



INTERNATIONAL STANDARD ISO/IEC 14496-3:2009
TECHNICAL CORRIGENDUM 7

Published 2015-11-15

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 7

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

RECTIFICATIF TECHNIQUE 7

Technical Corrigendum 7 to ISO/IEC 14496-3:2009 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 14496-3:2009/COR7:2015

Information technology — Coding of audio-visual objects — Part 3: Audio, TECHNICAL CORRIGENDUM 7

Add the following Clauses after 4.6.20.4

4.6.20.5 Adaptation to systems using lower sampling rates

4.6.20.5.1 General

For certain applications, ER AAC LD can change the playout sample rate in order to avoid additional resampling steps (see 4.6.17.2.7). ER AAC ELD can apply similar downscaling steps using the Low Delay MDCT window and the LD-SBR tool. In case AAC-ELD operates with the LD-SBR tool, the downscaling factor is limited to multiples of 2. Without LD-SBR, the downsampled frame size needs to be an integer number.

4.6.20.5.2 Downscaling of Low Delay MDCT window

The LD-MDCT window w_{LD} for $N=1024$ is downsampled by a factor F using a segmental spline interpolation. The number of leading zeros in the window coefficients, i.e. $N/8$, determines the segment size. The downsampled window coefficients $w_{LD,d}$ are used for the inverse MDCT as described in 4.6.20.2 but with a downsampled window length $N_d = N / F$. Please note that the algorithm is also able to generate downsampled lifting coefficients of the LD-MDCT. The following pseudo code outlines the window interpolation algorithm.

```
fs_window_size = 2048; /* Number of fullscale window coefficients. According to ISO/IEC 14496-3:2009,
                        use 2048. For lifting implementations, please adjust this variable accordingly */
ds_window_size = N * fs_window_size / (1024 * F); /* downsampled window coefficients; N determines the
                                                transformation/window length according to 4.6.20.2 */

fs_segment_size = 128;
num_segments    = fs_window_size / fs_segment_size;
ds_segment_size = ds_window_size / num_segments;
tmp[128], y[128], c[128], r[128]; /* temporary buffers */

/* loop over segments */
for (b = 0; b < num_segments; b++) {
    /* copy current segment to tmp */
    copy(&w_LD[b * fs_segment_size], tmp, fs_segment_size);

    /* apply cubic spline interpolation for downscaling */
    /* calculate interpolating phase */
    phase = (fs_window_size - ds_window_size) / (2 * ds_window_size);

    /* calculate the coefficients c of the cubic spline given tmp */
    /* array of precalculated constants */
    m = {0.1666666672, 0.25, 0.2666666681, 0.267857134,
         0.267942578, 0.267948717, 0.267949164};
    n = fs_segment_size; /* for simplicity */

    /* calculate vector r needed to calculate the coefficients c */
    for (i = n - 3; i >= 0; i--)
        r[i] = 3 * ((tmp[i + 2] - tmp[i + 1]) - (tmp[i + 1] - tmp[i]));
    for (i = 1; i < 7; i++)
        r[i] -= m[i - 1] * r[i - 1];
    for (i = 7; i < n - 4; i++)
        r[i] -= 0.267949194 * r[i - 1];

    /* calculate coefficients c */
    c[n - 2] = r[n - 3] / 6;
    c[n - 3] = (r[n - 4] - c[n - 2]) * 0.25;
    for (i = n - 4; i > 7; i--)
        c[i] = (r[i - 1] - c[i + 1]) * 0.267949194;
    for (i = 7; i > 1; i--)
        c[i] = (r[i - 1] - c[i + 1]) * m[i - 1];
    c[1] = r[0] * m[0];
    c[0] = 2 * c[1] - c[2];
    c[n - 1] = 2 * c[n - 2] - c[n - 3];
```

```

/* keep original samples in temp buffer y because samples of
   tmp will be replaced with interpolated samples */
copy(tmp, y, fs_segment_size);

/* generate downsampled points and do interpolation */
for (k = 0; k < ds_segment_size; k++) {
    step = phase + k * fs_segment_size / ds_segment_size;
    idx = floor(step);
    diff = step - idx;
    di = (c[idx + 1] - c[idx]) / 3;
    bi = (y[idx + 1] - y[idx]) - (c[idx + 1] + 2 * c[idx]) / 3;
    /* calculate downsampled values and store in tmp */
    tmp[k] = y[idx] + diff * (bi + diff * (c[idx] + diff * di));
}

/* assemble downsampled window */
copy(tmp, &W_LD_d[b * ds_segment_size], ds_segment_size);
}

```

4.6.20.5.3 Downscaling of Low Delay SBR tool

4.6.20.5.3.1 General

In case the Low Delay SBR tool is used in conjunction with ELD, this tool can be downsampled to lower sample rates, at least for downscaling factors of a multiple of 2. The downscale factor F controls the number of bands used for the CLDFB analysis and synthesis filter bank. The following two paragraphs describe a downsampled CLDFB analysis and synthesis filter bank, see also 4.6.19.4. Please note, that also the LD-SBR synchronization and timing (see Figure 4.47) need to be scaled by the factor F .

4.6.20.5.3.2 Downsampled analyses CLDFB filter bank

- Define number of downsampled CLDFB bands $B=32/F$.
- Shift the samples in the array x by B positions. The oldest B samples are discarded and B new samples are stored in positions 0 to $B-1$.
- Multiply the samples of array x by the coefficient of window ci to get array z . The window coefficients ci are obtained by linear interpolation of the coefficients c , i.e. through the equation

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.

The window coefficients of c can be found in Table 4.A.90.

- Sum the samples to create the 2B-element array u

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen..

- Calculate B new subband samples by the matrix operation Mu , where

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.

In the equation, $\exp()$ denotes the complex exponential function and i is the imaginary unit.

4.6.20.5.3.3 Downsampled synthesis CLDFB filter bank

- Define number of downsampled CLDFB bands $B=64/F$.
 - Shift the samples in the array v by $2B$ positions. The oldest $2B$ samples are discarded.
 - The B new complex-valued subband samples are multiplied by the matrix N , where
- Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.**