INTERNATIONAL STANDARD

ISO/IEC 14496-2

Second edition
2001-12-01

AMENDMENT 3
2003-05-15

Information technology — Coding of audio-visual objects —

Part 2:
Visual

AMENDMENT 3: New levels and tools for MPEG-4 visual

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

AMENDEMENT 3: Nouveaux niveaux et outils pour codage visuel MPEG-4

Reference number
ISO/IEC 14496-2:2001/Amd.3:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web 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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this Amendment may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 3 to International Standard ISO/IEC 14496-2:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

Introduction

With the advent of new multimedia streaming applications, there are requests from industry parties to add new levels to existing visual profiles in order to establish interoperability points for these applications. Adding alpha channel support to VTC is also included in this amendment.

Information technology — Coding of audio-visual objects —

Part 2:
Visual

AMENDMENT 3: New levels and tools for MPEG-4 visual

Add the following row to the Table N-1 in Annex N:

Simple Scalable	L0	QCIF	1	1 x Simple or Simple Scalable	1	446	124	1856	N.A.	10	10	2048	N. A.	N. A.	128	1 spatial or temporal enhancement layer

and the following text after item 10 (the note for Simple Profile @ Level L0) in the notes of Annex N:

11. For Simple Scalable Profile @ Level L0, the following restrictions apply:

a) Base layer compliant to Simple Profile @ Level L0;

b) The enhancement layer maximum f_code shall be 1;

c) The enhancement layer intra_dc_vlc_threshold shall be 0;

d) For enhancement layer, if AC prediction is used, the following restriction applies: QP value shall not be changed within a VOP (or within a video packet if video packets are used in a VOP). If AC prediction is not used, there are no restrictions to changing QP value;

e) Spatial scalability shall be limited to a 1:1 scaling ratio.

Add the following row to the Table AMD2-15:

AS	L3b	352x288	4	4x AS or Simple	1	1188	396	11880	100	65	65	4096	1500	N.A.

Add the following two rows in Table G-1 in AMD2:

Simple Scalable Profile/Level 0	00010000
Advanced Simple Profile/Level 3b	11110111

Change the following two rows in Table G-1 in AMD2 from:

Reserved	00001001- 00010000
Reserved	11110110 - 11110111

to

Reserved	00001001- 00001111
Reserved	11110110

Addition of alpha channel to VTC:

In subclause 6.3.8, change
from

Table 6-33: Texture Object Layer Shape type.

texture_object_layer_shape	Meaning
00	rectangular
01	binary
10	reserved
11	reserved

to

Table 6-33: Texture Object Layer Shape type.

texture_object_layer_shape	Meaning
00	rectangular
01	binary
10	reserved
11	grayscale

In subclause 6.2.8 - Still Texture Object, change

from

Else if (texture_object_layer_shape == "01") {		
Horizontal_ref	15	uimsbf
Marker_bit	1	bslbf
Vertical_ref	15	uimsbf
Marker_bit	1	bslbf
Object_width	15	uimsbf
Marker_bit	1	bslbf
Object_height	15	uimsbf
Marker_bit	1	bslbf

to

Else if (texture_object_layer_shape == “01” or “11”) {		
horizontal_ref	15	uimsbf
marker_bit	1	bslbf
vertical_ref	15	uimsbf
marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf

from

else { /* !texture_error_resilience_disable */		
if (texture_object_layer_shape== “01” && tiling_disable==“0”) {		
if (texture_tile_type==“boundary tile”)		
shape_object_decoding()		
}		
for (color = “y”, “u”, “v”){		
wavelet_dc_decode()		
}		

to

else { /* !texture_error_resilience_disable */		
if (texture_object_layer_shape!= “00” && tiling_disable==“0”) {		
if (texture_tile_type==“boundary tile”)		
shape_object_decoding()		
}		
for (color = “y”, “u”, “v”,”α”){		
wavelet_dc_decode()		
}		

from

else if (quantisation_type == 3){		
for (color = “y”, “u”, “v”) {		
do{		
quant_byte	8	uimsbf
} while(quant_byte >>7)		
}		

to

else if (quantisation_type == 3){		
for (color = “y”, “u”, “v”,”α”) {		
do{		
quant_byte	8	Uimsbf
} while(quant_byte >>7)		
}		

In subclause 6.2.8 – DecodeStu, change

from

DecodeStu (){		
for (color = “y”, “u”, “v”)		
wavelet_dc_decode ()		

to

DecodeStu (){		
for (color = “y”, “u”, “v”,”α”)		
wavelet_dc_decode ()		

from

else if (quantisation_type == 3){		
for (color = “y”, “u”, “v”) {		
do{		
quant_byte	8	uimsbf
} while(quant_byte >>7)		
}		

to

else if (quantisation_type == 3){		
for (color = “y”, “u”, “v”, ”α”) {		
do{		
quant_byte	8	Uimsbf
} while(quant_byte >>7)		
}		

In subclause 6.2.8.1 – TextureLayerSQ, change

from

TextureLayerSQ() {	**No. of bits**	**Mnemonic**
if (scan_direction == 0) {		
for (“y”, “u”, “v”) {		

to

TextureLayerSQ() {	**No. of bits**	**Mnemonic**
if (scan_direction == 0) {		
for (“y”, “u”, “v”, ”α”) {		

from

for (i = 0; I<tree_blocks; i++) {		
for (color = “y”, “u”, “v”)		
arith_decode_highbands_td()		
}		

to

for (i = 0; I<tree_blocks; i++) {		
for (color = “y”, “u”, “v”, ”α”)		
arith_decode_highbands_td()		
}		

In subclause 6.2.8.3 – TextureSpatialLayerSQNSC, change

from

TextureSpatialLayerSQNSC() {	No. of bits	Mnemonic
for (color="y","u","v") {		
if ((first_wavelet_layer && color=="y") ‖ (second_wavelet_layer && color=="u","v")) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
}		
if (color =="y")		
max_bitplanes	5	uimsbf
else if (!first_wavelet_layer)		
max_bitplanes	5	uimsbf
}		
for (color="y","u","v")		
if (color="y" ‖ !first_wavelet_layer)		
arith_decode_highbands_bb()		
}		

to

TextureSpatialLayerSQNSC() {	No. of bits	Mnemonic
for (color="y","u","v","α") {		
if ((first_wavelet_layer && color=="y","α") ‖ (second_wavelet_layer && color=="u","v")) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
}		
if (color =="y","α")		
max_bitplanes	5	uimsbf
else if (!first_wavelet_layer)		
max_bitplanes	5	uimsbf
}		
for (color="y","u","v","α")		
if (color="y","α" ‖ !first_wavelet_layer)		
arith_decode_highbands_bb()		
}		

In subclause 6.2.8.7 – TextureSNRLayerMQNSC, change

from

TextureSNRLayerMQNSC(){	No. of bits	Mnemonic
if (spatial_scalability_levels == wavelet_decomposition_levels && spatial_layer_id == 0) {		
for (color = “y“) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
else {		
for (color=“y”, “u”, “v”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
if (scan_direction == 0) {		
for (i = 0; i<tree_blocks; i++) {		
for (color = “y”, “u”, “v”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_td()		
}		
}		
} else {		
for (i = 0; i< spatial_layers; i++) {		
for (color = “y”, “u”, “v”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_bb()		
}		
}		
}		
}		

to

TextureSNRLayerMQNSC(){	No. of bits	Mnemonic
if (spatial_scalability_levels == wavelet_decomposition_levels && spatial_layer_id == 0) {		
for (color = “y“,”α”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
else {		
for (color=“y”, “u”, “v”, ”α”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
if (scan_direction == 0) {		
for (i = 0; i<tree_blocks; i++) {		
for (color = “y”, “u”, “v”, ”α”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_td()		
}		
}		
} else {		
for (i = 0; i< spatial_layers; i++) {		
for (color = “y”, “u”, “v” ,”a”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_bb()		
}		
}		
}		
}		