

Information technology — Generic coding of moving pictures and associated audio information —

Part 7: Advanced Audio Coding (AAC)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage générique des images animées et du son associé —

Partie 7: Codage du son avancé (AAC)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to International Standard ISO/IEC 13818-7:1997 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

1) Add the following paragraph at the end of clause 5:

“

The number of bits for each data element is written in the second column. "X..Y" indicates that the number of bits is one of the values between X and Y including X and Y. "{X;Y}" means the number of bits is X or Y, depending on the value of other data elements in the bitstream.

”

2) Replace Table 6.13 in subclause 6.3 with the following:

“

Syntax	No. of bits	Mnemonic
<pre> section_data() { if(window_sequence == EIGHT_SHORT_SEQUENCE) sect_esc_val = (1<<3) - 1 else sect_esc_val = (1<<5) - 1 for(g=0; g < num_window_groups; g++) { k=0 i=0 while (k<max_sfb) { sect_cb[g][i] sect_len=0 while (sect_len_incr == sect_esc_val) sect_len += sect_esc_val sect_len += sect_len_incr sect_start[g][i] = k sect_end[g][i] = k+sect_len for (sfb=k; sfb<k+sect_len; sfb++) sfb_cb[g][sfb] = sect_cb[g][i]; k += sect_len i++ } num_sec[g] = i } } </pre>		
	4	uimsbf
	{3;5}	uimsbf

”

3) In Table 6.15 in subclause 6.3, replace:

No. of bits “4/6” with “{4;6}”

and

No. of bits “3/5” with “{3;5}”.

4) Replace Table 6.16 in subclause 6.3 with the following:

“

Syntax	No. of bits	Mnemonic
<pre> spectral_data() { for(g=0; g<num_window_groups; g++) { for (i=0; i<num_sec[g]; i++) { if (sect_cb[g][i] != ZERO_HCB && sect_cb[g][i] <= ESC_HCB) { for (k=sect_sfb_offset[g][sect_start[g][i]; k< sect_sfb_offset[g][sect_end[g][i]];) { if (sect_cb[g][i]<FIRST_PAIR_HCB) { hcod[sect_cb[g][i]][w][x][y][z] if(unsigned_cb[sect_cb[g][i]]) quad_sign_bits k += QUAD_LEN } else { hcod[sect_cb[g][i]][y][z] if(unsigned_cb[sect_cb[g][i]]) pair_sign_bits k += PAIR_LEN } } } } } } </pre>		
	1..16	bslbf
	0..4	bslbf
	1..15	bslbf
	0..2	bslbf

”

“

”

6) Replace Table 6.22 in subclause 6.3 with the following:

“

Syntax	No. of bits	Mnemonic
fill_element()		
{		
cnt = count	4	uimsbf
if (cnt == 15)		
cnt += esc_count - 1;	8	uimsbf
while (cnt > 0) {		
cnt -= extension_payload(cnt)		
}		
}		

”

and add the following tables, Table 6.24, Table 6.25 and Table 6.26, at the end of subclause 6.3:

“

Table 6.24 — Syntax of extension_payload()

extension_payload(cnt)		
{		
extension_type	4	uimsbf
switch(extension_type) {		
case EXT_DYNAMIC_RANGE:		
n = dynamic_range_info();		
return n;		
case EXT_FILL_DATA:		
fill_nibble /* must be '0000' */	4	uimsbf
for (i=0; i<cnt-1; i++)		
fill_byte[i] /* must be '10100101' */	8	uimsbf
return cnt		
case default:		
for (i=0; i<8*(cnt-1)+4; i++)		
other_bits[i]	1	uimsbf
return cnt		
}		
}		

Table 6.25 — Syntax of dynamic_range_info()

Syntax	No. of bits	Mnemonic
dynamic_range_info()		
{		
n = 1		
drc_num_bands = 1		
pce_tag_present	1	uimsbf
if (pce_tag_present == 1) {		
pce_instance_tag	4	uimsbf
drc_tag_reserved_bits	4	
n++		
}		
excluded_chns_present	1	uimsbf
if (excluded_chns_present == 1) {		
n += excluded_channels()		
}		
drc_bands_present	1	uimsbf
if (drc_bands_present == 1) {		
drc_band_incr	4	uimsbf
drc_bands_reserved_bits	4	uimsbf
n++		

Table 6.26 — Syntax of excluded `channels()`

”

7.1, replace “ATDS0” with “ATDS1”

“

”

“

"

“

22

11) Replace the last sentence in the second paragraph of subclause 8.1.2 with the following:

“

However, one non-normative transport stream, called Audio_Data_Transport_Stream (ADTS), is described. It may be used for applications in which the decoder can parse this stream.

”

12) Replace the first two paragraphs of subclause 8.2.3 with the following:

“

Assuming that the start of a raw_data_block is known, it can be decoded without any additional «transport-level» information and produces 1024 audio samples per output channel. The sampling rate of the audio signal, as specified by the **sampling_frequency_index**, may be specified in a program_config_element or it may be implied in the specific application domain. In the latter case, the **sampling_frequency_index** must be deduced in order for the bitstream to be parsed. Since a given **sampling_frequency_index** is associated with only one sampling frequency, and since maximum flexibility is desired in the range of possible sampling frequencies, the following table shall be used to associate an implied sampling frequency with the desired **sampling_frequency_index**. It is used as follows: identify the frequency in the table that is the highest frequency that is less than or equal to the implied frequency, and use the index that is in that same row.

Frequency	sampling_frequency_index
92017	0x0
75132	0x1
55426	0x2
46009	0x3
37566	0x4
27713	0x5
23004	0x6
18783	0x7
13856	0x8
11502	0x9
9391	0xa
0	0xb

Assuming that the start of the first raw_data_block in a raw_data_stream is known, the sequence can be decoded without any additional “transport-level” information and produces 1024 audio samples per raw_data_block per output channel.

”

13) Replace the second item of the second level bulleted list in subclause 8.3.5 with the following:

“

If there is only one group with length eight (num_window_group = 1, window_group_length[0]=8), the results is that spectral data of all eight SHORT_WINDOWS is interleaved by scalefactor window bands.

”

14) Replace definition of **num_valid_cce_elements** in subclause 8.5 with the following:

“

num_valid_cc_elements The number of CCE's that can add to the audio data for this program (Table 6.21)

”

15) Replace definition of **valid_cce_element_tag_select** in subclause 8. 5 with the following:

“

valid_cc_element_tag_select instance_tag of the CCE addressed (Table 6.21)

”

16) Replace subclause 8.7 with the following:

“

8.7 Fill element (FIL) including Dynamic Range Control (DRC)

Bitstream elements:

count	Initial value for length of fill data (Table 6.22)
esc_count	Incremental value of length of fill data (Table 6.22)
extension_type	Four bit field indicating the type of fill element content (Table 6.22)
fill_nibble	Four bit field for fill (Table 6.24)
fill_byte	Byte to be discarded by the decoder (Table 6.24)
other_bits	Bits to be discarded by the decoder (Table 6.24)
pce_tag_present	One bit indicating that program element tag is present (table 6.25).
pce_instance_tag	Tag field that indicates with which program the dynamic range information is associated (table 6.25)
drc_tag_reserved_bits	Reserved (table 6.25)
excluded_chns_present	One bit indicating that excluded channels are present (table 6.25)
drc_bands_present	One bit indicating that DRC multi-band information is present (table 6.25)
drc_band_incr	Number of DRC bands greater than 1 having DRC information (table 6.25)
drc_bands_reserved_bits	Reserved (table 6.25)
drc_band_top[i]	Indicates top of i-th DRC band in units of 4 spectral lines (table 6.25). If $drc_band_top[i]=k$, then the index (w.r.t zero) of the highest spectral coefficient that is in the i-th DRC band is $= k*4+3$. In case of an EIGHT_SHORT_SEQUENCE window_sequence the index is interpreted as pointing into the concatenated array of $8*128$ (de-interleaved) frequency points corresponding to the 8 short transforms.
prog_ref_level_present	One bit indicating that reference level is present (table 6.25).
prog_ref_level	Reference level. A measure of long-term program audio level for all channels combined (table 6.25).
prog_ref_level_reserved_bits	Reserved (table 6.25)
dyn_rng_sgn[i]	Dynamic range control sign information. One bit indicating the sign of dyn_rng_ctl (0 if positive, 1 if negative, table 6.25)
dyn_rng_ctl[i]	Dynamic range control magnitude information (table 6.25)
exclude_mask[i]	Boolean array indicating the audio channels of a program that are excluded from DRC processing using this DRC information.
additional_excluded_chns[i]	One bit indicating that additional excluded channels are present (table 6.26)

Fill elements have to be added to the bitstream if the total bits for all audio data together with all additional data is lower than the minimum allowed number of bits in this frame necessary to reach the target bitrate. Dynamic Range Control (DRC) bits must be added to the fill element whenever the encoder wishes to include DRC information. Under normal conditions fill bits are avoided and free bits are used to fill up the bit reservoir. Fill bits are written only if the bit reservoir is full. Any number of fill elements are allowed.

Decoding process:

The syntactic element **count** gives the initial value of the length of the fill data. In the same way as for the data element this value is incremented with the value of **esc_count** if **count** equals 15. The resulting number gives the number of bytes to be read.

DRC Decoding process:

Fill elements containing an extension_payload with a extension_type of EXT_DYNAMIC_RANGE (see below) are reserved for dynamic range information. In this case the fill_element **count** field must be set equal to the total length, in bytes, of all dynamic range information plus the extension_type field.

prog_ref_level_present indicates that **prog_ref_level** is being transmitted. This permits **prog_ref_level** to be sent as infrequently as desired (e.g. once), although periodic transmission would permit break-in.

prog_ref_level is quantized in 0.25 dB steps using 7 bits, and therefore has a range of approximately 32 dB. It indicates program level relative to full scale (i.e. dB below full scale), and is reconstructed as:

$$level = 32767 \cdot 2^{-prog_ref_level/24}$$