**INTERNATIONAL STANDARD ISO/IEC 14496-3:2005**
TECHNICAL CORRIGENDUM 5

Published 2008-04-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE



# Information technology — Coding of audio-visual objects —

## Part 3: **Audio**

TECHNICAL CORRIGENDUM 5

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

*Partie 3: Codage audio*

*RECTIFICATIF TECHNIQUE 5*

Technical Corrigendum 5 to ISO/IEC 14496-3:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

---

**ICS 35.040** **Ref. No. ISO/IEC 14496-3:2005/Cor.5:2008(E)**



Published in Switzerland

*In the following, changes in existing text and tables are highlighted by gray background.*

*In subclause 4.5.1.2 "Program config element (PCE)", replace:*

A program_config_element() may occur outside the AAC payload e. g. as part of the GASpecificConfig() or the adif_header(), but also inside the AAC payload as syntactic element in a raw_data_block().

Note that the channel configuration given in a program_config_element() inside the AAC payload is evaluated only, if no channel configuration is given outside the AAC payload. In the context of ISO/IEC 14496-3 this is only the case for MPEG-4 ADTS with channel_configation==0.

In any case only one program may be configured at a certain time.

*with:*

A program_config_element() may occur outside the AAC payload e. g. as part of the GASpecificConfig() or the adif_header(), but also inside the AAC payload as syntactic element in a raw_data_block().

Note that the channel configuration given in a program_config_element() inside the AAC payload is evaluated only, if no channel configuration is given outside the AAC payload. In the context of ISO/IEC 14496-3 this is only the case for MPEG-4 ADTS with channel_configation==0.

The sampling_frequency_index given in a program_config_element() can indicate a nominal sampling frequency that is different from the actual sampling frequency, i.e., the intended sampling frequency of the decoder output signal. This is the case when an actual sampling frequency is used that cannot be represented by means of the sampling_frequency_index in the program_config_element(). The actual sampling frequency is signaled in the AudioSpecificConfig() or is implicitly known to the system. The relation between the actual sampling frequency and the nominal sampling frequency is defined in Table 4.68 in subclause 4.5.1.1.

In any case only one program may be configured at a certain time.

*In subclause 4.6.7.3 (Decoding process), remove:*

To reduce memory consumption the samples in the LTP-buffer (x_rec) are stored as 16-bit integer numbers. These rounded 16-bit integer numbers are used to calculate the predicted time domain signal. The exact rounding algorithm to be used is shown in the following pseudo-C function LTP_Round().

```
static short
LTP_Round(double x_rec)
{
  short out_value;

  if(x_rec > 0x7FFF)
    out_value = 0x7FFF;
  else if(x_rec < -0x8000)
   out_value = -0x8000;
```

```
  else
  {
    out_value = INT(|x_rec| + 0.5f);
    if(x_rec  < 0.0f) out_value = -out_value;
  }

  return (out_value);
}
```

*In subclause 4.B.10 (Long term prediction), add:*

In the case of LTP tool it is necessary for the encoder to be aware of rounding noise in the decoder. Long periods of high feedback gain may make rounding noise audible. The encoder is responsible for keeping noise accumulation within reasonable limits by controlling the gain and, e.g., switching the predictor off often enough.

Particularly important is to take into account that the decoding algorithm has been updated. The old decoders have a rounding to 16 bits at the entry of the feedback delay line. This shows up as a noise source compared to the new decoders (and encoder expectations). A simple tested solution to keep this noise within reasonable limits (K=15 accuracy for the decoder) is to reset (switch off for 3 frames) the predictor every 60 frames.

*In subclause 9.5.1 "Bitrates", Table 9.1 "bitrate depending on bitrate_index and sampling frequency", in row '0000' replace "free" by "forbidden" (three times).*

*Remove subclause 9.5.5 "Intensity stereo mode".*

*In subclause 9.C.3 (Reconstruction instructions), replace:*

The simplest method sets the bitrate to the maximum size. This is the preferred method when feeding existing MPEG-1/2 Layer 3 decoders. main_data_begin is set to zero. Stuffing bits are added either before or after ancillary data (see Figure 9.C.1, example A and B).

A more advanced method can be derived from this simple method by setting the bitrate_index to the nearest higher value that corresponds to the length of the mp3_channel_element. With this modification, the bitrate can be significantly reduced (see Figure 9.C.1, example C and D).

To avoid the necessity of stuffing, main_data_begin is set to the value pointing back to the end of main_data of the previous frame. The bitrate_index is now set to the nearest higher value that matches the mp3_channel_element length minus main_data_begin of the current audio frame (see Figure 9.C.1, example E). Only if main_data_begin would exceed the allowed value, stuffing has to be performed.

*with:*

The simplest method sets the bitrate to the maximum size. This is the preferred method when feeding existing MPEG-1/2 Layer 3 decoders. main_data_begin is set to zero. Stuffing bits are added either before or after ancillary data (see Figure 9.C.1, example A and B).

A more advanced method can be derived from this simple method by setting the bitrate_index to the nearest higher value that corresponds to the length of the mp3_channel_element. With this modification, the bitrate can be significantly reduced (see Figure 9.C.1, example C and D).

Note that for sampling rates up to 24 kHz (i.e. in cases where one granule forms a frame), the granule size might exceed the maximum frame size. Even the maximum bitrate index might not allow to store a whole frame after the header (main_data_begin=0). This results from the fact that the maximum length of a granule