International Standard

ISO/IEC 23094-2

First edition
2021-11

AMENDMENT 1
2024-01

Information technology – General video coding —

Part 2: Low complexity enhancement video coding

AMENDMENT 1: Additional levels

Technologies de l'information – Codage vidéo général —

Partie 2: Codage vidéo d'amélioration de faible complexité

AMENDEMENT 1: Niveaux supplémentaires

Reference number
ISO/IEC 23094-2:2021/Amd. 1:2024(en)

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

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).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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 can be found on the ISO 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 2:
Low complexity enhancement video coding

AMENDMENT 1: Additional levels

Normative references

Replace "ITU-T H.273 | ISO/IEC 23091-2:2019" with "ITU-T H.273 | ISO/IEC 23091-2".

6.2

Replace:

The variables ShiftWidthC and ShiftHeightC are specified in Table 2, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. Other values of chroma_format_idc, ShiftWidthC and ShiftHeightC may be specified in the future by ISO/IEC.

with:

The variables ShiftWidthC and ShiftHeightC are specified in Table 2, depending on the chroma format sampling structure, which is specified through chroma_sampling_type.

7.3.9

Replace Table 13 with the following table:

Table 13 — Process payload – surface

Syntax	Descriptor
process_surface(surface) {	
if (surface.entropy_enabled_flag) {	
if (compression_type_size_per_tile == 0) {	
surface.size	mb
}	
if (surface.rle_only_flag) {	
surface.data_rle	surface.size
} else {	
surface.data_prefix_coding	surface.size
}	
}	
}	

7.3.12

Replace Table 16 with the following table:

Table 16 — Byte alignment syntax

Syntax	Descriptor
byte_alignment( ) {	
while(!byte_aligned( ))	
alignment_bit_equal_to_zero /* equal to 0 */	f(1)
}	

7.4.2.2

In Table 19, replace "per picture (if no_enhancement_bit_flag == 0)" *with:*

per picture (if no_enhancement_bit_flag == 0 or temporal_signalling_present == 1)

Add the following sentence below Table 19:

If a NAL unit as specified in Sec. 7.3.2 contains more than one payload of the same payload_type (where payload_type is equal to 0, 1, or 2), the values given by the last payload of such payload_type within the NAL unit shall be used.

7.4.3.3

Replace

In order to prevent incomplete TUs, as defined in 6.3.2, custom_tile_width shall be an integer multiple of the TU size (nTbS = 2 if transform_type is equal to 0 and nTbS = 4 if transform_type is equal to 1) for each sub-layer and for each plane within a sub-layer.

with

In order to prevent incomplete entropy encoded quantized transform coefficient tiles, as defined in 9.1.1, custom_tile_width shall be an integer multiple of 64 for each sub-layer and for each plane within a sub-layer.

Replace

In order to prevent incomplete Tus, as defined in 6.3.2, custom_tile_height shall be an integer multiple of the TU size (nTbS = 2 if transform_type is equal to 0 and nTbS = 4 if transform_type is equal to 1) for each sub-layer and for each plane within a sub-layer.

with

In order to prevent incomplete entropy encoded quantized transform coefficient tiles, as defined in 9.1.1, custom_tile_height shall be an integer multiple of 64 for each sub-layer and for each plane.

Replace

planes_type specifies the planes to be processed by the decoder according to Table 25.

with

planes_type specifies the planes to be processed by the decoder according to Table 25. If chroma_sampling_type is equal to 0, planes_type shall be equal to 0.

7.4.3.4

Replace:

temporal_signalling_present_flag specifies whether the temporal signalling coefficient group is present in the bitstream.

with:

temporal_signalling_present_flag specifies whether the temporal signalling coefficient group is present in the bitstream. If temporal_enabled_flag is equal to 0 or temporal_refresh_bit_flag is equal to 1, temporal_signalling_present_flag shall be equal to 0.

8.3.2

Delete the following:

— variable nPlanes is derived as follows:

```
if (processed_planes_type_flag == 0)
  nPlanes = 1
else
  nPlanes = 3
```

Delete "data = read_byte(bitstream)" *from the condition branch below*

```
                    if (surfaces[planeIdx][levelIdx][layerIdx].rle_only_flag) {
                        multibyte = read_multibyte(bitstream)
                        surfaces[planeIdx][levelIdx][layerIdx].size = multibyte
                        surfaces[planeIdx][levelIdx][layerIdx].data_rle =
                          bytestream_current(bitstream)
                    } else {
                        data = read_byte(bitstream)
                        multibyte = read_multibyte(bitstream)
                        surfaces[planeIdx][levelIdx][layerIdx].size = multibyte
                        surfaces[planeIdx][levelIdx][layerIdx].data_prefix_coding =
                            bytestream_current(bitstream)
                        bytestream_seek(bitstream, surfaces[planeIdx][levelIdx][layerIdx].size)
                    }
```

Delete "data = read_byte(bitstream)" *from the condition branch below*

```
    if (temporal_signalling_present_flag == 1) {
        if (temporal_surfaces[planeIdx].entropy_enabled_flag) {
            if (temporal_surfaces[planeIdx].rle_only_flag) {
                multibyte = read_multibyte(bitstream)
                temporal_surfaces[planeIdx].size = multibyte
                temporal_surfaces[planeIdx].data_rle = bytestream_current(bitstream)
            } else {
                data = read_byte(bitstream)
                multibyte = read_multibyte(bitstream)
                temporal_surfaces[planeIdx].size = multibyte
                temporal_surfaces[planeIdx].data_prefix_coding =
                bytestream_current(bitstream)
                bytestream_seek(bitstream, temporal_surfaces[planeIdx].size)
            }
```

```
            }
        }
    }
```

8.3.3

Delete the following:

— variable nPlanes is derived as follows:

```
f (processed_planes_type_flag == 0)
  nPlanes = 1
else
  nPlanes = 3
```

Delete the instruction "data = read_byte(bitstream)" *from the condition branch below:*

```
if (surfaces[planeIdx][levelIdx][layerIdx].rle_only_flag) {
    multibyte = read_multibyte(bitstream)
    surfaces[planeIdx][levelIdx][layerIdx].tiles[tileIdx].size = multibyte
    surfaces[planeIdx][levelIdx][layerIdx].tiles[tileIdx]data_rle =
            bytestream_current(bitstream)
} else {
    data = read_byte(bitstream)
    multibyte = read_multibyte(bitstream)
    surfaces[planeIdx][levelIdx][layerIdx].tiles[tileIdx].size = multibyte
    surfaces[planeIdx][levelIdx][layerIdx].tiles[tileIdx].data_prefix_
    coding = bytestream_current(bitstream)
    bytestream_seek(bitstream,
                surfaces[planeIdx][levelIdx][layerIdx].tiles[tileIdx].size)
}
```

Delete the instruction "data = read_byte(bitstream)" *from the condition branch below:*

```
if (temporal_surfaces[planeIdx].rle_only_flag) {
    multibyte = read_multibyte(bitstream)
    temporal_surfaces[planeIdx].tiles[tileIdx].size = multibyte
    temporal_surfaces[planeIdx].tiles[tileIdx].data_rle =
        bytestream_current(bitstream)
} else {
    data = read_byte(bitstream)
    multibyte = read_multibyte(bitstream)
    temporal_surfaces[planeIdx].tiles[tileIdx].size = multibyte
    temporal_surfaces[planeIdx].tiles[tileIdx].data_prefix_coding =
            bytestream_current(bitstream)
    bytestream_seek(bitstream, temporal_surfaces[planeIdx].
            Tiles[tileIdx].size)
}
```

8.3.4.2

In numbered item 2), replace "If nTbSs is equal to 4" *with* "If nTbS is equal to 2".

In numbered item 2), replace "TransformCoeffQ(1)(1)" *with* " TransformCoeffQ(0)(1)"

8.3.5.2

Replace "(xTb0 >> nTbs, yTb0 >> nTbs)" *with* "(xTb0 / nTbS, yTb0 / nTbS)"

Replace "((xTb0 % 32) * 32, (yTb0 % 32) * 32)" *with:*

((xTb0 / 32) * (32 / nTbS), (yTb0 / 32) * (32 / nTbS))

8.4.1

Replace "(xTbP >> nTbS, yTbP >> nTbS)," *with* "(xTbP / nTbS, yTbP / nTbS),"

Replace "((xTbP % 32) * 32, (yTbP % 32) * 32)" *with:*

((xTbP / 32) * (32 / nTbS), (yTbP / 32) * (32 / nTbS))

Replace

If variable temporal_tile_intra_signalling is equal to 1 and xTbP >> 5 is equal to 0 and yTbP >> 5 is equal to 0 and TileTempSig is equal to 1

with

If variable temporal_tile_intra_signalling is equal to 1 and xTbP % 32 is equal to 0 and yTbP % 32 is equal to 0 and TileTempSig is equal to 1

Replace

at the position (xTb0 >> nTbs, yTb0 >> nTbs); and if in addition temporal_tile_intra_signalling_ enabled_ flag is set to 1, a variable TileTempSig corresponding to the value in TempSigSurface (9.3.4) at the position ((xTb0 % 32) * 32, (yTb0 % 32) * 32)

with

at the position (xTbP >> nTbs, yTbP >> nTbs); and if in addition temporal_tile_intra_signalling_ enabled_flag is set to 1, a variable TileTempSig corresponding to the value in TempSigSurface (9.3.4) at the position ((xTbP % 32) * 32, (yTbP % 32) * 32)

8.4.2

Replace " tempPredL2Residuals " *with* " temporalBuffer "

8.5.2 and 8.5.3

Replace " qm[x + (levelIdxSwap * nTbS)][y]" *with* " qm[x][y]"

8.5.3

Delete the following:

— Where levelIdxSwap is derived as follows:

```
if (levelIdx == 2)
  levelIdxSwap = 0
else
  levelIdxSwap = 1
```

8.6.1.1

Add the following sentence to the end of the subclause

The output of the upscaling processes shall be in the same value range as the input values.

8.7.5

Replace

```
modifier = recLowerResSamples[xSrc][ySrc] – (recUpsampledSamples[xDst][yDst] +
                recUpsampledSamples[xDst + 1][yDst]) >> 1
```

with

```
modifier = recLowerResSamples[xSrc][ySrc] – (recUpsampledSamples[xDst][yDst] +
                recUpsampledSamples[xDst + 1][yDst] + 1) >> 1
```

Replace

```
modifier = recLowerResSamples[xSrc][ySrc] – (recUpsampledSamples[xDst][yDst] +
          recUpsampledSamples[xDst + 1][yDst] +  recUpsampledSamples[xDst][yDst + 1] +
          recUpsampledSamples[xDst + 1][yDst + 1]) >> 2
```

with

```
modifier = recLowerResSamples[xSrc][ySrc] – (recUpsampledSamples[xDst][yDst] +
          recUpsampledSamples[xDst + 1][yDst] +  recUpsampledSamples[xDst][yDst + 1] +
          recUpsampledSamples[xDst + 1][yDst + 1] + 2) >> 2
```

Replace

```
recModifiedUpsampledSamples[xDst][yDst + 1] = recUpsampledSamples[xDst + 1][yDst] +
                modifier
recModifiedUpsampledSamples[xDst + 1][yDst] = recUpsampledSamples[xDst][yDst + 1] +
                modifier
```

with

```
recModifiedUpsampledSamples[xDst][yDst + 1] = recUpsampledSamples[xDst][yDst + 1] +
                modifier
recModifiedUpsampledSamples[xDst + 1][yDst] = recUpsampledSamples[xDst + 1][yDst] +
                modifier
```

9.2.1

Replace Figure 17 with the following figure:

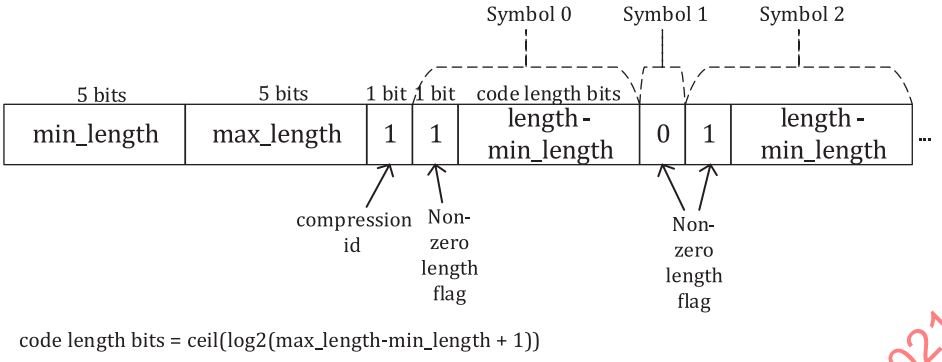

Figure 17 — Prefix coding decoder stream header for more than 31 non-zeros codes

Replace Figure 18 with the following figure:

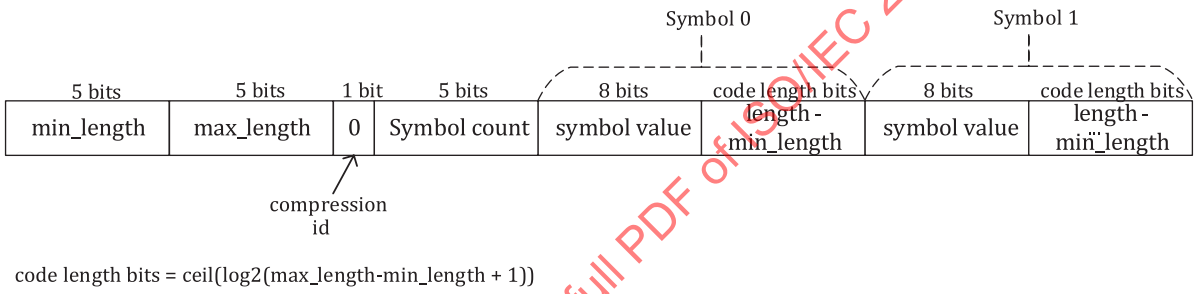

Figure 18 — Prefix coding decoder normal case

9.2.2

Replace

To find a Prefix Coding Code for a given set of symbols a Prefix Coding tree needs to be created using the following steps:

with

The Prefix Coding Code for a given set of symbols is generated at the encoder by building a Prefix Coding tree. The following steps provide an example of such generation:

Replace Figure 25 with the following:

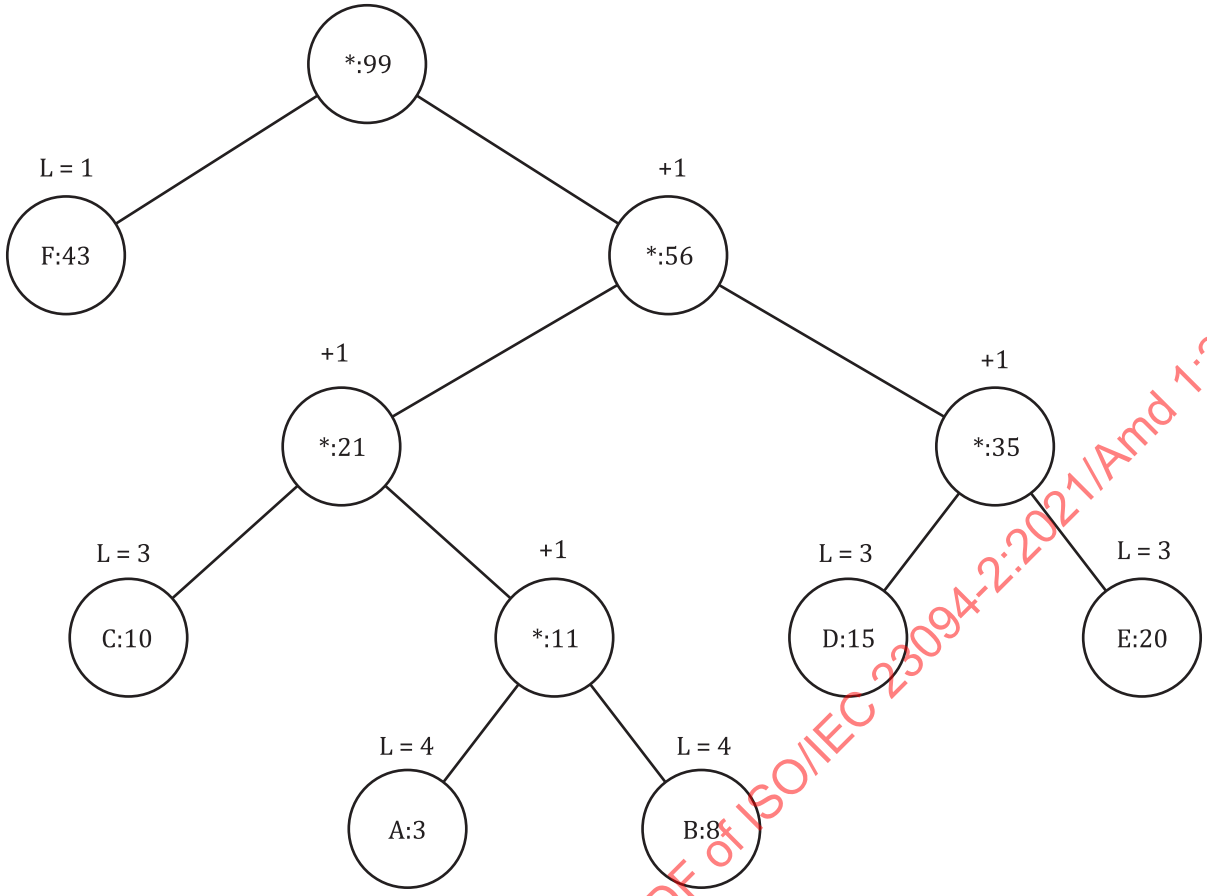

Figure 25 — Loop process completion

Replace list items 5) to 7) with the following list items:

5) Once the tree is built, the Prefix Coding length for every symbol is computed by traversing the tree from the root to each symbol, appending one bit each time a left branch or a right branch is taken.

6) Starting with a codeword of all zeroes, from the highest to the lowest codeword length, and from the lowest to the highest symbol values, a codeword is assigned, increasing the codeword binary value by 1, and shifting right when going from a higher to a lower codeword length. In the example above, this gives results to the following code in Table 43.

7) The decoder, after receiving the symbols and code lengths as described in subclause 9.2.1, repeats the process of step 6 to derive the codewords. Each codeword related to each symbol is then passed to the process 9.3.

9.3.4.1

Replace Figure 33 with the following figure: