International Standard

ISO/IEC 18181-1

Information technology — JPEG XL image coding system —

Part 1: Core coding system

Technologies de l'information — Système de codage d'images JPEG XL —

Partie 1: Système de codage de noyau

Second edition
2024-07

Reference number
ISO/IEC 18181-1:2024(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

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).

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 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.

This second edition cancels and replaces the first edition (ISO/IEC 18181-1:2022), which has been technically and editorially revised. It also incorporates the Amendment ISO/IEC 18181-1:2022/Amd 1:2022.

The main changes are as follows:

— technical corrections and clarifications, in particular to correct the values of various constants, correct errors in pseudocode, and clarify ambiguities in order to remove discrepancies between this document and ISO/IEC 18181-3 and ISO/IEC 18181-4;

— a thorough update of the document structure in order to improve clarity of presentation and to obtain a more logical ordering of the material from the point of view of decoder implementation.

A list of all parts in the ISO/IEC 18181 series 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 — JPEG XL image coding system —

Part 1:
Core coding system

1 Scope

This document specifies a set of compression methods for coding one or more images of bi-level, continuous-tone greyscale, or continuous-tone colour, or multichannel digital samples.

This document:

— specifies decoding processes for converting compressed image data to reconstructed image data;

— specifies a codestream syntax containing information for interpreting the compressed image data;

— provides guidance on encoding processes for converting source image data to compressed image data.

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 15076-1, *Image technology colour management — Architecture, profile format and data structure — Part 2: Based on ICC.1:2022*

ISO/IEC 60559, *Information technology — Microprocessor Systems — Floating-Point arithmetic*

IEC 61966-2-1, *Multimedia systems and equipment — Colour measurement and management — Part 2-1: Colour management — Default RGB colour space — sRGB*

ITU-R BT.2100-2, *Image parameter values for high dynamic range television for use in production and international programme exchange*

ITU-R BT.709-6, *Parameter values for the HDTV standards for production and international programme exchange*

IETF RFC 7932:2016, *Brotli Compressed Data Format*

SMPTE ST 428-1, *D-Cinema Distribution Master — Image Characteristics*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology 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

3.1.1
bitstream
sequence of bytes

3.1.2
codestream
bitstream representing compressed image data

3.1.3
bundle
structured data consisting of one or more fields

3.1.4
field
numerical value or bundle, or an array of either

3.1.5
histogram
array of unsigned integers representing a probability distribution, used for entropy coding

3.2 Inputs

3.2.1
pixel
vector of dimension corresponding to the number of channels, consisting of samples

3.2.2
sample
integer or real value, of which there is one per channel per pixel

3.2.3
grid
2-dimensional array

3.2.4
sample grid

common coordinate system for all samples of an image, with top-left coordinates (0, 0), the first coordinate increasing towards the right, and the second increasing towards the bottom

3.2.5
channel component
rectangular array of samples having the same designation, regularly aligned along a sample grid

3.2.6
rectangle
rectangular area within a channel or grid

3.2.7
width
width in samples (number of columns) of a sample grid or a rectangle

3.2.8
height
height in samples (number of rows) of a sample grid or a rectangle

3.2.9
frame
single image (possibly part of an animation, composite, or multi-page image), i.e. a 2-dimensional array of pixels

3.2.10
greyscale
image representation in which each pixel is defined by a single sample representing intensity (either luminance or luma depending on the ICC profile)

3.2.11
opsin
photosensitive pigments in the human retina, having dynamics approximated by the XYB colour space

3.2.12
animation
series of pictures and timing delays to display as a video medium

3.2.13
tick
unit of time such that animation frame durations are integer multiples of the tick duration

3.2.14
composite
series of images that are superimposed

3.2.15
multi-page image
sequence of pictures to display in a paged way

3.2.16
preview
lower-fidelity rendition of one of the frames (e.g., lower resolution), or a frame that represents the entire content of all frames

3.3 Processes

3.3.1
decoding process
process which takes as its input a codestream and outputs an image

3.3.2
decoder
embodiment of a decoding process

3.3.3
encoding process
process which takes as its input image(s) and outputs compressed image data in the form of a codestream

3.3.4
encoder
embodiment of an encoding process

3.3.5
lossless
descriptive term for encoding and decoding processes in which the output of a decoding procedure is identical to the input to the encoding procedure

3.3.6
lossy
descriptive term for encoding and decoding processes which are not lossless

3.3.7
upsampling
procedure by which the (nominal) spatial resolution of a channel is increased

3.3.8
downsampling
procedure by which the spatial resolution of a channel is reduced

3.3.9
entropy encoding
lossless procedure designed to convert a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols

3.3.10
entropy encoder
embodiment of an entropy encoding procedure

3.3.11
entropy decoding
lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder

3.3.12
entropy decoder
embodiment of an entropy decoding procedure

3.3.13
channel decorrelation
method of reducing total encoded entropy by removing correlations between channels

3.3.14
channel correlation factor
factor by which a channel should be multiplied by before adding it to another channel to undo the channel decorrelation process

3.3.15
VarDCT
lossy encoding of a frame that applies DCT to varblocks

3.3.16
quantization
method of reducing the precision of (DCT) coefficients

3.3.17
quantization weight
factor that a quantized coefficient is multiplied by prior to application of the inverse DCT in the decoding process

3.4 Image and codestream organization

3.4.1
raster order
access pattern from left to right in the top row, then in the row below and so on

3.4.2
naturally aligned
positioning of a power-of-two sized rectangle such that its top and left coordinates are divisible by its width and height, respectively

3.4.3
block
naturally aligned square rectangle covering up to 8 × 8 input pixels

3.4.4
varblock
variable-size rectangle of one or more blocks

3.4.5
group
naturally aligned rectangle covering up to $2^n \times 2^n$ input pixels, with n between 7 and 10, inclusive

3.4.6
table of contents
data structure that enables seeking to a group or the next frame within a codestream

3.4.7
section
part of the codestream with an offset and length that are stored in a frame's table of contents

3.4.8
coefficient
input value to the inverse DCT

3.4.9
pass
data enabling decoding of successively more detail

3.4.10
LF group
LF values from a naturally aligned rectangle covering up to $2^{n+3} \times 2^{n+3}$ input pixels

3.5 Abbreviated terms

DCT	discrete cosine transform (as specified in I.7)
HF	all DCT coefficients apart from the LF coefficients, i.e. the high frequency coefficients
IDCT	inverse discrete cosine transform (as specified in I.7)
LF	lowest frequency coefficients of DCT coefficients, i.e. block averages
RGB	additive colour model with red, green, blue channels
XYB	absolute colour space based on gamma-corrected LMS (a colour space representing the response of cone cells in the human eye), in which X is derived from the difference between L and M (red-green), Y is an average of L and M (behaves similarly to luminance), and B is derived from S (blue)

4 Conventions

4.1 Mathematical symbols

[a, b], (c, d), [e, f)	closed or open or half-open intervals containing all integers or real numbers x (depending on context) such that $a \leq x \leq b$, $c < x < d$, $e \leq x < f$
{a, b, c}	ordered sequence of elements
`pi`	the smallest positive zero of the sine function (π)

4.2 Functions

`pow(x,y)`	exponentiation, `x` to the power of `y`.
`sqrt(x)`	square root, such that `pow(sqrt(x),2) == x` and `sqrt(x) >= 0`. Undefined for `x < 0`.
`cbrt(x)`	cube root, such that `pow(cbrt(x),3) == x`.
`cos(r)`	cosine of the angle `r` (in radians).
`erf(x)`	Gauss error function: `erf(x)` $= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$
`log(x)`	natural logarithm of `x`. Undefined for `x <= 0`.
`log2(x)`	base-two logarithm of `x`. Undefined for `x <= 0`.
`floor(x)`	the largest integer that is less than or equal to `x`.
`ceil(x)`	the smallest integer that is greater than or equal to `x`.
`abs(x)`	absolute value of `x`: equal to `-x` if `x < 0`, otherwise `x`.
`sign(x)`	sign of `x`, 0 if `x` is 0, +1 if `x` is positive, –1 if `x` is negative.
`len(a)`	length (number of elements) of array `a`.
`sum(a)`	sum of all elements of the array/tuple/sequence `a`.
`max(a)`	maximal element of the array/tuple/sequence `a`.
`min(a)`	smallest element of the array/tuple/sequence `a`.
`UnpackSigned(u)`	equivalent to `u/2` if `u` is even, and `-(u + 1)/2` if `u` is odd.
`clamp(x, lo, hi)`	equivalent to `min(max(lo, x), hi)`.
`InterpretAsF16(u)`	the real number resulting from interpreting the unsigned 16-bit integer `u` as a binary16 floating-point number representation (cf. ISO/IEC 60559)
`InterpretAsF32(u)`	the real number resulting from interpreting the unsigned 32-bit integer `u` as a binary32 floating-point number representation (cf. ISO/IEC 60559)
`NarrowToI32(x)`	the signed 32-bit integer corresponding to the 32 least significant bits of the two's complement representation of `x`
`BitSet(u,b)`	the bit corresponding to `b` is 1 in `u`: true if and only if `u & b` (bitwise AND)

4.3 Operators

This document uses the operators defined by the C++ programming language (ISO/IEC 14882), with the following differences:

`/`	division of real numbers without truncation or rounding. Division by zero is undefined.
`<<`	left shift: `x << s` is defined as `x * pow(2,s)`
`>>`	right shift: `x >> s` is defined as `floor(x / pow(2,s))`
`Umod`	remainder: `a Umod d` is the unique integer `r` in `[0, d)` for which `a == r + q * d` for a suitable integer `q`

`Idiv` integer division: `a Idiv b` is equivalent to `a / b`, rounded towards zero to an integer value

`and` logical AND (`&&` in C++), true if and only if both operands are true, with short-circuit evaluation

`or` logical OR (`||` in C++), false if and only if both operands are false, with short-circuit evaluation

The order of precedence for these operators is listed in Table 1 in descending order. If several operators appear in the same line, they have equal precedence. When several operators of equal precedence appear at the same level in an expression, evaluation proceeds according to the associativity of the operator (either from right to left or from left to right).

Table 1 — Operator precedence

Operators	Type of operation	Associativity
`++x, - -x`	prefix increment/decrement	left to right
`x++, x- -`	postfix increment/decrement	left to right
`!, ~`	logical/bitwise NOT	right to left
`*, /, Idiv, Umod`	multiplication, division, integer division, remainder	left to right
`+, -`	addition and subtraction	left to right
`<<, >>`	left shift and right shift	left to right
`< , >, <=, >=, ==, !=`	relational	left to right
`&`	bitwise AND	left to right
`^`	bitwise XOR	left to right
`|`	bitwise OR	left to right
`and`	logical AND	left to right
`or`	logical OR	left to right
`=`	assignment	right to left
`+=, -=, *=`	compound assignment	right to left

4.4 Pseudocode

This document describes functionality using pseudocode formatted as follows:

```
// Informative comment
var = u(8);
if (var == 1) return;  // Stop executing this code snippet
/* Normative specification: var != 0 */
(out1, out2) = Function(var, kConstant);
```

Variables such as `var` are typically referenced by text outside the source code.

The semantics of this pseudocode are those of the C++ programming language (ISO/IEC 14882), except that:

— Symbols from 4.1 and functions from 4.2 are allowed;

— Division, remainder and exponentiation are expressed as specified in 4.3;

— Functions can return tuples which unpack to variables as in the above example;

— `/* */` enclose normative directives specified using prose;

— All integers are stored using two's complement;

— Expressions and variables of which types are omitted, are understood as real numbers.

Where unsigned integer wraparound and truncated division are required, `Umod` and `Idiv` (see 4.3) are used for those purposes.

5 Functional concepts

5.1 Image organization

A channel is defined as a rectangular array of (integer or real) samples regularly aligned along a sample grid of `width` sample positions horizontally and `height` sample positions vertically. The number of channels may be 1 to 4099 (see `num_extra` in D.3).

A pixel is defined as a vector of dimension corresponding to the number of channels, consisting of samples with a position matching that of the pixel.

An image is defined as a two-dimensional array of pixels, and its width is `width` and height is `height`. Unless otherwise mentioned, `raster order' is used: left to right column within the topmost row, then left to right column within the row below the top, and so on until the rightmost column of the bottom row.

A codestream may contain multiple image frames. These can constitute an animation (a sequence of images) or a composite still image. Frames can have dimensions that differ from the image dimensions, and they can be blended over preceding frames. The frame that is being decoded is referred to as the `current' frame.

5.2 Mirroring

Some operations access samples with coordinates `cx`, `cy` that are outside the current frame bounds. The decoder redirects such accesses to a valid sample at the coordinates `Mirror(cx, cy)` as defined in the following code:

```
Mirror1D(coord, size) {
  if (coord < 0) return Mirror1D(-coord - 1, size);
  else if (coord >= size) return Mirror1D(2 * size - 1 - coord, size);
  else return coord;
}

Mirror(x, y) {
  return (Mirror1D(x, width), Mirror1D(y, height));
}
```


5.3 Group splitting

Channels are logically partitioned into naturally-aligned groups of `group_dim` × `group_dim` samples. The effective dimension of a group (i.e. how many pixels to read) can be smaller than `group_dim` for groups on the right or bottom of the image. The decoder ensures the decoded image has the dimensions specified in SizeHeader (D.2) by cropping at the right and bottom as necessary. Unless otherwise specified, `group_dim` is 256.

LF groups likewise consist of `group_dim` × `group_dim` LF samples, with the possibility of a smaller effective size on the right and bottom of the image.

Groups can be decoded independently. A 'table of contents' (F.3) stores the size (in bytes) of each group to allow seeking to any group. An optional permutation allows arbitrary reordering of groups within the codestream.

5.4 Codestream organization

A bitstream is a sequence of bytes. A codestream is a bitstream that represent compressed image data and metadata. N bytes can also be viewed as 8 × N bits. The first 8 bits are the bits constituting the first byte, in least to most significant order, the next eight bits (again in least to most significant order) constitute the second byte, and so on. Unless otherwise specified, bits are read from the codestream as specified in B.2.1.

NOTE Ordering bits from least to most significant allows using special CPU instructions to isolate the least-significant bits.

6 Encoder requirements

An encoder is an embodiment of the encoding process. This document does not specify an encoding process, and any encoding process is acceptable as long as the codestream conforms to the codestream syntax specified in this document. Annex O provides an informative description of such an encoding process.

7 Decoder requirements

A decoder is an embodiment of the decoding process. The decoder reconstructs sample values (arranged on a rectangular sampling grid) from a codestream as specified in this document. Annexes A to N define an output that alternative implementations shall duplicate.

Annex A
(normative)

Codestream overview

A.1 Codestream structure

The codestream consists of image headers (Annexes D and N) and an ICC profile, if present (E.4), followed by one or more frames (Annex F), as shown in Table A.1. This table and those it references, together with the syntax description (B.1), specify how a decoder reads the headers and frame(s).

Table A.1 — Codestream structure

condition	type	name	Annex
	Headers	`headers`	Annex D, Annex N
`headers.metadata.colour_encoding.want_icc`		`icc`	E.4
`headers.metadata.have_preview`	Frame	`preview_frame`	Annex F
	Frame	`frames[0]`	Annex F
`for (i = 1; !frames[i - 1].frame_header.is_last; ++i)`	Frame	`frames[i]`	Annex F

The `preview_frame` is a self-contained frame whose FrameHeader (F.2) specifies `lf_level=0`, `frame_type=kRegularFrame` and `save_as_reference=0`.

A.2 Decoding process

Annex B specifies how the decoder parses header information.

Annex C specifies how the decoder reads entropy-coded data.

After reading the image header (Annexes D and E) and the frame header (Annex F), the decoder reads the frame data (Annex G). The reconstruction of the decoded image can be viewed as a pipeline with multiple stages.

Some Annexes or subclauses begin with a condition; the decoder only applies the processing steps described in such an Annex or subclause if its condition holds true.

Annex H specifies the **kModular** frame encoding and Modular sub-bitstreams.

Annex I specifies the **kVarDCT** frame encoding.

The decoder applies restoration filters as specified in Annex J.

The presence/absence of additional image features (patches, splines and noise) is indicated in the frame header. The decoder draws these as specified in Annex K. Image features (if present) are rendered after restoration filters (if enabled), in the listed order.

Finally, the decoder performs colour transforms as specified in Annex L.

NOTE For an introduction to the coding tools and additional background, please refer to Reference [3].