**ISO/IEC 23090-23:2021(E)**

ISO/IEC JTC 1/SC 29/WG 04

Date: 2021-05-21

**Information technology — Coded Representation of Immersive Media — Part 23: Conformance and Reference Software for MPEG Immersive Video**

WD stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

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

[Foreword iv](#_Toc72493451)

[Introduction v](#_Toc72493452)

[1 Scope 1](#_Toc72493453)

[2 Normative references 1](#_Toc72493454)

[3 Terms and definitions 1](#_Toc72493455)

[4 Abbreviated terms 2](#_Toc72493456)

[5 Conventions 2](#_Toc72493457)

[6 Conformance testing 2](#_Toc72493458)

[6.1 Introduction 2](#_Toc72493459)

[6.2 Bitstream conformance 2](#_Toc72493460)

[6.3 Decoder conformance 2](#_Toc72493461)

[6.4 Reference bitstreams 2](#_Toc72493462)

[6.5 Procedure to test bitstreams 2](#_Toc72493463)

[6.6 Procedure to test decoders 3](#_Toc72493464)

[6.6.1 Conformance bitstreams 3](#_Toc72493465)

[6.6.2 Contents of the bitstream zip-files 3](#_Toc72493466)

[6.6.3 Requirements on decoder output and timing 3](#_Toc72493467)

[7 Reference software 4](#_Toc72493468)

[7.1 Software location 4](#_Toc72493469)

[7.2 Software license 4](#_Toc72493470)

[7.3 Software installation 4](#_Toc72493471)

[7.4 Software architecture 4](#_Toc72493472)

[7.4.1 Reference software encoder 4](#_Toc72493473)

[7.4.2 Reference software decoder 4](#_Toc72493474)

[8 Decoder output logging process 4](#_Toc72493475)

[8.1 General decoder output logging process 4](#_Toc72493476)

[8.2 General hashing process 5](#_Toc72493477)

[8.3 Video data hashing process 5](#_Toc72493478)

[8.4 Block to patch map hashing process 6](#_Toc72493479)

[8.5 Patch params list hashing process 6](#_Toc72493480)

[8.6 View params list hashing process 6](#_Toc72493481)

[Annex A (informative) Validation and verification of candidate bitstreams 8](#_Toc72493482)

[A.1 General validation and verification of candidate bitstreams 8](#_Toc72493483)

[A.2 Application of alternative encoders 8](#_Toc72493484)

[A.3 Application of alternative decoders 8](#_Toc72493485)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 ISO documents 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](https://www.iso.org/directives-and-policies.html)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO 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](https://www.iso.org/iso-standards-and-patents.html)).

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](https://www.iso.org/foreword-supplementary-information.html).

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 23090 series can be found on the ISO website.

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](https://www.iso.org/members.html).

Introduction

This document specifies how tests can be designed to verify whether bitstreams and decoders meet the requirements as specified in ISO/IEC 23090-12.

ISO/IEC 23090-12 was developed to support compression of immersive video content, in which a real or virtual 3D scene is captured by multiple real or virtual cameras. The use of this document enables storage and distribution of immersive video content over existing and future networks, for playback with 6 degrees of freedom of view position and orientation.

Information technology — Coded Representation of Immersive Media — Part 23: Conformance and Reference Software for MPEG Immersive Video

# Scope

This document specifies a set of tests and procedures designed to indicate whether encoders or decoders meet the normative requirements specified in ISO/IEC 23090-12.

# 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 DIS 23090‑12, *Coded Representation of Immersive Media — Part 12: MPEG Immersive Video*

# Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC DIS 23090‑12 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 <http://www.electropedia.org/>

3.

alternative decoder

a *decoder* (3.5) other than the *reference software decoder* (3.8)

3.

alternative encoder

an *encoder* (3.7) other than the *reference software encoder* (3.9)

3.3

bitstream

sequence of bits that forms the representation of *coded volumetric frames* and associated data forming one or more *coded MIV sequences*

3.

bitstream zip-file

an archive file containing a bitstream plus files with information on that bitstream for the purpose of decoder conformance testing

3.5

decoder

embodiment of the decoding process specified by ISO/IEC DIS 23090‑12

NOTE 1 – The decoding process does not include post-decoding, reconstruction and display process, which are outside of the scope of this document.

3.

decoder output log

log-file according to the format specified in clause 8 of this document, that is output by a *decoder* (3.5) in response to a *bitstream* (3.3) for the purpose of decoder conformance testing

3.

encoder

embodiment of a process, not specified in this document, that produces a *bitstream* (3.5)

3.

reference software decoder

the *decoder* (3.5) that is specified in subclause 7.4.2 of this document

3.

reference software encoder

the *encoder* (3.7) that is specified in subclause 7.4.1 of this document

# Abbreviated terms

For the purposes of this document, the abbreviated terms of ISO/IEC DIS 23090‑12 apply.

# Conventions

For the purposes of this document, the conventions specified in ISO/IEC DIS 23090‑12 apply.

# Conformance testing

## Introduction

The following clauses specify normative tests for verifying the conformance of bitstreams as well as decoders. Those normative tests make use of test data (bitstream test suites) provides as an electronic annex to this document and the reference software decoder.

## Bitstream conformance

The bitstream conformance of ISO/IEC DIS 23090‑12 is specified in that document.

## Decoder conformance

The decoder conformance of ISO/IEC DIS 23090‑12 is specified in that document.

## Reference bitstreams

[Ed.(BK): Attached to this document is a single example zip-file based on release candidate 9.0-rc1 of TMIV.]

Table 1: List of reference bitstreams

|  |  |  |
| --- | --- | --- |
| Name | Profile-tier-level | Features |
|  |  |  |
|  |  |  |
|  |  |  |

## Procedure to test bitstreams

A bitstream that claims conformance with ISO/IEC DIS 23090‑12 shall pass the following normative test:

The bitstream shall be decoded by processing it with the decoder. When processed by the 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 ISO/IEC DIS 23090‑12.

Additional tests may be necessary to more thoroughly check that the bitstream properly meets all the requirements specified in ISO/IEC DIS 23090‑12 including the hypothetical reference decoder (HRD) conformance, supplemental enhancement information messages (SEI), and volumetric usability information (VUI). These complementary tests may be performed using other bitstream verifiers that perform more complete tests than those implemented by the reference software decoder.

ISO/IEC DIS 23090‑12 contains several informative recommendations that are not an integral part of that document. When testing a bitstream for conformance, it may also be useful to test whether or not the bitstream follows those recommendations.

To check the 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 DIS 23090‑12.

A verifier may not necessarily perform all stages of the decoding process specified in ISO/IEC DIS 23090‑12 to verify bitstream correctness. Many tests can be performed on syntax elements in a state prior to their use in some processing stages.

## Procedure to test decoders

### Conformance bitstreams

A bitstream has values of ptl\_profile\_codec\_group\_idc, ptl\_profile\_toolset\_idc and ptl\_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 to properly perform the decoding process.

### Contents of the bitstream zip-files

The conformance bitstreams are available as an electronic attachment to this document. The following information is included in a single zip-file for each bitstream:

* the test bitstream in V3C sample stream format, with file extension .bit,
* a decoder output log, as specified in clause 8, with file extension .dec.
* a short textual description of the bitstream, with file extension .txt, including
  + contact information,
  + profile-tier-level information,
  + a description of the encoder, including version information, that was used to create the bitstream,
  + the version of the reference software decoder that was used to create the decoder output log,
  + a description of the alternative decoders (if any), including version information, that were used to verify the decoder output log,
  + a list of features of ISO/IEC 23090-12 that are specifically exercised by this bitstream.

### Requirements on decoder output and timing

The output of the decoding process is specified in ISO/IEC DIS 23090‑12.

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 ISO/IEC DIS 23090‑12.

Post-decoding, reconstruction and display processes are outside of the scope of this document.

# Reference software

## Software location

The reference software is available at the following locations:

* MPEG-internal location: <http://mpegx.int-evry.fr/software/MPEG/MIV/RS/TM1>
* Public mirror location: <https://gitlab.com/mpeg-i-visual/tmiv>

## Software license

The license is included with the software:

<http://mpegx.int-evry.fr/software/MPEG/MIV/RS/TM1/-/blob/main/LICENSE>

## Software installation

A software manual that includes build and run instructions is included with the software:

<http://mpegx.int-evry.fr/software/MPEG/MIV/RS/TM1/-/blob/main/README.md>

## Software architecture

The reference software consists of multiple executables and libraries.

### Reference software encoder

The reference software encoder consists of the Encoder and Multiplexer executables. To encode a bitstream, the following steps are followed:

1. Invoke the Encoder to output:
   1. a partial bitstream that does not include the video sub bitstreams,
   2. an uncompressed video data file per video sub bitstream.
2. For each uncompressed video data file:
   1. Invoke a video encoder to encode the video data file, outputting a video sub bitstream,
3. Invoke the Multiplexer to output a bitstream.

### Reference software decoder

The reference software decoder consists of a single executable named Decoder.

The reference software decoder is configurable to output a decoder output log according to the format that is specified in clause 8.

# Decoder output logging process

[Ed.(BK): This clause and its sub-clauses are written in a combination of informal English and C++17. To be improved in the next version of this document. Ambiguity may be removed by using variables and arrays defined in MIV.]

## General decoder output logging process

The decoder output log is a text file with a space-separated table and LF line endings. An example of this format is:

0 0 1024 1024 e3767bcc 34af3963 c5aa7e7b d9546fc1  
0 1 1024 1024 c9b8f52e 0ba8e5c2 6d33a137 d9546fc1  
1 0 1024 1024 e3cecd72 34af3963 c5aa7e7b d9546fc1  
1 1 1024 1024 193f5983 0ba8e5c2 6d33a137 d9546fc1  
2 0 1024 1024 6097b60f fbe26652 b9e13c08 896070ed  
2 1 1024 1024 42868cd4 0ba8e5c2 1cf0ff47 896070ed

There is one row per volumetric frame and atlas. The rows are ordered by frame order count (presentation order) and atlas ID.

The columns in order from left to right are:

* Frame order count
* Atlas ID
* Atlas frame width
* Atlas frame height
* Decoded video data hash, specified in sub-clause 8.3
* Block to patch map hash, specified in sub-clause 8.4
* Patch params list hash, specified in sub-clause 8.5
* View params list hash, specified in sub-clause 8.6

All hashes are formatted as an eight-digit zero-padded lowercase hexadecimal number.

## General hashing process

This process calculates the CRC-32 hash[[1]](#footnote-1).

The input to this process is a byte string.

The output of this process is the value *hashValue* in range 0 .. 232 - 1.

First a table *crc32Table* of length 256 is calculated (once):

for (i = 0; i < 0x100; i++) {  
 n = i  
 for (j = 0; j < 8; j++) {  
 n = (n & 1) == 0 ? n >> 1 : (n >> 1) ^ 0xEDB88320  
 }  
 crc32Table[ i ] = n  
}

The variable *state* is initialized to 0xFFFFFFFF.

For each byte *n* in 0 .. 255 of the byte string, in order of increasing byte position, the variable *state* is updated as follows:

state = (state >> 8) ^ crc32Table[ (state ^ n) & 0xFF ]

Finally, the output of the process is:

hashValue = ~state

## Video data hashing process

The input of this process is a decoded MIV access unit and an atlas ID atlasId.

The output of this process is the value *hashValue* in range 0 .. 232 - 1.

This process applies sub-clause 8.2 on a byte stream that is formed by concatenating the following values using u(32) descriptors (network order):

1. The decoded geometry samples of the atlas with atlas ID atlasId (if any) ordered by row and column,
2. The decoded occupancy samples of the atlas with atlas ID atlasId (if any) ordered by row and column,
3. The decoded attribute samples of type texture of the atlas with atlas ID atlasId (if any) ordered by plane, row and column,
4. The decoded attribute samples of type transparency of the atlas with atlas ID atlasId (if any) ordered by row and column,
5. The decoded packed video samples of the atlas with atlas ID atlasId (if any) ordered by plane, row and column.

[Ed.(BK): Too late for WD1, but in hindsight, and in my opinion, it would be better to have separate columns per V3C component, e.g. V3C\_GVD 1920 1080 89abcdef for bullet 1.]

Bitstreams for which vuh\_attribute\_partition\_index != 0 or vuh\_map\_index != 0 or vuh\_auxiliary\_video\_flag != 0 are outside of the scope of this document.

NOTE 1: Because post-decoding processes are outside of the scope of this document, video frames may not be at nominal resolution.

## Block to patch map hashing process

The input of this process is a decoded block to patch map and an atlas ID atlasId.

The output of this process is the value *hashValue* in range 0 .. 232 - 1.

This process applies sub-clause 8.2 on a byte stream that is formed by concatenating the samples of the block to patch map using u(32) descriptors, whereby the value -1 is mapped to 0xFFFFFFFF.

## Patch params list hashing process

auto patchParamsListHash(const MivBitstream::PatchParamsList &ppl) noexcept

-> HashFunction::Result {

auto hash = HashFunction{};

for (const auto &pp : ppl) {

hash.consume(pp.atlasPatch3dOffsetU());

hash.consume(pp.atlasPatch3dOffsetV());

hash.consume(pp.atlasPatch3dOffsetD());

hash.consume(pp.atlasPatch3dRangeD());

hash.consume(pp.atlasPatchProjectionId());

hash.consume(static\_cast<uint32\_t>(pp.atlasPatchOrientationIndex()));

hash.consume(pp.atlasPatchLoDScaleX());

hash.consume(pp.atlasPatchLoDScaleY());

hash.consume(pp.atlasPatchEntityId().value\_or(0));

hash.consume(pp.atlasPatchDepthOccMapThreshold().value\_or(0));

hash.consume(pp.atlasPatchAttributeOffset()[0]);

hash.consume(pp.atlasPatchAttributeOffset()[1]);

hash.consume(pp.atlasPatchAttributeOffset()[2]);

hash.consume(static\_cast<uint32\_t>(pp.atlasPatchInpaintFlag()));

}

return hash.result();

}

## View params list hashing process

auto viewParamsListHash(const MivBitstream::ViewParamsList &vpl) noexcept -> HashFunction::Result {

auto hash = HashFunction{};

for (const auto &vp : vpl) {

hash.consume(static\_cast<uint32\_t>(vp.ci.ci\_cam\_type()));

hash.consume(vp.ci.ci\_projection\_plane\_width\_minus1());

hash.consume(vp.ci.ci\_projection\_plane\_height\_minus1());

switch (vp.ci.ci\_cam\_type()) {

case MivBitstream::CiCamType::equirectangular:

hash.consumeF(vp.ci.ci\_erp\_phi\_max());

hash.consumeF(vp.ci.ci\_erp\_theta\_max());

break;

case MivBitstream::CiCamType::perspective:

hash.consumeF(vp.ci.ci\_perspective\_focal\_hor());

hash.consumeF(vp.ci.ci\_perspective\_focal\_ver());

hash.consumeF(vp.ci.ci\_perspective\_center\_hor());

hash.consumeF(vp.ci.ci\_perspective\_center\_ver());

break;

case MivBitstream::CiCamType::orthographic:

hash.consumeF(vp.ci.ci\_ortho\_width());

hash.consumeF(vp.ci.ci\_ortho\_height());

break;

}

hash.consumeF(vp.pose.position[0]);

hash.consumeF(vp.pose.position[1]);

hash.consumeF(vp.pose.position[2]);

// TODO(#465): This may be too brittle

// orientation = w + i x + j y + k z

hash.consumeF(static\_cast<float>(vp.pose.orientation[3])); // w

hash.consumeF(static\_cast<float>(vp.pose.orientation[0])); // x

hash.consumeF(static\_cast<float>(vp.pose.orientation[1])); // y

hash.consumeF(static\_cast<float>(vp.pose.orientation[2])); // z

hash.consumeF(vp.dq.dq\_norm\_disp\_low());

hash.consumeF(vp.dq.dq\_norm\_disp\_high());

hash.consume(vp.dq.dq\_depth\_occ\_threshold\_default());

}

return hash.result();

}

1. (informative)  
     
   Validation and verification of candidate bitstreams
   1. General validation and verification of candidate bitstreams

A candidate bitstream zip-file should be validated and verified by at least two organizations. The following aspects shall be checked:

* Validate that all intended features are included in the bitstream
* Validate that the bitstream zip-file is complete and correctly named
* Verify that the reference software decoder can reproduce the decoder output log
  1. Application of alternative encoders

It is encouraged to create bitstreams using alternative encoders.

There is no requirement to make alternative encoders available.

There is no requirement on the reference software encoder to be able to produce a bitstream with the same features as a bitstream created by an alternative encoder.

* 1. Application of alternative decoders

It is encouraged to create the decoder output log using an alternative decoder.

It is encouraged to verify the decoder output log using an alternative decoder.

There is no requirement to make alternative decoders available.

If an assumedly conformant bitstream cannot be correctly decoded by the reference software decoder, then it is a condition of acceptance of that bitstream that the reference software decoder is improved.

1. [https://en.wikipedia.org/wiki/Cyclic\_redundancy\_check#CRC-32\_algorithm](https://en.wikipedia.org/wiki/Cyclic_redundancy_check%23CRC-32_algorithm) [↑](#footnote-ref-1)