FINAL
DRAFT

INTERNATIONAL
STANDARD

**ISO/IEC
FDIS
14496-31**

ISO/IEC JTC **1**/SC **29**

Secretariat: JISC

Voting begins on:
**2017-07-27**

Voting terminates on:
**2017-09-21**

# Information technology — Coding of audio-visual objects —

## Part 31:
## Video coding for browsers

*Technologies de l'information — Codage des objets audiovisuels —*

*Partie 31: Titre manque*





Reference number
ISO/IEC FDIS 14496-31:2017(E)



ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

# 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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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).

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).

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

For an explanation on 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 the following URL: www.iso.org/iso/foreword.html.



This document was prepared by 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 14496 series can be found on the ISO website.

# Information technology — Coding of audio-visual objects —

# Part 31: Video coding for browsers

## 1 Scope

This document specifies the video coding for browsers (VCB) syntax format and decoding process.

## 2 Normative references

There are no normative references in this document.

## 3 Terms and definitions

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



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

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

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



**3.1**
**4:2:0 colour format**
colour format with two chroma arrays that each have half the height and half the width of the luma array

**3.2**
**AC transform coefficient**
*transform coefficient* (3.55) for which the *frequency index* (3.23) in at least one of the two dimensions is non-zero

**3.3**
**bitstream**
sequence of bits that forms the representation of *coded frames* (3.11)

**3.4**
**bit string**
ordered string with a limited number of bits, in which the left-most bit is the most significant bit (MSB) and the right-most bit is the least significant bit (LSB)

**3.5**
**block**
MxN (M-column by N-row) array of *sample values* (3.22) or MxN array of *transform coefficients* (3.55)

**3.6**
**Boolean decoder**
arithmetic decoder that processes one Boolean element at a time, wherein each Boolean element indicates how to refine the segmentation of a number line and hence specifies the decoded bits

**3.7**
**Boolean decoding**
process of decoding a *syntax element* (3.54) using a *Boolean decoder* (3.6)

**3.8**
**byte**
8-bit *bit string* (3.4)

**3.9**
**byte aligned**
position that is an integer multiple of eight, such that the first bit in the *bitstream* (3.3) has position 0 and the position of other bits is stated relative to this position

**3.10**
**chroma**
sample array or single *sample* (3.51), denoted by the symbols Cb and Cr, representing one of the two colour difference signals related to the primary colours

**3.11**
**coded frame**
set of *syntax elements* (3.54) that represents a *frame* (3.22) in the *bitstream* (3.3)

**3.12**
**component**
array or single *sample* (3.51) from one of the three arrays [*luma* (3.32) and two *chroma* (3.10)] that compose a picture in *4:2:0 colour format* (3.1)



**3.13**
**control partition**
partition containing the frame header and all of the macroblock headers for a *coded frame* (3.11)



**3.14**
**DC transform coefficient**
*transform coefficient* (3.55) for which the *frequency index* (3.23) in both of the two dimensions is zero

**3.15**
**decoded frame**
*frame* (3.22) reconstructed from a *coded frame* (3.11) by a *decoder* (3.16)

**3.16**
**decoder**
embodiment of the *decoding process* (3.17)

**3.17**
**decoding process**
process that derives *decoded frames* (3.15) from *syntax elements* (3.54)

**3.18**
**dequantization**
process in which *transform coefficients* (3.55) are obtained by scaling the *quantized transform coefficients* (3.45)

**3.19**
**encoder**
embodiment of an *encoding process* (3.20)

**3.20**
**encoding process**
process that generates a *bitstream* (3.3)

Note 1 to entry: Other than establishing criteria for conformance of bitstreams produced by an encoder, the encoding process is not specified in this document (although an example encoding process is informatively described in Annex B).

**3.21**
**flag**
binary variable

**3.22**
**frame**
array of luma samples (Y) and two corresponding arrays of chroma samples (Cb and Cr) in *4:2:0 colour format* (3.1)

**3.23**
**frequency index**
one-dimensional or two-dimensional index associated with a *transform coefficient* (3.55) prior to an *inverse transform* (3.29) part of the *decoding process* (3.17)

**3.24**
**inter coding**
*macroblock* (3.33) or *frame* (3.22) coded using *inter prediction* (3.25)



**3.25**
**inter prediction**
process of deriving the *prediction* (3.37) for the current *frame* (3.22) using previously *decoded frames* (3.15)



**3.26**
**intra coding**
*macroblock* (3.33) or *frame* (3.22) coded using *intra prediction* (3.28)

**3.27**
**I frame**
*frame* (3.22) that may be decoded using only *intra prediction* (3.28)

**3.28**
**intra prediction**
process of deriving the *prediction value* (3.42) for the current *sample* (3.51) using previously decoded *sample values* (3.52) in the same *decoded frame* (3.15)

**3.29**
**inverse transform**
part of the *decoding process* (3.17) by which a set of *transform coefficients* (3.55) are converted into spatial-domain values

**3.30**
**key frame**
*frame* (3.22) where the *decoding process* (3.17) is reset, such that a key frame and all following frames are always decodable without access to preceding frames and such that a frame is a key frame if and only if it is an *I frame* (3.27)

**3.31**
**level**
specified set of constraints on the permissible values for the *syntax elements* (3.54) and other characteristics of the *bitstreams* (3.3) produced by conforming *encoders* (3.19) established in combination with defined *profiles* (3.43) for ensuring interoperability of the bitsreams produced by conforming encoders with the *decoding process* (3.17) specified by conforming *decoders* (3.16)

**3.32**
**luma**
sample array or *single sample* (3.51), represented by the symbol Y, that ordinarily represents the brightness signal related to the primary colours

**3.33**
**macroblock**
16x16 luma sample value *block* (3.5) and its two corresponding 8x8 chroma sample value blocks

**3.34**
**motion vector**
two-dimensional vector used for *inter prediction* (3.25) that provides an offset from the coordinates in the *decoded frame* (3.15) to the coordinates in a *reference frame* (3.49)

**3.35**
**parse**
process of extracting a *syntax element* (3.54) from a *bitstream* (3.3)

**3.36**
**partitioning**
process of dividing a set into subsets such that each element in the set belongs to only one subset

**3.37**
**prediction**
prediction process consisting of either *inter* (3.25) or *intra prediction* (3.28)



**3.38**
**prediction frame**
**P frame**
*frame* (3.22) decoded by referencing another frame



**3.40**
**prediction process**
process of estimating the decoded *sample value* (3.52) or data element using a predictor

**3.41**
**prediction residual**
difference between the reconstructed *samples* (3.51) and the corresponding *prediction values* (3.42)

**3.42**
**prediction value**
combination of the previously decoded *sample values* (3.52) or data elements, used in the *prediction process* (3.40) of the next sample value or data element

**3.43**
**profile**
specified subset of syntax, semantics and processes established in combination with defined *levels* (3.31) for ensuring interoperability of the *bitstreams* (3.3) produced by conforming *encoders* (3.19) with the *decoding process* (3.17) specified for conforming *decoders* (3.16)

**3.44**
**quantization parameter**
variable used for scaling the *quantized coefficients* (3.45) in the *decoding process* (3.17)

**3.45**
**quantized coefficient**
*transform coefficient* (3.55) before *dequantization* (3.18)

**3.46**
**random access point**
point, other than the beginning of the *bitstream* (3.3), for starting the *decoding process* (3.17) for the remainder of a bitstream

**3.47**
**raster scan**
mapping of a two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc. rows of the pattern (going down) each scanned from left to right

**3.48**
**reconstruction**
addition of the decoded *prediction residual* (3.41) and the corresponding *prediction values* (3.42)

**3.49**
**reference frame**
previously *decoded frame* (3.15) used during *inter prediction* (3.25)

**3.50**
**reserved**
*syntax element* (3.54) value which may be used to extend this specification in the future

Note 1 to entry: Such values are not allowed to be present in bitstreams conforming to this document.



**3.51**
**sample**
basic element of the array of the *decoded frame* (3.15)



**3.52**
**sample value**
value of a *sample* (3.51), which is an integer in the range of 0 to 255, inclusive

**3.53**
**sub-block**
4x4 *block* (3.5)

**3.54**
**syntax element**
element of data represented in the *bitstream* (3.3)

**3.55**
**transform coefficient**
scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* (3.23) in an *inverse transform* (3.29) part of the *decoding process* (3.17)

## 4 Abbreviated terms

| | |
|---|---|
| DCT | Discrete cosine transform |
| LSB | Least significant bit |
| MB | Macroblock |
| MSB | Most significant bit |

SB Sub-block

VCB Video coding for browsers

WHT Walsh hadamard transform

# 5 Conventions

## 5.1 General

The mathematical operators and their precedence rules used in this document are similar to those used in the C programming language. However, the operation of integer division with truncation is specifically defined.

In addition, an array with two elements used to hold a motion vector (such as Mv) can be accessed using either normal array notation (e.g. Mv[ 0 ] and Mv[ 1 ]), or by name (i.e. Mv). The only operations defined when using the name are assignment and equality testing. The assignment of an array is represented using the normal notation A = B and is specified to mean the same as doing both the individual assignments A[ 0 ] = B[ 0 ] and A[ 1 ] = B[ 1 ]. Equality testing of two motion vectors is represented using the notation A == B and is specified to mean the same as (A[ 0 ] == B[ 0 ] && A[ 1 ] == B[ 1 ]).

## 5.2 Arithmetic operators

+ Addition

− Subtraction (as a binary operator) or negation (as a unary prefix operator)

* Multiplication

/ Integer division with truncation of the result toward zero. For example, 7/4 and (−7)/(−4) are truncated to 1, and −7/4 and 7/(−4) are truncated to −1.

÷ Used to denote division in mathematical equations where no truncation or rounding is intended.

a % b Remainder of a divided by b, defined only for a >= 0 and b > 0.



## 5.3 Logical operators

a && b Logical AND operation between a and b

a || b Logical OR operation between a and b

! Logical NOT operation

## 5.4 Relational operators

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

== Equal to

!= Not equal to

 

## 5.5 Bitwise operators

& AND operation

| OR operation

~ Negation operation

a >> b Shift a in 2's complement binary integer representation format to the right by b bit positions. This operator is only used with b being a non-negative integer. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of a prior to the shift operation.

a << b Shift a in 2's complement binary integer representation format to the left by b bit positions. This operator is only used with b being a non-negative integer. Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

## 5.6 Assignment operators

= Assignment operator

++ Increment, x++ is equivalent to x = x + 1. When this operator is used for an array index, the variable value is obtained before the increment operation.

−− Decrement, i.e. x−− is equivalent to x = x − 1. When this operator is used for an array index, the variable value is obtained before the decrement operation.



+= Addition assignment operator, for example:

x += 3 corresponds to x = x + 3, and,

x += (−3) is equivalent to x = x + (−3).



−= Subtraction assignment operator, for example:

x −= 3 corresponds to x = x − 3, and,

x −= (−3) is equivalent to x = x − (−3).

## 5.7 Range notation

x = y..z x takes on integer values starting from y to z, inclusive, with x, y and z being integer numbers and z being greater than y.

## 5.8 Mathematical functions

The following mathematical functions are defined as follows:

$$\mathrm{Abs}(x) = \begin{cases} x; & x >= 0 \\ -x; & x < 0 \end{cases}$$

$$\mathrm{Clip1}(x) = \mathrm{Clip3}(0, 255, x)$$

$$\mathrm{Clip3}(x, y, z) = \begin{cases} x; & z < x \\ y; & z > y \\ z; & \text{otherwise} \end{cases}$$

$$\text{Min}(x, y) = \begin{cases} x; & x <= y \\ y; & x > y \end{cases}$$

$$\text{Max}(x, y) = \begin{cases} x; & x >= y \\ y; & x < y \end{cases}$$

## 5.9 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by the use of parentheses, the following rules apply.

— Operations of a higher precedence are evaluated before any operation of a lower precedence.

— Operations of the same precedence are evaluated sequentially from left to right.

Table 1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE For those operators that are also used in the C programming language, the order of precedence used in this document is the same as used in the C programming language.

**Table 1 — Operation precedence from highest (at top of the table) to lowest (at the bottom of the table)**

| operations (with operands x, y, and z) |
|---|
| "x++", "x−−" |
| "!x", "−x" (as a unary prefix operator) |
| $x^y$ |
| "x * y", "x / y", "x ÷ y", $\frac{x}{y}$, "x % y" |
| "x + y", "x − y" (as a two argument operator), $\sum_{i=x}^{y} f(i)$ |
| "x << y", "x >> y" |
| "x < y", "x <= y", "x > y", "x >= y" |
| "x == y", "x != y" |
| "x & y" |
| "x \| y" |
| "x \|\| y" |
| "x ? y : z" |
| "x..y" |
| "x = y", "x += y", "x −= y" |



## 5.10 Method of describing bitstream syntax

The bitstream syntax is specified in Clause 6 in a style similar to the C programming language. A syntax element is completely described by its name together with either one or two descriptors for its method of coded representation. Syntax element names consist of descriptive groups of lowercase letters separated by an underscore character. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements.

 

A syntax element is represented in **bold** type in two places; where it is read from the bitstream (in Clause 6) and where it is defined (in Clause 7). Otherwise, syntax elements are represented in non-bold type throughout.

In some cases, the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lowercase and uppercase letters without any underscore characters. Variables starting with an uppercase letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an uppercase letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lowercase letter are only used within the subclause from which they are derived.

The association of values and names is specified in the text. In some cases, "mnemonic" names for syntax elements or variables are used interchangeably with their numerical values. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an uppercase letter and may contain more uppercase letters.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used when the number of bits is an integer multiple of 4. For example, "0x1a" represents a bit-string "0001 1010".

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value other than zero.

Table 2 lists examples of the syntax specification format. When **syntax_element** appears (with bold face font), it specifies that a syntax element is parsed from a bitstream.



**Table 2 — Examples of syntax specification formats**



| syntax | type |
|---|---|
| /* A statement can be a syntax element with an associated descriptor or can be an expression used to specify its existence, type and value, as in the following examples */ | |
| **syntax_element** | ue(v) |
| conditioning statement | |
| | |
| /* A group of statements enclosed in brackets is a compound statement and is treated functionally as a single statement. */ | |
| { | |
| statement | |
| statement | |
| ... | |
| } | |
| | |
| /* A "while" structure specifies that the statement is to be evaluated repeatedly while the condition remains true. */ | |
| while (condition) | |
| statement | |
| | |
| /* A "do ... while" structure executes the statement once and then tests the condition. It repeatedly evaluates the statement while the condition remains true. */ | |
| do | |
| statement | |
| while (condition) | |
| | |