decodes pcm_flags, and pcm_sample_luma and pcm_sample_chroma data.

6.6.11.3 Test bitstreams #IPCM_C

Specification: All slices are coded as I slices. Each picture contains only one slice. pcm_enabled_flag is equal to 1. Both pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 are equal to 7. log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, and pcm_loop_filter_disable_flag are equal to 0, 1 and 1, respectively.

Functional stage: Test parsing of pcm_flags in the coding unit syntax. Test parsing of pcm_sample_luma and pcm_sample_chroma data in PCM sample syntax. Test skipping of loop filtering on samples associated with pcm_flag equal to 1.

Purpose: Check that the decoder properly decodes pcm_flags, pcm_sample_luma and pcm_sample_chroma data and skips loop filtering on samples associated with pcm_flag equal to 1.

6.6.11.4 Test bitstreams #IPCM_D

Specification: All slices are coded as I slices. Each picture contains only one slice. pcm_enabled_flag is equal to 1. Both pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 are equal to 7. log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, and pcm_loop_filter_disable_flag are equal to 0, 1 and 0, respectively. transquant_bypass_enable_flag is equal to 1.

Functional stage: Test parsing of pcm_flags in the coding unit syntax. Test parsing of pcm_sample_luma and pcm_sample_chroma data in PCM sample syntax. Test skipping of loop filtering on samples associated with both cu_transquant_bypass_flag and pcm_flag equal to 1.

Purpose: Check that the decoder properly decodes pcm_flags, pcm_sample_luma and pcm_sample_chroma data and skips loop filtering on samples associated with both cu_transquant_bypass_flag and pcm_flag equal to 1.

6.6.11.5 Test bitstreams #IPCM_E

Specification: Contain single coded picture. The coded picture contains only one intra slice. pcm_enabled_flag is equal to 1. pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 are equal to 5 and 7, respectively. log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, and pcm_loop_filter_disable_flag are equal to 1, 0 and 0, respectively.

Functional stage: Test parsing of pcm_flags in the coding unit syntax. Test parsing of pcm_sample_luma and pcm_sample_chroma data in PCM sample syntax with different bit depths.

Purpose: Check that the decoder can properly decode pcm_flags, and pcm_sample_luma and pcm_sample_chroma data with different pcm_sample_bit_depth_luma_minus1 and pcm_sample_bit_depth_chroma_minus1 values.

6.6.11.6 Test bitstreams #TS_A

Specification: Each picture contains only one slice. transform_skip_enabled_flag is set equal to 1 for pictures 0 to 8 and 0 for picture 9 to 16.

Functional stage: Test reconstruction process of slices with transform_skip_enabled_flag is equal to 1.

Purpose: Check that the decoder can properly decode transform_skip_enabled_flag.

6.6.11.7 Test bitstreams #AMP_A, #AMP_D, and #AMP_E

Specification: All slices are coded as I, P or B slices. Each picture contains only one slice. All asymmetric motion partition modes (2NxnU, 2NxND, nLx2N, nRx2N) are included.

Functional stage: Test reconstruction process of slices with amp_enabled_flag equal to 1.

Purpose: Check that the decoder can properly decode slices with all asymmetric motion partition modes.

6.6.11.8 Test bitstreams #AMP_B

Specification: All slices are coded as I, P or B slices. Each picture contains only one slice. Asymmetric motion partition is only utilized for PUs of which size is larger than minimum CU.

Functional stage: Test reconstruction process of slices with amp_enabled_flag equal to 1.

Purpose: Check that the decoder can properly decode slices with a constraint on asymmetric motion partition modes.

6.6.11.9 Test bitstreams #LS_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. All the CUs are transform/quantization/filtering bypass CUs. SAO and deblocking filter are enabled in the SPS and PPS.

Functional stage: Test reconstruction process of transform/quantization/filtering bypass coding.

Purpose: Check that the decoder can properly decode transform/quantization/filtering bypass coding.

6.6.11.10 Test bitstreams #LS_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. At least 50% of the CUs do not use transform/quantization/filtering bypass, and where there at are least 100 CUs in each of the following categories

- The luma CB is 64x64, it has cu_transquant_bypass_flag on, at least one of the neighbouring CUs uses SAO, at least one of the neighbouring CUs uses deblocking filtering.
- The luma CB is 32x32, it has cu_transquant_bypass_flag on, at least one of the neighbouring CUs uses SAO, at least one of the neighbouring CUs uses deblocking filtering.
- The luma CB is 16x16, it has cu_transquant_bypass_flag on, at least one of the neighbouring CUs uses SAO, at least one of the neighbouring CUs uses deblocking filtering.
- The luma CB is 8x8, it has cu_transquant_bypass_flag on, at least one of the neighbouring CUs uses SAO, at least one of the neighbouring CUs uses deblocking filtering.

Functional stage: Test reconstruction process of transform/quantization/filtering bypass coding.

Purpose: Check that the decoder can properly decode transform/quantization/filtering bypass coding.

6.6.12 Test bitstreams – High level syntax

6.6.12.1 Test bitstreams #NUT_A

Specification: All slices are coded as I or P slices. Each picture contains only one slice. Three temporal layers are used. The bitstream exercises various VCL NAL unit types.

Functional stage: Test decoding of various VCL NAL unit types.

Purpose: Check that the decoder can properly decode the VCL NAL unit types TRAIL_N, TRAIL_R, TSA_N, TSA_R, STSA_N, STSA_R, RADL_R, RASL_R, RADL_N, RASL_N, BLA_W_LP, BLA_W_DLP, BLA_N_LP, IDR_W_DLP, IDR_N_LP, and CRA_NUT.

6.6.12.2 Test bitstreams #FILLER_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. This bitstream contains some NAL units with nal_unit_type equal to 38 (filler data) at the end of every access unit.

Functional stage: Test decoding with filler data NAL units present.

Purpose: Check that the decoder can properly decode a bitstream containing NAL units with the NAL unit type FD_NUT.

6.6.12.3 Test bitstreams #VPSID_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. This bitstream contains two VPSs with the correct one having the vps_video_parameter_set_id value 4. The bitstream has 3 temporal layers and the correct VPS has the temporal_nesting_flag turned off.

Functional stage: Test decoding of vps_video_parameter_set_id.

Purpose: Check that the decoder properly decodes vps_video_parameter_set_id.

6.6.12.4 Test bitstreams #PS_B

Specification: All slices are coded as I or P slices. Each picture contains only one slice. sps_extension_flag is set equal to 1. Data is included after the sps_extension_flag.

Functional stage: Test decoding of VPS, SPS and PPS.

Purpose: Check that the decoder properly handles the extension_flag in SPS. This bitstream does not conform to the Main profile or Main 10 profile of the first edition of the HEVC specification since sps_extension_flag is equal to 1. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the SPS extension.

6.6.12.5 Test bitstreams #PPS_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. Bitstream includes multiple PPS signalling with random PPS parameters (constrained intra, transform skip, tile configurations, WP, loop filter, etc.) that get randomly selected by coded pictures.

Functional stage: Test decoding of PPS.

Purpose: Check that the decoder properly handles multiple PPSs being signalled.

6.6.12.6 Test bitstreams #SLPPLP_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The bitstream contains three temporal layers (two sublayers).

Functional stage: Test decoding of sub_layer_profile_present_flag and sub_layer_level_present_flag.

Purpose: Check that the decoder properly handles sub_layer_profile_present_flag and sub_layer_level_present_flag.

6.6.12.7 Test bitstreams #OPFLAG_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The value of output_flag_present_flag is 1, indicating that the syntax element pic_output_flag is signalled in the slice header.

Functional stage: Test parsing of output_flag_present_flag in the PPS. Test parsing of pic_output_flag in slice header syntax.

Purpose: Check that the decoder properly decodes slice headers containing pic_output_flag.

6.6.12.8 Test bitstreams #OPFLAG_B

Specification: All slices are coded as I slices. Each picture contains only one slice. The value of output_flag_present_flag is 1, indicating that the syntax element pic_output_flag is signalled in the slice header. Pictures with PicOrderCntVal equal to 39 and 73 are set to be not for output.

Functional stage: Test picture output.

Purpose: Check that the decoder properly decodes slice headers containing pic_output_flag.

6.6.12.9 Test bitstreams #OPFLAG_C

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The value of output_flag_present_flag is 1, indicating that the syntax element pic_output_flag is signalled in the slice header. Picture with PicOrderCntVal equal to 20, 31, 56, and 72 are set to be not for output.

Functional stage: Test picture output.

Purpose: Check that the decoder properly decodes slice headers containing pic_output_flag.

6.6.12.10 Test bitstreams #NoOutPrior_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The value of NoRaslOutputFlag and no_output_of_prior_pics_flag is equal to 1 when a CRA picture follows an end of sequence NAL unit. One CRA picture is included in the middle of the bitstream. An end of sequence NAL unit is present in the bitstream in the decoding order that is right before the CRA picture.

Functional stage: Test picture output.

Purpose: Check that the decoder properly decodes a CRA picture that occurs immediately after an end of sequence NAL unit which should result in all stored decoded pictures in the DPB to be removed without outputting them.

6.6.12.11 Test bitstreams #NoOutPrior_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. Two IDR pictures are included in the bitstream. The first IDR picture is the first picture in the bitstream. The second IDR picture is in the middle of the bitstream. The value of no_output_of_prior_pics_flag for the second IDR picture is set equal to 1.

Functional stage: Test picture output.

Purpose: Check that the decoder properly decodes an IDR picture with no_output_of_prior_pics_flag equal to 1 which should result in all stored decoded pictures in the DPB to be removed without outputting them.

6.6.12.12 Test bitstreams #PICSIZE_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. Each picture includes four tile columns.

Functional stage: Test picture size capability.

Purpose: Check that the decoder properly decodes pictures whose size is 1056x8440.

6.6.12.13 Test bitstreams #PICSIZE_B

Specification: All slices are coded as I or B slices. The bitstream is designed to test maximum height for level 5.1. Picture size is 8440x1056. Picture width is not a multiple of 16.

Functional stage: Test picture size capability.

Purpose: Check that the decoder properly decodes pictures whose size is 8440x1056 (picture width not a multiple of 16).

6.6.12.14 Test bitstreams #PICSIZE_C

Specification: All slices are coded as I or B slices. Each picture includes two tile columns.

Functional stage: Test picture size capability.

Purpose: Check that the decoder properly decodes pictures whose size is 528x4216.

6.6.12.15 Test bitstreams #PICSIZE_D

Specification: All slices are coded as I or B slices. The bitstream is designed to test maximum height for level 4.1. Picture size is 4216x528.

Functional stage: Test picture size capability.

Purpose: Check that the decoder properly decodes pictures whose size is 4216x528.

6.6.12.16 Test bitstreams #POC_A

Specification: All slices are coded as I or B slices. The bitstream is designed to test some rules related to PicOrderCntVal derivation.

Functional stage: Test PicOrderCntVal derivation process.

Purpose: Check that the decoder properly decodes different PicOrderCntVal values.

6.6.12.17 Test bitstreams #RAP_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The first picture in the bitstream is a CRA picture and is followed by seven RASL pictures that are not decodable. There are two subsequent CRA pictures with RASL pictures, following the first CRA picture in this bitstream. These subsequent RASL pictures should be decodable since the associated CRA picture is not the first CRA picture in the bitstream.

Functional stage: Test reconstruction process starting with a CRA picture followed by RASL pictures that cannot be decoded of slices with the inter RPS prediction.

Purpose: Check that the decoder properly decodes when the CRA picture is the first picture in the bitstream and is followed by RASL pictures that are not decodable.

6.6.12.18 Test bitstreams #RAP_B

Specification: All slices are coded as I or B slices. A CRA picture with leading pictures following end of sequence NAL unit is included. VPS, SPS and PPS are present in the bitstream repeatedly. The conformance window for cropping is signalled in the bitstream.

Functional stage: Test reconstruction process of a CRA picture.

Purpose: Check that the decoder properly decodes a CRA picture with leading pictures following an end of sequence NAL unit.

6.6.12.19 Test bitstreams #RPS_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The slice header includes the inter-RPS prediction method for sending the RPS for short-term pictures. The last three frames of this 44-frame sequence contain slice header RPS using the inter-RPS in addition to the RPS sent in the PPS.

Functional stage: Test reconstruction process of slices with inter-RPS prediction.

Purpose: Check that the decoder properly decodes slices using the inter-RPS prediction method.

6.6.12.20 Test bitstreams #RPS_B

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The bitstream includes random RPS signalling in slice headers along with random picture coding order within a series of pictures.

Functional stage: Test reconstruction process of slices without inter-RPS prediction.

Purpose: Check that the decoder properly decodes slices using the inter-RPS prediction method.

6.6.12.21 Test bitstreams #RPS_C

Specification: All slices are coded as I or P slices. Each picture contains only one slice. Two temporal layers are used. 15 reference pictures are used. The bitstream exercises short-term reference pictures in the RPS.

Functional stage: Test short-term RPS handling.

Purpose: Check that the decoder properly decodes when short-term picture handling is used in the RPS.

6.6.12.22 Test bitstreams #RPS_D

Specification: All slices are coded as I or P slices. Each picture contains only one slice. Two temporal layers are used. The bitstream exercises short-term and long-term reference pictures in the RPS.

Functional stage: Test long-term and short-term RPS handling.

Purpose: Check that the decoder properly decodes when long-term and short-term picture handling is used in the RPS.

6.6.12.23 Test bitstreams #RPS_E

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The bitstream includes random RPS signalling with LTRPs in slice headers along with random picture coding order within a series of pictures.

Functional stage: Test reconstruction process of slices without inter-RPS prediction.

Purpose: Check that the decoder properly decodes slices with long-term reference pictures without inter-RPS prediction.

6.6.12.24 Test bitstreams #RPS_F

Specification: The inter-RPS prediction signals some RPS entries that are not used by the current picture. (used_by_curr_pic_flag[j] equal to 0 and use_delta_flag[j] equal to 1).

Functional stage: Test reconstruction process of slices without inter-RPS prediction.

Purpose: Check that the decoder properly decode slices with the inter-RPS prediction method including RPS entries that are not used by the current picture.

6.6.12.25 Test bitstreams #LTRPSPS_A

Specification: The bitstream is coded under typical random access conditions, with the following modifications. Eight long-term reference picture candidates (four different slice_pic_order_cnt_lsb values and two values of used_by_curr_pic_lt_flag[i], giving a total of eight) are signalled in the SPS. The slice headers refer to long-term reference pictures that are either referred to from the SPS or may be explicitly signalled in the slice header. Reference picture list modification is applied on some pictures.

Functional stage: Test parsing of long_term_ref_pics_present_flag, num_long_term_ref_pics_sps, lt_ref_pic_poc_lsb_sps, and used_by_curr_pic_lt_sps_flag in SPS. Test parsing of num_long_term_sps and lt_idx_sps in slice header syntax.

Purpose: Check that the decoder can properly decode slice headers when long-term reference pictures from the list of candidates in the SPS are specified.

6.6.12.26 Test bitstreams #RPLM_A

Specification: All slices are coded as I or B slices. Each picture contains only one slice. The bitstream includes random reference picture list modification with varying list sizes.

Functional stage: Test reconstruction process of slices with reference list modification.

Purpose: Check that the decoder properly decodes slices with reference list modification.

6.6.12.27 Test bitstreams #RPLM_B

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. The bitstream includes random reference picture list modification with varying list sizes.

Functional stage: Test reconstruction process of slices with reference list modification.

Purpose: Check that the decoder properly decodes slices with reference list modification.

6.6.12.28 Test bitstreams #SLICES_A

Specification: Each picture contains more than one slice with different slice type.

Functional stage: Test reconstruction process of pictures comprising of slices with different slice_type values.

Purpose: Check that the decoder properly decodes pictures comprising of slices with different slice_type values.

6.6.12.29 Test bitstreams #DSLICE_A

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. dependent_slice_segments_enabled_flag is set equal to 1.

Functional stage: Test reconstruction process of independent and dependent slice segments.

Purpose: Check that the decoder properly decodes independent and dependent slice segments.

6.6.12.30 Test bitstreams #DSLICE_B

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. dependent_slice_segments_enabled_flag is set equal to 1. entropy_coding_sync_enabled_flag is set equal to 1.

Functional stage: Test reconstruction process of dependent slice segments.

Purpose: Check that the decoder properly decodes dependent slice segments in combination with entropy coding synchronization.

6.6.12.31 Test bitstreams #DSLICE_C

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. dependent_slice_segments_enabled_flag is set equal to 1. tiles_enabled_flag is set equal to 1.

Functional stage: Test reconstruction process of dependent slice segments.

Purpose: Check that the decoder properly decodes dependent slice segments in combination with tiles.

6.6.12.32 Test bitstreams #BUMPING_A

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. All pictures with PicOrderCntVal values in the range of 0 to 65 are to be output except the pictures with PicOrderCntVal values equal to 4, 5, 6, 7, 15, 21, 22, 23, 30, 31, 36, 37, 38, 39, 54, 55, and 56. Those pictures are not output since they have not been output yet when IRAP pictures with no_output_of_prior_pics_flag equal to 1 are encountered in the bitstream. Four temporal layers are used and IRAP pictures with no_output_of_prior_pics_flag equal to 1 are present in the bitstream.

Functional stage: Test bumping process.

Purpose: Check that the decoder properly handles tests output order conformance, in particular when applying the bumping process.

6.6.12.33 Test bitstreams #CONFWIN_A

Specification: All slices are coded as I or B slices. Each picture contains more than one slice. The value of conf_win_bottom_offset, conf_win_top_offset, conf_win_left_offset and conf_win_right_offset are set equal to 1.

Functional stage: Test conformance window usage.

Purpose: Check that the decoder properly handles conf_win_bottom_offset, conf_win_top_offset, conf_win_left_offset, and conf_win_right_offset.

6.6.12.34 Test bitstreams #HRD_A

Specification: All slices are coded as I or B slices. Each access unit contains four decoding units. sub_pic_cpb_params_present_flag is set equal to 1 and du_common_cpb_removal_delay_flag is set equal to 0.

Functional stage: Test decoding-unit-based HRD.

Purpose: Check that the decoder can properly decode with decoding-unit-based CPB removal time signalling.

6.6.12.35 Test bitstreams #EXT_A

Specification: A three-picture bitstream containing extension data in the VPS, SPS, PPS, and slice headers. Note that this bitstream is not a conforming bitstream, but conforming decoders are required to be able to parse it and decode the pictures in the bitstream.

Functional stage: Test decoding of bitstreams that contain extension data.

Purpose: Check that the decoder can properly handle extension data. This bitstream does not conform to the Main profile or Main 10 profile since extension data is present. However, Main profile and Main 10 profile decoders shall be able to decode this bitstream and ignore the extension data.

6.6.13 Test bitstreams – 10 bit

6.6.13.1 Test bitstreams #WP_A_MAIN10

Specification: All slices are coded as I or P slices. Each picture contains only one slice. bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. weighted_pred_flag is set equal to 1. Plural reference indices are assigned to each reference picture.

Functional stage: Weighted sample prediction process for P slices with plural reference indices.

Purpose: Check that the decoder can properly decode weighted sample prediction for P slices with plural reference indices.

6.6.13.2 Test bitstreams #WP_B_MAIN10

Specification: All slices are coded as I, P or B slices. Each picture contains only one slice. bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. weighted_pred_flag is set equal to 1 and weighted_bipred_flag is equal to 1. Plural reference indices are assigned to each reference picture.

Functional stage: Weighted sample prediction process for P and B slices with plural reference indices.

Purpose: Check that the decoder can properly decode weighted sample prediction for P and B slices with plural reference indices.

6.6.13.3 Test bitstreams #TSUNEQBD_A_MAIN10

Specification: Each picture contains only one slice. bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 1. In the PPS, transform_skip_enabled_flag set equal to 1. In residual_coding(), transform_skip_flag set equal to 1 for Y, Cb, Cr for both intra and inter prediction modes.

Functional stage: Test parsing and reconstruction process of slices with transform skip mode for luma and chroma with different bit depths.

Purpose: Check that the decoder can properly decode intra and inter prediction slices with transform_skip and unequal bit depth (luma: 10-bit, chroma: 9-bit).

6.6.13.4 Test bitstreams #DBLK_A_MAIN10

Specification: All slices are coded as I or B slices. Each picture contains only one slice. bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. Some QP values are set to negative values.

Functional stage: Test deblocking filter process for 10 bit video.

Purpose: Check that the decoder can properly decode negative values of QP that affect the deblocking filter process.

6.6.13.5 Test bitstreams #INITQP_B_MAIN10

Specification: All slices are coded as I or B slices. bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. The value of init_qp_minus26 is set from −38 to 25.

Functional stage: Test the initial QP.

Purpose: Check that the decoder can properly decode different init_qp_minus26 values.

6.6.13.6 Test bitstreams #WPP_A_MAIN10

Specification: bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.
- One slice in the frame, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.13.7 Test bitstreams #WPP_B_MAIN10

Specification: bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 32x32 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.

- One slice in the frame, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.13.8 Test bitstreams #WPP_C_MAIN10

Specification: bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 16x16 is used. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. The first three of these groups of eight pictures have pictures with the following characteristics:

- One slice in the frame, QP is constant.
- One slice in the frame, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, QP is constant.
- Maximum number of independent slice segments in the frame, at least one slice segment is one CTU long, at least one slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, QP is constant.

- Maximum number of dependent slice segments in the frame, at least one dependent slice segment is one CTU long, at least one dependent slice segment is two CTUs long, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.
- Random combination of independent/dependent slice segments, QP is constant.
- Random combination of independent/dependent slice segments, the QP of each CU is set equal to a value such that Abs(QP − $SliceQp_Y$) > 2.

The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. The final three of these groups of eight pictures feature a mixture of single slice pictures, and pictures coded using multiple slices and multiple slice segments. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized with different slice types. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when entropy coding synchronization is enabled and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.13.9 Test bitstreams #WPP_D_MAIN10

Specification: bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The picture is one CTU wide. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. Each of these groups of eight pictures are encoded with varying numbers of slices and slice segments. Even frames have fixed QP, while the QP established at the CU level in odd frames is set such that Abs(QP − $SliceQp_Y$) > 2. The first of these groups of eight pictures is coded using all Intra CUs, and the second is coded with CU skipping disabled. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized when a picture is one CTU wide. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.

Purpose: Check that the decoder properly decodes when a picture is one CTU wide and handle entry points when slice segment header extension data bytes are present. This bitstream does not conform to the Main 10 profile since slice_segment_header_extension_present_flag is equal to 1. However, Main 10 profile decoders shall be able to decode this bitstream and ignore the slice segment header extension data bytes.

6.6.13.10 Test bitstreams #WPP_E_MAIN10

Specification: bit_depth_luma_minus8 is set equal to 2 and bit_depth_chroma_minus8 is set equal to 2. entropy_coding_sync_enabled_flag is set equal to 1. A luma CTB size of 64x64 is used. The picture is two CTUs wide. The bitstream contains six repeated patterns of pictures with a particular ordering and referencing relationship, which are each eight pictures long. These are encoded with varying numbers of slices and slice segments. Even frames have fixed QP, while the QP established at the CU level in odd frames is set such that Abs(QP − $SliceQp_Y$) > 2. The first of these groups of eight pictures is coded using all Intra CUs, an dthe second is coded with CU skipping disabled. Random amounts of slice segment header extension bytes are encoded in each slice header.

Functional stage: Tests that entropy coding is correctly synchronized when a picture is two CTUs wide. Tests that the QP predictor is reset to $SliceQp_Y$ at the beginning of every CTU row. May be used to test handling of entry points by a parallel decoder.