INTERNATIONAL STANDARD

ISO/IEC 23094-4

First edition
2022-01

Information technology — General video coding —

Part 4: Conformance and reference software for essential video coding

Technologies de l'information — Codage vidéo général —

Partie 4: Conformité et logiciel de référence pour le codage vidéo essentiel

Reference number
ISO/IEC 23094-4:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23094 series can be found on the ISO website and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Information technology — General video coding —

Part 4: Conformance and reference software for essential video coding

1 Scope

This document specifies a set of tests and procedures designed to indicate whether encoders or decoders essential video coding (EVC), which contains tests and a reference software designed to verify whether bitstreams and decoders meet normative requirements specified in ISO/IEC 23094-1. An encoder can claim conformance to ISO/IEC 23094-1 if the bitstreams that it generates are conforming bitstreams. Characteristics of coded bitstreams and decoders are defined in ISO/IEC 23094-1. Decoder characteristics define the properties and capabilities of the applied decoding process. The capabilities of a decoder specify which bitstreams the decoder can decode and reconstruct. A bitstream can be decoded by a decoder if the characteristics of the bitstream are within the specified decoder capabilities.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23094-1:2020, *Information technology — General video coding — Part 1: Essential video coding*

ISO/IEC 9899, *Information technology — Programming languages — C*

ISO/IEC/IEEE 9945, *Information technology — Portable Operating System Interface (POSIX®) Base Specifications, Issue 7*

3 Terms and definitions

For the purposes of this document, the terms and definitions and symbols specified in ISO/IEC 23094-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
bitstream
sequence of bits, in the form of a NAL unit stream or a raw bitstream, that forms the representation of coded pictures and associated data forming one or more coded video sequences

3.2
decoder
embodiment of a process that operates on a bitstream and may conform to the decoding process requirements specified for conformance

Note 1 to entry: The decoder does not include the display process, which is outside the scope of this document

3.3
encoder
embodiment of a process that produces a bitstream

Note 1 to entry: The process is not specified in this document (except in regard to identification of the reference software encoder).

3.4
reference software decoder
software which may decode a *bitstream* (3.1) encoded according to the syntax structure

4 Abbreviated terms

ETM test model of essential video coding

HRD hypothetical reference decoder

5 Conventions

For the purposes of this document, relevant conventions are specified in Clause 5 of ISO/IEC 23094-1:2020.

6 Conformance testing for ISO/IEC 23094-1

6.1 Introduction

The following subclauses specify the tests for verifying conformance of video bitstreams as well as decoders. These tests make use of test data (bitstream test suites) provided as an electronic annex to this document and the reference software decoder specified in ISO/IEC 23094-1.

The electronic annex to this document is available at the following address:

— https://standards.iso.org/iso-iec/23094/-4/ed-1/en/

6.2 Bitstream conformance

Bitstream conformance is specified by Clause C.4 of ISO/IEC 23094-1:2020.

6.3 Decoder conformance

Decoder conformance is specified by Clause C.5 of ISO/IEC 23094-1:2020.

6.4 Procedure to test bitstreams

A bitstream that claims conformance with ISO/IEC 23094-1 should pass the following normative test.

The bitstream should be decoded by processing it with the reference software decoder. When processed by the reference software decoder, the bitstream should 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 ISO/IEC 23094-1.

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 ISO/IEC 23094-1:2020 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 ISO/IEC 23094-1:2020 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.

When testing a bitstream for conformance, it may also be useful to test whether or not the bitstream follows the informative recommendations specified in ISO/IEC 23094-1.

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 ISO/IEC 23094-1.

A verifier may not necessarily perform all stages of the decoding process specified in ISO/IEC 23094-1 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 profile_idc and level_idc corresponding to a set of specified constraints on a bitstream for which a decoder conforming to a specified profile and level is required in Annex A of ISO/IEC 23094-1:2020 to properly perform the decoding process.

6.5.2 Contents of the bitstream file

The conformance bitstreams are included in this document as an electronic attachment available at the following address:

— https://standards.iso.org/iso-iec/23094/-4/ed-1/en/

The following information is included in a single zipped file for each such bitstream.

— bitstream, and

— decoded pictures or hashes of decoded pictures (may not be present), and

— short description of the bitstream, and

— 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 should 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 ISO/IEC 23094-1:2020.

For output order conformance, it is a requirement that all of the decoded pictures specified for output in Annex C of ISO/IEC 23094-1:2020 should 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 should be (exactly equal to) the values specified in Clause 8 of ISO/IEC 23094-1:2020.

For output timing conformance, it is a requirement that a conforming decoder should also output the decoded samples at the rates and times specified in Annex C of ISO/IEC 23094-1:2020.

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

6.5.4 Recommendations

In addition to the requirements, it is desirable that conforming decoders implement various informative recommendations specified in ISO/IEC 23094-1. This subclause 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 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 should 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 should 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 ISO/IEC 23094-1:2020, and to verify that the HRD models (as specified by the CPB and DPB specification in Annex C of ISO/IEC 23094-1:2020) 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 scope of ISO/IEC 23094-1, 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 should be correct.

If buffering period and picture timing SEI messages are included in the test bitstream, HRD conformance should 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 should be made to generate the missing parameters:

— fixed_pic_rate_flag should be inferred to be equal to 1, and

— low_delay_hrd_flag should be inferred to be equal to 0, and

— cbr_flag should be inferred to be equal to 0, and

— The frame rate of the bitstream should be inferred to be equal to the frame rate value specified in the corresponding Table 1 of 6.7, where the bitstream is listed. If this is missing, then a frame rate of either 25 or 30 000 ÷ 1 001 can be inferred, and

— time_scale should be set equal to 90 000 and the value of num_units_in_tick should be computed based on frame rate, and

— The bit rate of the bitstream should be inferred to be equal to the maximum value for the level specified in Table A.1 in ISO/IEC 23094-1:2020, and

— CPB and DPB sizes should be inferred to be equal to the maximum value for the level specified in Table A.1 in ISO/IEC 23094-1:2020.

With the above inferences, the HRD should 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 should be inferred to be equal to the total CPB buffer size divided by the bit rate divided by 90 000 (rounded downwards) and initial_cpb_removal_delay_offset should be inferred to be equal to zero, and

— The first access unit is removed at time t = initial_cpb_removal_delay ÷ 90 000 and subsequent access units are removed at intervals based on the frame distance, i.e., 2 * (90 000 ÷ num_units_in_tick) or the field distance, i.e., (90 000 / num_units_in_tick), depending on whether the access unit is coded as a frame picture or field picture, and

— 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 and level

In order for a decoder of a particular profile and level to claim output order conformance to ISO/IEC 23094-1 as specified by this document, the decoder should successfully pass the static test specified in subclause 6.5.5 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile and level combination.

In order for a decoder of a particular profile and level to claim output timing conformance to ISO/IEC 23094-1 as specified by this document, the decoder should successfully pass both the static test specified in subclause 6.5.5 and the dynamic test specified in subclause 6.5.6 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile and level. Table 1 specify the normative test suites for each profile and level combination. The test suite for a particular profile and level combination is the list of bitstreams that are marked with an 'X' in the column corresponding to that profile and level combination.

'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 Baseline profile or Main profile with 10 bit-depth at a specific level should 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" should be interpreted as an approximation of an exact value of 30 000 ÷ 1 001 and the value "59,94" should be interpreted as an approximation of an exact value of 60 000 ÷ 1 001.

A set of test vectors have been provided for conformance and are available at the following address:

— https://standards.iso.org/iso-iec/23094/-4/ed-1/en/

6.6.2 Test bitstreams

6.6.2.1 Test bitstream BP_SET_A

Specification: Streams with sets of coding tools in Baseline profile.

Functional stage: Test the decoding process of Baseline profile.

Purpose: Check that the decoder can properly decode bitstreams in which the full set of coding tools in Baseline profile is enabled.

6.6.2.2 Test bitstream BP_SET_B

Specification: Streams with sets of coding tools in Baseline Still Picture profile.

Functional stage: Test the decoding process of Baseline Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which the full set of coding tools in Baseline Still Picture profile is enabled.

6.6.2.3 Test bitstream MP_MIN_A

Specification: Streams with all tools disabled in Main profile in which SPS tool flags are equal to 0 in Main profile.

Functional stage: Test the decoding process of Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which all the SPS indicated tools are disabled in Main profile.

6.6.2.4 Test bitstream MP_MIN_B

Specification: Streams with all tools disabled in Main Still Picture profile in which SPS tool flags are equal to 0.

Functional stage: Test the decoding process of Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which all the SPS indicated tools are disabled in Main Still Picture profile

6.6.2.5 Test bitstream MP_SET_A

Specification: Streams with all tools enabled in Main profile in which SPS tool flags are equal to 1 in Main profile.

Functional stage: Test the decoding process of Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which all the SPS indicated tools are enabled in Main profile.

6.6.2.6 Test bitstream MP_SET_B

Specification: Streams with all tools enabled in Main Still Picture profile in which SPS tool flags are equal to 1 in Main profile.

Functional stage: Test the decoding process of Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which all the SPS indicated tools are enabled in Main Still Picture profile.

6.6.2.7 Test bitstream CTU_A

Specification: Streams with max CTU size of 64 enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 64 is enabled in Main profile.

6.6.2.8 Test bitstream CTU_B

Specification: Streams with max CTU size of 32 enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 32 is enabled in Main profile.

6.6.2.9 Test bitstream CTU_C

Specification: Streams with max CTU size of 128 and min CTU size of 8 enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 128 and min CTU size of 8 are enabled in Main profile.

6.6.2.10 Test bitstream CTU_D

Specification: Streams with CTU size of 32 enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which CTU size of 32 is enabled in Main profile.

6.6.2.11 Test bitstream CTU_E

Specification: Streams with max CTU size of 64 enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 64 is enabled in Main Still Picture profile.

6.6.2.12 Test bitstream CTU_F

Specification: Streams with max CTU size of 32 enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 32 is enabled in Main Still Picture profile.

6.6.2.13 Test bitstream CTU_G

Specification: Streams with max CTU size of 128 and min CTU size of 8 enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which max CTU size of 128 and min CTU size of 8 are enabled in Main Still Picture profile.

6.6.2.14 Test bitstream CTU_H

Specification: Streams with CTU size of 32 enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which CTU size of 32 is enabled in Main Still Picture profile.

6.6.2.15 Test bitstream BTT_A

Specification: Streams with BTT tool disabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is disabled in Main profile.

6.6.2.16 Test bitstream BTT_B

Specification: Streams with only BTT tool enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only BTT tool is enabled in Main profile.

6.6.2.17 Test bitstream BTT_C

Specification: Streams with BTT tool enabled with only binary split on in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled with only binary split on in Main profile.

6.6.2.18 Test bitstream BTT_D

Specification: Streams with BTT tool enabled with only binary split on and only 1:1, 1:2, 2:1 ratio CU allowed in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled and only binary split on and only 1:1, 1:2, 2:1 ratio CU are allowed in Main profile.

6.6.2.19 Test bitstream BTT_E

Specification: Streams with BTT tool enabled with both binary and ternary split on and only 1:1, 1:2, 2:1 ratio CU allowed in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled and both binary and ternary split on and only 1:1, 1:2, 2:1 ratio CU are allowed in Main profile.

6.6.2.20 Test bitstream BTT_F

Specification: Streams with BTT tool disabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is disabled in Main Still Picture profile.

6.6.2.21 Test bitstream BTT_G

Specification: Streams with only BTT tool enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which only BTT tool is enabled in Main Still Picture profile.

6.6.2.22 Test bitstream BTT_H

Specification: Streams with BTT tool enabled with only binary split on in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled with only binary split on in Main Still Picture profile.

6.6.2.23 Test bitstream BTT_I

Specification: Streams with BTT tool enabled with only binary split on and only 1:1, 1:2, 2:1 ratio CU allowed in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled and only binary split on and only 1:1, 1:2, 2:1 ratio CU are allowed in Main Still Picture profile.

6.6.2.24 Test bitstream BTT_J

Specification: Streams with BTT tool enabled with both binary and ternary split on and only 1:1, 1:2, 2:1 ratio CU allowed in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which BTT tool is enabled and both binary and ternary split on and only 1:1, 1:2, 2:1 ratio CU are allowed in Main Still Picture profile.

6.6.2.25 Test bitstream BOUNDARY_A

Specification: Streams with all tools enabled in Main profile.

Functional stage: Test the decoding process of the boundary handling in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+8)x(128*M+112) is set in Main profile.

6.6.2.26 Test bitstream BOUNDARY_B

Specification: Streams with all tools enabled in Main profile.

Functional stage: Test the decoding process of the boundary handling in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+24)x(128*M+96) is set in Main profile.

6.6.2.27 Test bitstream BOUNDARY_C

Specification: Streams with all tools enabled in Main profile.

Functional stage: Test the decoding process of the boundary handling in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+40)x(128*M+80) is set in Main profile.

6.6.2.28 Test bitstream BOUNDARY_D

Specification: Streams with all tools enabled in Main profile.

Functional stage: Test the decoding process of the boundary handling in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+56)x(128*M+64) is set in Main profile.

6.6.2.29 Test bitstream BOUNDARY_E

Specification: Streams with all tools enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the boundary handling in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+8)x(128*M+112) is set in Main Still Picture profile.

6.6.2.30 Test bitstream BOUNDARY_F

Specification: Streams with all tools enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the boundary handling in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+24)x(128*M+96) is set in Main Still Picture profile.

6.6.2.31 Test bitstream BOUNDARY_G

Specification: Streams with all tools enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the boundary handling in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+40)x(128*M+80) is set in Main Still Picture profile.

6.6.2.32 Test bitstream BOUNDARY_H

Specification: Streams with all tools enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the boundary handling in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which pictures of resolution (128*N+56)x(128*M+64) is set in Main Still Picture profile.

6.6.2.33 Test bitstream SUCO_A

Specification: Streams with SUCO tool disabled in Main profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is disabled in Main profile.

6.6.2.34 Test bitstream SUCO_B

Specification: Streams with only SUCO tool enabled in Main profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only SUCO tool is enabled in Main profile.

6.6.2.35 Test bitstream SUCO_C

Specification: Streams with SUCO tool enabled and MaxSucoLog2Size equal to 5 in Main profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is enabled and the variable MaxSucoLog2Size is set equal to 5 in Main profile.

6.6.2.36 Test bitstream SUCO_D

Specification: Streams with SUCO tool enabled and MinSucoLog2Size equal to 5 in Main profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is enabled and the variable MinSucoLog2Size is set equal to 5 in Main profile.

6.6.2.37 Test bitstream SUCO_E

Specification: Streams with SUCO tool disabled in Main Still Picture profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is disabled in Main Still Picture profile.

6.6.2.38 Test bitstream SUCO_F

Specification: Streams with only SUCO tool enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which only SUCO tool is enabled in Main Still Picture profile.

6.6.2.39 Test bitstream SUCO_G

Specification: Streams with SUCO tool enabled and MaxSucoLog2Size equal to 5 in Main Still Picture profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is enabled and the variable MaxSucoLog2Size is set equal to 5 in Main Still Picture profile.

6.6.2.40 Test bitstream SUCO_H

Specification: Streams with SUCO tool enabled and MinSucoLog2Size equal to 5 in Main Still Picture profile.

Functional stage: Test the decoding process of the determined decoding order in the every depth in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which SUCO tool is enabled and the variable MinSucoLog2Size is set equal to 5 in Main Still Picture profile.

6.6.2.41 Test bitstream ADMVP_A

Specification: Streams with ADMVP tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ADMVP tool is disabled in Main profile.

6.6.2.42 Test bitstream ADMVP_B

Specification: Streams with only ADMVP tool enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ADMVP tool is enabled in Main profile.

6.6.2.43 Test bitstream ADMVP_C

Specification: Streams with only ADMVP tool enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, random access configuration.

Purpose: Check that the decoder can properly decode bitstreams in which ADMVP tool is enabled in Main profile, random access configuration.

6.6.2.44 Test bitstream ADMVP_D

Specification: Streams with only ADMVP tool enabled in Main profile (MP_MIN).

Functional stage: Test the decoding process of the inter prediction in Main profile, low delay configuration.

Purpose: Check that the decoder can properly decode bitstreams in which ADMVP tool is enabled in Main profile, low delay configuration.

6.6.2.45 Test bitstream ADMVP_E

Specification: Streams with ADMVP and dependent tools disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, random access configuration.

Purpose: Check that the decoder can properly decode bitstreams in which ADMVP and dependent tools are disabled in Main profile, random access configuration.

6.6.2.46 Test bitstream ADMVP_F

Specification: Streams with ADMVP and dependent tools disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, low delay configuration.

Purpose: Check that the decoder can properly decode bitstreams in which ADMVP and dependent tools are disabled in Main profile, low delay configuration.

6.6.2.47 Test bitstream ADMVP_G

Specification: Streams with only ADMVP tool enabled and temporal_mvp_assigned_flag set 0 in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ADMVP tool is enabled and temporal_mvp_assigned_flag is set 0 in Main profile.

6.6.2.48 Test bitstream ADMVP_H

Specification: Streams with only ADMVP and its dependent tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ADMVP and its dependent tools are enabled in Main profile.

6.6.2.49 Test bitstream AFF_A

Specification: Streams with Affine tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which Affine tool is disabled in Main profile.

6.6.2.50 Test bitstream AFF_B

Specification: Streams with only regular Affine prediction (EIF is never used).

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which regular affine prediction is enabled in Main profile.

6.6.2.51 Test bitstream AFF_C

Specification: Streams with only EIF Affine prediction.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which EIF affine prediction is enabled in Main profile.

6.6.2.52 Test bitstream AFF_D

Specification: Streams with both regular and EIF Affine predictions.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which regular and EIF Affine predictions are enabled in Main profile.

6.6.2.53 Test bitstream AFF_E

Specification: Streams with only Affine and ADMVP tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only Affine and ADMVP tools are enabled in Main profile.

6.6.2.54 Test bitstream AMVR_A

Specification: Streams with AMVR tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which AMVR tool is disabled in Main profile.

6.6.2.55 Test bitstream AMVR_B

Specification: Streams with only AMVR and its dependent tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only AMVR and its dependent tools are enabled in Main profile.

6.6.2.56 Test bitstream DMVR_A

Specification: Streams with DMVR tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which DMVR tool is disabled in Main profile.

6.6.2.57 Test bitstream DMVR_B

Specification: Streams with only DMVR and its dependent tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only DMVR and its dependent tools are enabled in Main profile.

6.6.2.58 Test bitstream MMVD_A

Specification: Streams with MMVD tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which MMVD tool is disabled in Main profile.

6.6.2.59 Test bitstream MMVD_B

Specification: Streams with only MMVD and its dependent tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only MMVD and its dependent tools are enabled in Main profile.

6.6.2.60 Test bitstream MMVD_C

Specification: Streams with only MMVD and its dependent tools enabled and mmvd_group_enable_flag set 0 in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only MMVD and its dependent tools are enabled and mmvd_group_enable_flag set 0 in Main profile.

6.6.2.61 Test bitstream HMVP_A

Specification: Streams with only HMVP and ADMVP tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, random access configuration.

Purpose: Check that the decoder can properly decode bitstreams in which only HMVP and ADMVP tools are enabled in Main profile, random access configuration.

6.6.2.62 Test bitstream HMVP_B

Specification: Streams with only HMVP and ADMVP tools enabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, low delay configuration.

Purpose: Check that the decoder can properly decode bitstreams in which only HMVP and ADMVP tools are enabled in Main profile, low delay configuration.

6.6.2.63 Test bitstream HMVP_C

Specification: Streams with HMVP tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, random access configuration.

Purpose: Check that the decoder can properly decode bitstreams in which HMVP tool is disabled in Main profile, random access configuration.

6.6.2.64 Test bitstream HMVP_D

Specification: Streams with HMVP tool disabled in Main profile.

Functional stage: Test the decoding process of the inter prediction in Main profile, low delay configuration.

Purpose: Check that the decoder can properly decode bitstreams in which HMVP tool is disabled in Main profile, low delay configuration.

6.6.2.65 Test bitstream EIPD_A

Specification: Streams with EIPD tool disabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which EIPD tool is disabled in Main profile.

6.6.2.66 Test bitstream EIPD_B

Specification: Streams with only EIPD and its dependent tools enabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only EIPD and its dependent tools are enabled in Main profile.

6.6.2.67 Test bitstream EIPD_C

Specification: Streams with constrained intra prediction tool enabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which constrained intra prediction is enabled in Main profile.

6.6.2.68 Test bitstream EIPD_D

Specification: Streams with only constrained intra prediction tool enabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only constrained intra prediction is enabled in Main profile.

6.6.2.69 Test bitstream EIPD_E

Specification: Streams with EIPD tool disabled in Main Still Picture profile.

Functional stage: Test the decoding process of the intra prediction in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which EIPD tool is disabled in Main Still Picture profile.

6.6.2.70 Test bitstream EIPD_F

Specification: Streams with only EIPD and its dependent tools enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the intra prediction in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which only EIPD and its dependent tools are enabled in Main Still Picture profile

6.6.2.71 Test bitstream IBC_A

Specification: Streams with IBC tool disabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which IBC tool is disabled in Main profile.

6.6.2.72 Test bitstream IBC_B

Specification: Streams with only IBC and EIPD tools enabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only IBC and EIPD tools are enabled in Main profile.

6.6.2.73 Test bitstream IBC_C

Specification: Streams with only IBC and EIPD tools enabled in Main profile.

Functional stage: Test the decoding process of the intra prediction in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which variable IBC sizes are used in Main profile.

6.6.2.74 Test bitstream CM_init_A

Specification: Streams with CM_INIT tool disabled in Main profile.

Functional stage: Test the decoding process of the context model initialization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which CM_INIT tool is disabled in Main profile.

6.6.2.75 Test bitstream CM_init_B

Specification: Streams with only CM_INIT tool enabled in Main profile.

Functional stage: Test the decoding process of the context model initialization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only CM_INIT tool is enabled in Main profile.

6.6.2.76 Test bitstream ADCC_A

Specification: Streams with only CM_INIT and ADCC tools enabled in Main profile.

Functional stage: Test the decoding process of the coefficient coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only CM_INIT and ADCC tools are enabled in Main profile.

6.6.2.77 Test bitstream ADCC_B

Specification: Streams with ADCC tool disabled in Main profile.

Functional stage: Test the decoding process of the coefficient coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ADCC tool is disabled in Main profile.

6.6.2.78 Test bitstream ADCC_C

Specification: Streams with ADCC and all other tools enabled in Main profile.

Functional stage: Test the decoding process of the coefficient coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ADCC and all other tools are enabled in Main profile.

6.6.2.79 Test bitstream IQT_A

Specification: Streams with IQT tool disabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which IQT tool is disabled in Main profile.

6.6.2.80 Test bitstream IQT_B

Specification: Streams with only IQT tool enabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only IQT tool is enabled in Main profile.

6.6.2.81 Test bitstream IQT_C

Specification: Streams with all tools enabled and slice_cb_qp_offset 1, slice_cr_qp_offset 2 in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which slice_cb_qp_offset 1 and slice_cr_qp_offset 2 are used in Main profile.

6.6.2.82 Test bitstream IQT_D

Specification: Streams with all tools enabled and slice_cb_qp_offset (-1), slice_cr_qp_offset (-2) in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which slice_cb_qp_offset (-1) and slice_cr_qp_offset (-2) are used in Main profile.

6.6.2.83 Test bitstream IQT_E

Specification: Streams with IQT tool disabled in Main Still Picture profile.

Functional stage: Test the decoding process of the transform and the quantization in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which IQT tool is disabled in Main Still Picture profile.

6.6.2.84 Test bitstream IQT_F

Specification: Streams with only IQT tool enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the transform and the quantization in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which only IQT tool is enabled in Main Still Picture profile.

6.6.2.85 Test bitstream IQT_G

Specification: Streams with all tools enabled and slice_cb_qp_offset 1, slice_cr_qp_offset 2 in Main Still Picture profile.

Functional stage: Test the decoding process of the transform and the quantization in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which slice_cb_qp_offset 1 and slice_cr_qp_offset 2 are used in Main Still Picture profile.

6.6.2.86 Test bitstream IQT_H

Specification: Streams with all tools enabled and slice_cb_qp_offset (-1), slice_cr_qp_offset (-2) in Main Still Picture profile.

Functional stage: Test the decoding process of the transform and the quantization in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which slice_cb_qp_offset (-1), slice_cr_qp_offset (-2) are used in Main Still Picture profile.

6.6.2.87 Test bitstream IQT_I

Specification: Streams with chroma QP mapping tables signalling in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which chroma QP mapping tables signalling is used in Main profile.

6.6.2.88 Test bitstream ATS_A

Specification: Streams with ATS tool disabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ATS tool is disabled in Main profile.

6.6.2.89 Test bitstream ATS_B

Specification: Streams with only ATS and IQT tools enabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ATS and IQT tools are enabled in Main profile.

6.6.2.90 Test bitstream DQP_A

Specification: Streams with DQP tool enabled in Baseline profile.

Functional stage: Test the decoding process of the transform and the quantization in Baseline profile.

Purpose: Check that the decoder can properly decode bitstreams in which DQP tool is enabled in Baseline profile.

6.6.2.91 Test bitstream DQP_B

Specification: Streams with only DQP tool enabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only DQP tool is enabled in Main profile.

6.6.2.92 Test bitstream DQP_C

Specification: Streams with DQP tool enabled in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which DQP tool is enabled in Main profile.

6.6.2.93 Test bitstream DQP_D

Specification: Streams with DQP tool enabled and cu_qp_delta_area set at 10 in Main profile.

Functional stage: Test the decoding process of the transform and the quantization in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which DQP tool is enabled and cu_qp_delta_area is set at 10 in Main profile.

6.6.2.94 Test bitstream ADDB_A

Specification: Streams with only ADDB tool enabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ADDB tool is enabled in Main profile.

6.6.2.95 Test bitstream ADDB_B

Specification: Streams with ADDB and all other tools enabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ADDB and all other tools are enabled in Main profile.

6.6.2.96 Test bitstream ALF_A

Specification: Streams with ALF and all other tools enabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ALF and all other tools are enabled in Main profile.

6.6.2.97 Test bitstream ALF_B

Specification: Streams with only ALF tool disabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ALF tool is disabled in Main profile.

6.6.2.98 Test bitstream ALF_C

Specification: Streams with only ALF tool enabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only ALF tool is enabled in Main profile.

6.6.2.99 Test bitstream HTDF_A

Specification: Streams with only HTDF tool disabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which HTDF tool is disabled in Main profile.

6.6.2.100 Test bitstream HTDF_B

Specification: Streams with only HTDF tool enabled in Main profile.

Functional stage: Test the decoding process of the loop filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only HTDF tool is enabled in Main profile.

6.6.2.101 Test bitstream HTDF_C

Specification: Streams with only HTDF tool disabled in Main Still Picture profile.

Functional stage: Test the decoding process of the loop filtering in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which HTDF tool is disabled in Main Still Picture profile.

6.6.2.102 Test bitstream HTDF_D

Specification: Streams with only HTDF tool enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the loop filtering in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which only HTDF tool is enabled in Main Still Picture profile.

6.6.2.103 Test bitstream DRA_A

Specification: Streams with DRA and all other tools enabled in Main profile.

Functional stage: Test the decoding process of the post filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which DRA and all other tools are enabled in Main profile.

6.6.2.104 Test bitstream DRA_B

Specification: Streams with only DRA tool enabled in Main profile.

Functional stage: Test the decoding process of the post filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only DRA tool is enabled in Main profile.

6.6.2.105 Test bitstream PIC_SLICE_TILE_A

Specification: Streams with 4x4 uniform tiles and 2 rectangular slices in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which 4x4 uniform tiles and 2 rectangular slices are used in Main profile.

6.6.2.106 Test bitstream PIC_SLICE_TILE_B

Specification: Streams with 5x3 uniform tiles and 1 slice in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which 5x3 uniform tiles and 1 slice are used in Main profile.

6.6.2.107 Test bitstream PIC_SLICE_TILE_C

Specification: Streams with 4x4 uniform tiles and 2 arbitrary slices in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which 4x4 uniform tiles and 2 arbitrary slices are used in Main profile.

6.6.2.108 Test bitstream PIC_SLICE_TILE_D

Specification: Streams with picture crop offsets enabled in Main profile.

Functional stage: Test the decoding process of the partitioning in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which picture crop offsets are enabled in Main profile.

6.6.2.109 Test bitstream PIC_SLICE_TILE_E

Specification: Streams with 4x4 uniform tiles and 2 rectangular slices in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which 4x4 uniform tiles and 2 rectangular slices are used in Main Still Picture profile.

6.6.2.110 Test bitstream PIC_SLICE_TILE_F

Specification: Streams with 5x3 uniform tiles and 1 slice in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which 5x3 uniform tiles and 1 slice are used in Main Still Picture profile.

6.6.2.111 Test bitstream PIC_SLICE_TILE_G

Specification: Streams with 4x4 uniform tiles and 2 arbitrary slices in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which 4x4 uniform tiles and 2 arbitrary slices are used in Main Still Picture profile.

6.6.2.112 Test bitstream PIC_SLICE_TILE_H

Specification: Streams with picture crop offsets enabled in Main Still Picture profile.

Functional stage: Test the decoding process of the partitioning in Main Still Picture profile.

Purpose: Check that the decoder can properly decode bitstreams in which picture crop offsets are enabled in Main Still Picture profile.

6.6.2.113 Test bitstream RPL_A

Specification: Streams with only RPL tool enabled in Main profile.

Functional stage: Test the decoding process of the reference frame indication in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only RPL tool is enabled in Main profile.

6.6.2.114 Test bitstream RPL_B

Specification: Streams with only RPL tool disabled in Main profile.

Functional stage: Test the decoding process of the reference frame indication in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which RPL tool is disabled in Main profile.

6.6.2.115 Test bitstream RPL_C

Specification: Streams with all tools enabled and changed RPL in Main profile.

Functional stage: Test the decoding process of the reference frame indication in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which changed RPL tool is disabled in Main profile.

6.6.2.116 Test bitstream POCS_A

Specification: Streams with only POCS tool disabled in Main profile.

Functional stage: Test the decoding process of the reference frame indication in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which POCS tool is disabled in Main profile.

6.6.2.117 Test bitstream POCS_B

Specification: Streams with only POCS tool enabled in Main profile.

Functional stage: Test the decoding process of the reference frame indication in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which only POCS tool is enabled in Main profile.

6.6.2.118 Test bitstream APS_A

Specification: Streams with more than 32 ALF APS in Main profile.

Functional stage: Test the decoding process of the APS signal in Main profile.

Purpose: Check that the decoder can properly manage ALF APS buffer and decoder ALF APS in Main profile. ALF is enabled for APS signalling, all other tools of Main profile are disabled (MP_MIN).

6.6.2.119 Test bitstream APS_B

Specification: Streams 32 APS and 64 PPS, wide range of DRA params in Main profile.

Functional stage: Test the decoding process of the DRA APS signal in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which DRA APS is used in Main profile. DRA is enabled for APS signalling, all other tools of Main profile are disabled (MP_MIN).

6.6.2.120 Test bitstream BF_A

Specification: Streams with ALF disabled across tile boundaries with 2 tile rows and 3 tile columns in Main profile.

Functional stage: Test the decoding process of the boundary filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ALF is disabled and with M tile rows and N tile columns are set in Main profile.

6.6.2.121 Test bitstream BF_B

Specification: Streams with ALF enabled across tile boundaries and with 3 tile rows and 5 tile columns in Main profile.

Functional stage: Test the decoding process of the boundary filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which ALF is enabled and with M tile rows and N tile columns are set in Main profile.

6.6.2.122 Test bitstream BF_C

Specification: Streams with loop filter enabled and 5x3 uniform tiles and 1 slice in Main profile.

Functional stage: Test the decoding process of the boundary filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which loop filter is enabled and MxN uniform tiles and K slice are set in Main profile.

6.6.2.123 Test bitstream BF_D

Specification: Streams with loop filter enabled and 4x4 uniform tiles and 2 rectangular slices in Main profile.

Functional stage: Test the decoding process of the boundary filtering in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which loop filter is enabled and MxN uniform tiles and K rectangular slices are set in Main profile.

6.6.2.124 Test bitstream NAL_A

Specification: Streams with wide range of signalled NAL units types in Main profile.

Functional stage: Test the decoding process of the NAL signal in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which wide range of signalled NAL units in Main profile. DRA and ALF are enabled for APS signalling, all other tools of main profile are disabled.

6.7 Test suites for ISO/IEC 23094-1

Table 1 — Bitstreams for Baseline and Main profiles

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
Tool set	Baseline profile tool set	Tool set of Baseline profile	BP_SET_A	X				5.1	60
	Baseline Still Picture profile tool set	Tool set of Baseline Still Picture profile	BP_SET_B			X		NA	NA
	Main profile minimum tool set	Minimum tool set of Main profile	MP_MIN_A		X			5.1	60
	Main Still Picture profile minimum tool set	Minimum tool set of Main Still Picture profile	MP_MIN_B				X	NA	NA
	Main profile tool set	All tools in Main profile enabled	MP_SET_A		X			5.1	60
	Main Still Picture profile tool set	All tools in Main Still Picture profile enabled	MP_SET_B				X	NA	NA
Block structure	CTU,CU	CTU = 64 (cb_max: 6, cb_min: 2, cu14_max: 6, tris_max: 6, tris_min: 4)	CTU_A		X			5.1	60
		CTU = 32 (cb_max: 5, cb_min: 2, cu14_max: 5, tris_max: 5, tris_min: 4, suco_max: 5)	CTU_B		X			5.1	60
		CTU = 128, minCU = 8 (cb_max: 7, cb_min: 3, cu14_max: 6, tris_max: 6, tris_min: 5)	CTU_C		X			5.1	60
		CTU = 32, minCU = 32 (cb_max: 5, cb_min: 5, cu14_max: 5, tris_max: 5, tris_min: 7) (picture size should be multiple of 32 in the current encoder.)	CTU_D		X			5.1	60
		CTU = 64 (cb_max: 6, cb_min: 2, cu14_max: 6, tris_max: 6, tris_min: 4)	CTU_E				X	NA	NA
		CTU = 32 (cb_max: 5, cb_min: 2, cu14_max: 5, tris_max: 5, tris_min: 4, suco_max: 5)	CTU_F				X	NA	NA
		CTU = 128, minCU = 8 (cb_max: 7, cb_min: 3, cu14_max: 6, tris_max: 6, tris_min: 5)	CTU_G				X	NA	NA
		CTU = 32, minCU = 32 (cb_max: 5, cb_min: 5, cu14_max: 5, tris_max: 5, tris_min: 7) (picture size should be multiple of 32 in the current encoder.)	CTU_H				X	NA	NA
X Bitstream is for static and dynamic tests									

Table 1 *(continued)*

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
	BTT (Binary and ternary split)	BTT structure Off test	BTT_A		X			5.1	60
		BTT structure On test based on MP_MIN (cb_max: 7, cb_min: 2, cu14_max: 6, tris_max: 6, tris_min: 4)	BTT_B		X			5.1	60
		Binary split on, ternary off (cb_max: 7, cb_min: 2, cu14_max: 6, tris_max: 2, tris_min: 4)	BTT_C		X			5.1	60
		Binary split on, ternary off, only1:1/1:2/2:1 ratio CUs allowed (cb_max: 7, cb_min: 2, cu14_max: 2, tris_max: 2, tris_min: 4)	BTT_D		X			5.1	60
		Binary split on, ternary on, only1: 1/1:2/2:1 ratio CUs allowed (cb_max: 7, cb_min: 2, cu14_max: 2, tris_max: 6, tris_min: 4)	BTT_E		X			5.1	60
		BTT structure Off test	BTT_F				X	NA	NA
		BTT structure On test based on MP_MIN (cb_max: 7, cb_min: 2, cu14_max: 6, tris_max: 6, tris_min: 4)	BTT_G				X	NA	NA
		Binary split on, ternary off (cb_max: 7, cb_min: 2, cu14_max: 6, tris_max: 2, tris_min: 4)	BTT_H				X	NA	NA
		Binary split on, ternary off, only1:1/1:2/2:1 ratio CUs allowed (cb_max: 7, cb_min: 2, cu14_max: 2, tris_max: 2, tris_min: 4)	BTT_I				X	NA	NA
		Binary split on, ternary on, only1: 1/1:2/2:1 ratio CUs allowed (cb_max: 7, cb_min: 2, cu14_max: 2, tris_max: 6, tris_min: 4)	BTT_J				X	NA	NA
	BOUNDARY (Boundary partition)	width=128*N+8, height=128*M+112	BOUNDARY_A		X			5.1	60
		width=128*N+24, height=128*M+96	BOUNDARY_B		X			5.1	60
		width=128*N+40, height=128*M+80	BOUNDARY_C		X			5.1	60
		width=128*N+56, height=128*M+64	BOUNDARY_D		X			5.1	60
		width=128*N+8, height=128*M+112	BOUNDARY_E				X	NA	NA
		width=128*N+24, height=128*M+96	BOUNDARY_F				X	NA	NA
		width=128*N+40, height=128*M+80	BOUNDARY_G				X	NA	NA
		width=128*N+56, height=128*M+64	BOUNDARY_H				X	NA	NA
X Bitstream is for static and dynamic tests									

Table 1 *(continued)*

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
	SUCO (Split unit coding ordering)	SUCO Off test	SUCO_A		X			5.1	60
		SUCO On test based on MP_MIN (default setting == (suco_max: 6, suco_min: 4))	SUCO_B		X			5.1	60
		suco_max: 5, suco_min: 4	SUCO_C					5.1	60
		suco_max: 6, suco_min: 5	SUCO_D					5.1	60
		SUCO Off test	SUCO_E				X	NA	NA
		SUCO On test based on MP_MIN (default setting == (suco_max: 6, suco_min: 4))	SUCO_F				X	NA	NA
		suco_max: 5, suco_min: 4	SUCO_G				X	NA	NA
		suco_max: 6, suco_min: 5	SUCO_H				X	NA	NA
Inter	ADMVP (Advanced motion vector prediction)	ADMVP Off test	ADMVP_A		X			5.1	60
		ADMVP On test based on MP_MIN and ADMVP On (dependent tools = off)	ADMVP_B		X			5.1	60
		ADMVP On test based on MP_MIN and ADMVP On (dependent tools = on)	ADMVP_C		X			5.1	60
		ADMVP On test based on MP_MIN and ADMVP On (dependent tools = on)	ADMVP_D		X			5.1	60
		ADMVP Off test (dependent tools=off)	ADMVP_E		X			5.1	60
		ADMVP Off test (dependent tools=off)	ADMVP_F		X			5.1	60
		ADMVP On test based on MP_MIN and ADMVP On (dependent tools = off, temporal_mvp_assigned_flag = 0)	ADMVP_G		X			5.1	60
		ADMVP On test based on MP_MIN and ADMVP On (dependent tools=on)	ADMVP_H		X			5.1	60
	AFF (Affine model based motion compensation)	AFF Off test	AFF_A		X			3.1	50
		Only regular affine prediction (EIF affine is never used)	AFF_B		X			3.0	60
		Only EIF affine is used for affine motion compensation	AFF_C		X			2.0	50
		Both EIF affine and regular affine are used for motion compensation	AFF_D		X			3.0	60
		AFF On test based on MP_MIN and ADMVP On	AFF_E		X			3.1	50
	AMVR (Adaptive motion vector resolution)	AMVR Off test	AMVR_A		X			5.1	60
		AMVR On test based on MP_MIN and ADMVP On	AMVR_B		X			5.1	60
	DMVR (Decoder side motion vector refinement)	DMVR Off test	DMVR_A		X			3.1	50
		DMVR On test based on MP_MIN and ADMVP On	DMVR_B		X			3.1	50
X Bitstream is for static and dynamic tests									

Table 1 *(continued)*

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
	MMVD (Merge with motion vector difference)	MMVD Off test	MMVD_A		X			5.1	60
		MMVD On test based on MP_MIN and ADMVP On (mmvd_group_enable_flag==1)	MMVD_B		X			5.1	60
		MMVD On test based on MP_MIN and ADMVP On (mmvd_group_enable_flag==0)	MMVD_C		X			5.1	60
	HMVP (History based motion vector prediction)	HMVP On test based on MP_MIN and ADMVP On	HMVP_A		X			5.1	60
		HMVP On test based on MP_MIN and ADMVP On	HMVP_B		X			5.1	50
		HMVP Off test	HMVP_C		X			5.1	60
		HMVP Off test	HMVP_D		X			5.1	50
Intra	EIPD (Extended intra prediction modes)	EIPD Off test	EIPD_A		X			5.1	60
		EIPD On test based on MP_MIN	EIPD_B		X			5.1	60
		Constrained intra prediction (on based on MP_SET)	EIPD_C		X			5.1	60
		Constrained intra prediction (on based on MP_MIN)	EIPD_D		X			5.1	60
		EIPD Off test	EIPD_E				X	NA	NA
		EIPD On test based on MP_MIN	EIPD_F				X	NA	NA
	IBC (Intra block copy)	IBC Off test	IBC_A		X			5.1	30
		IBC On test based on MP_MIN and EIPD On	IBC_B		X			5.1	30
		Exercise range of IBC sizes	IBC_C		X			5.1	30
Entropy	CM (Context initialization)	CM Off test	CM_init_A		X			5.1	60
		CM On test based on MP_MIN	CM_init_B		X			5.1	60
	ADCC (Advanced residual coding)	ADCC On test based on MP_MIN and CM On	ADCC_A		X			5.1	60
		ADCC Off test	ADCC_B		X			5.1	60
		ADCC On with MP_SET	ADCC_C		X			5.1	50
Transform & Quantization	IQT (Improved quantization and transform)	IQT Off test	IQT_A		X			3.1	50
		IQT On test based on MP_MIN	IQT_B		X			3.1	50
		Exercise range of Chroma QP offset (positive values)	IQT_C		X			3.1	50
		Exercise range of Chroma QP offset (negative values)	IQT_D		X			3.1	50
		IQT Off test	IQT_E				X	NA	NA
		IQT On test based on MP_MIN	IQT_F				X	NA	NA
		Exercise range of Chroma QP offset (positive values)	IQT_G				X	NA	NA
		Exercise range of Chroma QP offset (negative values)	IQT_H				X	NA	NA
		Exercise on Chroma QP mapping table, MP_SET	IQT_I		X			5.1	25
	ATS (Adaptive transform selection)	ATS Off test	ATS_A		X			3.1	50
		ATS On test based on MP_MIN and IQT On	ATS_B		X			3.1	50

X Bitstream is for static and dynamic tests

Table 1 *(continued)*

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
	DQP (Delta QP signalling)	DQP on using BP_SET	DQP_A	X				5.1	60
		DQP on using MP_MIN	DQP_B		X			5.1	60
		DQP on using MP_SET	DQP_C		X			5.1	60
		Exercise range of DQP sizes and DQP values (log2_cu_qp_delta_area: 10)	DQP_D		X			5.1	60
Loop-filter	ADDB (Advanced deblocking filter)	ADDB On test on MP_SET	ADDB_A		X			5.1	20
		ADDB On test based on MP_MIN	ADDB_B		X			5.1	50
	ALF (Adaptive loop filter)	ALF On test based on MP_SET	ALF_A		X			5.1	60
		ALF Off test	ALF_B		X			5.1	50
		ALF On test based on MP_MIN	ALF_C		X			5.1	60
	HTDF (Hadamard transform domain filter)	HTDF Off test	HTDF_A		X			3.1	50
		HTDF On test based on MP_MIN	HTDF_B		X			3.1	50
		HTDF Off test	HTDF_C				X	NA	NA
		HTDF On test based on MP_MIN	HTDF_D				X	NA	NA
Post-filter	DRA (Dynamical range adjustment)	DRA On test	DRA_A		X			5.1	24
		DRA On test based on MP_MIN	DRA_B		X			5.1	24
High level syntax	PIC_SLICE_TILE(Picture/Slices/Tile)	Pictures partitions in tiles and slices (4x4 uniform tiles and 2 rectangular slices)	PIC_SLICE_TILE_A		X			5.1	60
		Exercise tile combinations (5x3 non-uniform tiles and 1 slice)	PIC_SLICE_TILE_B		X			5.1	60
		Exercise on Arbitrary slices (4x4 uniform tiles and 2 arbitrary slices)	PIC_SLICE_TILE_C		X			5.1	60
		Exercise on picture size (4x4 uniform tiles and 2 arbitrary slices)	PIC_SLICE_TILE_D		X			5.1	60
		Pictures partitions in tiles and slices (4x4 uniform tiles and 2 rectangular slices)	PIC_SLICE_TILE_E				X	NA	NA
		Exercise tile combinations (5x3 non-uniform tiles and 1 slice)	PIC_SLICE_TILE_F				X	NA	NA
		Exercise on Arbitrary slices (4x4 uniform tiles and 2 arbitrary slices)	PIC_SLICE_TILE_G				X	NA	NA
		Exercise on picture size (4x4 uniform tiles and 2 arbitrary slices)	PIC_SLICE_TILE_H				X	NA	NA
	RPL (Reference picture lists)	RPL on test based on MP_MIN	RPL_A		X			3.1	50
		RPL off with other features	RPL_B		X			3.1	50
		Exercise RPL combinations using related syntaxes	RPL_C		X			3.1	50
	POCS	POCs Off test	POCS_A		X			3.1	50
		POCs On test based on MP_MIN	POCS_B		X			3.1	50
	APS (Adaptation parameter set)	Multiple APSs of each type(ALF)	APS_A		X			5.1	60
		Multiple APSs of each type (DRA)	APS_B		X			5.1	30

X Bitstream is for static and dynamic tests

Table 1 *(continued)*

Categories	Sub category	Description	Bitstream	Baseline	Main	Base Still Picture	Main Still Picture	Level	Frame rate (Frame/sec)
	BF (Tile / Slice boundary filtering)	ALF is disabled across tile boundaries (2x3 tiles grid)	BF_A		X			5.1	60
		ALF is enabled across tile boundaries (3x5 tiles grid)	BF_B		X			5.1	60
		Loop filters off across tile/slice boundaries	BF_C		X			5.1	60
		Loop filters off across tile/slice boundaries	BF_D		X			5.1	60
	NAL(NAL unit type)	Exercise all types of NAL units	NAL_A		X			5.1	50
X Bitstream is for static and dynamic tests									

7 Reference software description

7.1 General

ETM is implemented in C programming language, according to ISO/IEC 9899 and supports multiplatform build for a wide range of operation systems compliant with ISO/IEC/IEEE 9945.

ETM software includes five main projects:

— Encoder application project: the application project, which serves file I/O operations, performs encoder configuring and calls encoder API.

— Decoder application project: the application project, which serves file I/O operations, performs decoder configuring and calls decoder API

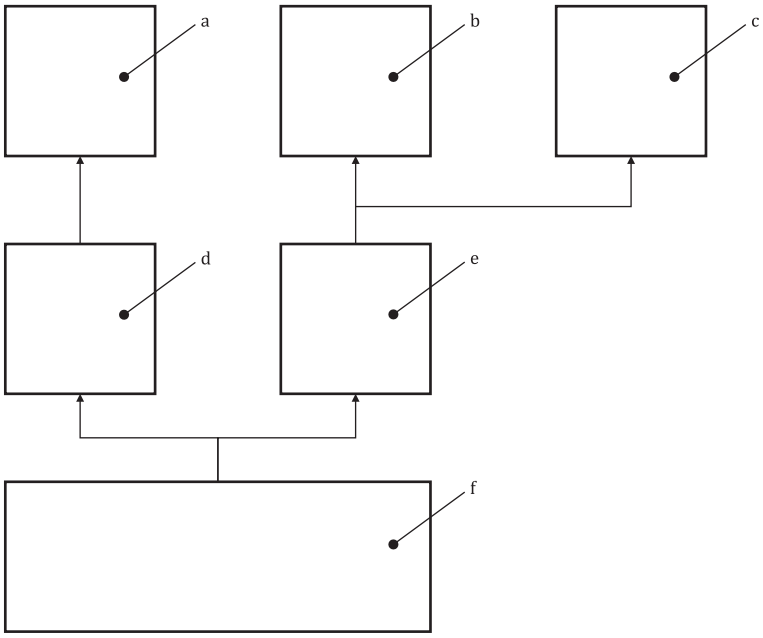
— Encoder library project: the library project, which encapsulates encoder routines and provides encoder API.

— Decoder library project: the library project, which encapsulates decoder routines and provides decoder API.

— Common library project: the library project, which encapsulates common routines for both encoder and decoder.

Besides five main projects, ETM software also includes bitstream merge service project which allows to merge several independently encoder bitstreams into the one joint valid bitstream.

Figure 1 demonstrates dependencies between the projects.



Key

a encoder application
b decoder application
c bitstream merge application
d encoder library
e decoder library
f common library

Figure 1 — The diagram of the ETM projects' dependencies

7.2 ETM repository

ETM software is available at https://standards.iso.org/iso-iec/23094/-4/ed-1/en/

7.3 Encoder and decoder usage

All executable files being called without parameters output their detailed usage. Some examples of basic encoder and decoder usage are given below.

EXAMPLE Encoder command line:

```
evca_encoder.exe -i BasketballDrill_832x480_50.yuv -o str.bin -w 832 -h 480 -q 37
-z 50 -p 48 -v 1 -f 17 -d 8 --output_bit_depth 10 -r rec.yuv --config
"..\..\cfg\encoder_randomaccess.cfg"
```

— -i, --input [STRING]: file name of input video

— -o, --output [STRING] (optional): file name of output bitstream

— -w, --width [INTEGER]: pixel width of input video

— -h, --height [INTEGER]: pixel height of input video

— -q, --op_qp [INTEGER] (optional): QP value (0~51)

— -z, --hz [INTEGER]: frame rate (Hz)

— -p, --iperiod [INTEGER] (optional): I-picture period
— -v, --verbose [INTEGER] (optional): verbose level
  — 0: no message
  — 1: frame-level messages (default)
  — 2: all messages
— -f, --frames [INTEGER] (optional): maximum number of frames to be encoded
— -d, --input_bit_depth [INTEGER] (optional): input bitdepth (8(default), 10)
— --output_bit_depth [INTEGER] (optional): output bitdepth (8, 10)(default: same as input bitdpeth)
— -r, --recon [STRING] (optional): file name of reconstructed video
— --config [STRING] (optional): file name of configuration.

EXAMPLE Decoder command line:

```
evca_decoder.exe -i str.bin -o rec_dec.yuv -v 1 --output_bit_depth 10
```

— evca_decoder.exe -i str.bin -o rec_dec.yuv -v 1 --output_bit_depth 10
— -i, --input [STRING]: file name of input bitstream
— -o, --output [STRING] (optional): file name of decoded output
— -v, --verbose [INTEGER] (optional): verbose level
  — 0: no message
  — 1: frame-level messages (default)
  — 2: all messages
— --output_bit_depth [INTEGER] (optional): output bitdepth (8(default), 10).

ICS 35.040.40

Price based on 32 pages