3GPP TS 26.402 V1.0.0 (2004-06)

Technical Specification

**3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
General Audio Codec audio processing functions;
Enhanced aacPlus general audio codec;
Additional Decoder Tools
(Release 6)**

Keywords

CODEC, SBR, Enhanced aacPlus, General audio coder

Contents

Foreword

The present document describes tools used in the Enhanced aacPlus general audio codec for the general audio service within the 3GPP system.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of this TS, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 Indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the specification;

1 Scope

This Telecommunication Standard (TS) describes the error concealment algorithm, SBR parameter downmix and output resampling for the Enhanced aacPlus general audio codec, based on the MPEG-4 High Efficiency AAC Profile decoder including Parametric Stereo as specified in [1], [2], [3] and [4].

2 Normative references

This TS incorporates by dated and undated reference, provisions from other publications. These normative references are cited in the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this TS only when incorporated in it by amendment or revision. For undated references, the latest edition of the publication referred to applies.

[1] ISO/IEC 14496-3:2001, Information technology - Coding of audio-visual objects - Part 3: Audio.

[2] ISO/IEC 14496-3:2001/Amd.1:2003, Bandwidth Extension.

[3] ISO/IEC 14496-3:2001/Amd.1:2003/DCOR1.

[4] ISO/IEC 14496-3:2001/FDAM2, Parametric Coding for High Quality Audio.

[5] 3GPP TS 26.401: Enhanced aacPlus general audio codec; General Description

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of this TS, the following definitions apply:

band: (as in limiter band, noise floor band, etc.) a group of consecutive QMF subbands

envelope scalefactor: an element representing the averaged energy of a signal over a region described by a frequency band and a time segment

frequency band: interval in frequency, group of consecutive QMF subbands

frequency border: frequency band delimiter, expressed as a specific QMF subband

noise floor: a vector of noise floor scalefactors

noise floor scalefactor: an element associated with a region described by a frequency band and a time segment, representing the ratio between the energy of the noise to be added to the envelope adjusted HF generated signal and the energy of the same

SBR envelope: a vector of envelope scalefactors

SBR frame: time segment associated with one SBR extension data element

SBR range: the frequency range of the signal generated by the SBR algorithm

subband: a frequency range represented by one row in a QMF matrix, carrying a subsampled signal

time border: time segment delimiter, expressed as a specific time slot

time segment: interval in time, group of consecutive time slots

time / frequency grid: a description of SBR envelope time segments and associated frequency resolution tables as well as description of noise floor time segments

time slot: finest resolution in time for SBR envelopes and noise floors. One time slot equals two subsamples in the QMF domain

3.2 Symbols

For the purposes of this TS, the following symbols apply:

$\mathbf{E}_{Orig}$	has L_E columns where each column is of length N_{Low} or N_{High} depending on the frequency resolution for each SBR envelope. The elements in $\mathbf{E}_{Orig}$ contains the envelope scalefactors of the original signal.
Fs_{out}	the output sampling rate from the SBR Tool.
Fs_{SBR}	internal sampling frequency of the SBR Tool, twice the sampling frequency of the core coder (after sampling frequency mapping, ISO/IEC 14496-3:2001, Table 4.55). The sampling frequency of the SBR enhanced output signal is equal to the internal sampling frequency of the SBR Tool, unless the SBR Tool is operated in downsampled mode. If the SBR Tool is operated in downsampled mode, the output sampling frequency Fs_{out} is equal to the sampling frequency of the core coder.
$\mathbf{F} = \left[\mathbf{f}_{TableLow}, \mathbf{f}_{TableHigh}\right]$	has two column vectors containing the frequency border tables for low and high frequency resolution.
$\mathbf{f}_{TableHigh}$	is of length N_{High}+1 and contains frequency borders for high frequency resolution SBR envelopes.
$\mathbf{f}_{TableLow}$	is of length N_{Low}+1 and contains frequency borders for low frequency resolution SBR envelopes.
L_E	number of SBR envelopes.
L_Q	number of noise floors.
N_Q	number of noise floor bands.
$\mathbf{n} = [N_{Low}, N_{High}]$	number of frequency bands for low and high frequency resolution.
$numTimeSlots$	number of SBR envelope time slots that exist within an AAC frame, 16 for a 1024 AAC frame and 15 for a 960 AAC frame.
$\mathbf{panOffset} = [24,12]$	offset-values for the SBR envelope and noise floor data, when using coupled channels.
$\mathbf{Q}_{Orig}$	has L_Q columns where each column is of length N_Q and contains the noise floor scalefactors.
$\mathbf{r} = [r_0, ..., r_{L-1}]$	frequency resolution for all SBR envelopes in the current SBR frame, zero for low resolution, one for high resolution.
$\mathbf{t}_E$	is of length L_E+1 and contains start and stop time borders for all SBR envelopes in the current SBR frame.
$\mathbf{t}_Q$	is of length L_Q+1 and contains start and stop time borders for all noise floors in the current SBR frame.
$\mathbf{Y}$	is the complex output QMF bank subband matrix from the HF adjuster.

3.3 Abbreviations

For the purposes of this TS, the following abbreviations apply.

SBR	Spectral Band Replication

4 Outline description

This TS is structured as follows:

Section 5 gives a detailed description of the error concealment algorithms in the Enhanced aacPlus decoder. In Section 5.1 the error concealment of the AAC is described, and in section 5.2 the error concealment of the SBR algorithm is outlined.

Section 6 gives a detailed description of how stereo SBR parameters are down mixed to mono SBR parameters.

Section 7 gives a detailed description of the additional downsampler tool, enabeling the Enhanced aacPlus codec to give output sampling rates of 8 and 16kHz, disregarded the sampling rate used for the coded signal.

5 Error concealment

5.1 AAC error concealment

The AAC core decoder includes a concealment function that increases the delay of the decoder by one frame.

There are various tests inside the core decoder, starting with simple CRC tests and ending in a variety of plausibility checks. If such a check indicates an invalid bitstream, then concealment is applied. Concealment is also applied when the calling main program indicates a distorted or missing data frame using the frameOK flag. This is used for error detection on the transport layer.

Concealment works on the spectral data just before the final frequency to time conversion. In case a single frame is corrupted, concealment interpolates between the last good and the first good frame to create the spectral data for the missing frame. Always the previous frame will be processed by the frequency to time conversion, so here the missing frame to be replaced is the previous frame, the last good frame is the frame before the previous one and the first good frame is the actual frame. If multiple frames are corrupted, concealment implements first a fade out based on slightly modified spectral values from the last good frame. As soon as good frames are available, concealment fades in the new spectral data.

Interpolation of one corrupt frame:

In the following the actual frame is frame number *n*, the corrupt frame to be interpolated is the frame *n*-1 and the last but one frame has the number *n*-2.

The determination of window sequence and the window shape of the corrupt frame follows from the table below:

Table 1: Interpolated window sequences and window shapes

window sequence *n-2*	window sequence *n*	window sequence *n*-1	window shape *n*-1
ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	ONLY_LONG_SEQUENCE	0
ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	EIGHT_SHORT_SEQUENCE	LONG_START_SEQUENCE	1
EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE	1
EIGHT_SHORT SEQUENCE	ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	LONG_STOP_SEQUENCE	0

The scalefactor band energies of frames *n*-2 and *n* are calculated. If the window sequence in one of these frames is an EIGHT_SHORT_SEQUENCE and the final window sequence for frame *n*-1 is one of the long transform windows, the scalefactor band energies are calculated for long block scalefactor bands by mapping the frequency line index of short block spectral coefficients to a long block representation. The new interpolated spectrum is built by reusing the spectrum of the older frame *n*-2 multiplying a factor to each spectral coefficient. An exception is made in the case of a short window sequence in frame *n*-2 and a long window sequence in frame *n*, here the spectrum of the actual frame n is modified by the interpolation factor. This factor is constant over the range of each scalefactor band and is derived from the scalefactor band energy differences of frames *n*-2 and *n*. Finally the sign of the interpolated spectral coefficients will be flipped randomly.

Fade out and in:

A complete fading out takes 5 frames. The spectral coefficients from the last good frame are copied and attenuated by a factor of:

$$fadeOutFac = 2^{-(nFadeOutFrame/2)}$$

with $nFadeOutFrame$ as frame counter since the last good frame.

After 5 frames of fading out the concealment switches to muting, that means the complete spectrum will be set to 0.

The decoder fades in when receiving good frames again. The fade in process takes 5 frames, too and the factor multiplied to the spectrum is:

$$fadeInFac = 2^{-(5-nFadeInFrame)}$$

where $nFadeInFrame$ is the frame counter since the first good frame after concealing multiple frames.

5.2 SBR error concealment

The SBR error concealment algorithm is based on using previous envelope and noise-floor values with an applied decay, as a substitute for the corrupt data. In the flowchart of Figure 1 the basic operation of the SBR error concealment algorithm is outlined. If the frame error flag is set, error concealment bitstream data is generated to be used instead of the corrupt bitstream data. The concealment data is generated according to the following.

The time frequency grids are set to:

$L_E = 1$

$\mathbf{t}_E(0) = \mathbf{t}'_E(L'_E) - numTimeSlots$

$\mathbf{t}_E(1) = numTimeSlots$

$\mathbf{r}(l) = HI \quad , 0 \le l < L_E$

$bs\_pointer = 0$

$L_Q = 1$

$\mathbf{t}_Q = \left[\mathbf{t}_E(0), \mathbf{t}_E(1)\right]$

The delta coding direction for both the envelope data and noise-floor data are set to be in the time-direction. The envelope data is calculated according to:

$$\mathbf{E}_{Delta}(k,l) = \begin{cases} -step & , \mathbf{E}_{prev}(k,l) > target \\ step & , otherwise \end{cases} \quad , 0 \le k < \mathbf{n}(\mathbf{r}(l)), 0 \le l < L_E$$

where

$$step = \begin{cases} 2 & , if \quad bs\_amp\_res = 1 \\ 1 & , otherwise \end{cases}$$

$$target = \begin{cases} \mathbf{panOffset}(bs\_amp\_res) & , if \quad bs\_coupling = 1 \\ 0 & , otherwise \end{cases}$$

And where *bs_amp_res* and *bs_coupling* are set to the values of the previous frame.

The noise floor data is calculated according to:

$$\mathbf{Q}_{Delta}(k,l) = 0 \quad , \begin{cases} 0 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

Furthermore, the inverse-filtering levels in *bs_invf_mode* are set to the values of the previous frame, and all elements in *bs_add_harmonic* are set to zero.

If the frame error is not set, the present time grid and envelope data may need modification if the previous frame was corrupt. If the previous frame was corrupt the time grid of the present frame is modified in order to make sure that there is a continuous transition between the frames. The envelope data for the first envelope is modified according to:

$$\mathbf{E}_{mod}(k,0) = \mathbf{E}(k,0) + a \cdot \log_2\left(\frac{\mathbf{t}_E(1) - \mathbf{t}_E(0)}{\mathbf{t}_E(1) - estimated\_start\_pos}\right), \quad 0 \le k < \mathbf{F}(\mathbf{r}(l), 0)$$

where

$$estimated\_start\_pos = \mathbf{t}'_E(L'_E) - numberTimeSlots\ .$$

After the delta coded data has been decoded, a plausibility check is performed to make sure that the decoded data is within reasonable limits. The required limits are:

For the envelope data the logarithmic values shall fulfil:

$$\mathbf{E}(k,l) \le \begin{cases} 35 & , ampRes = 0 \\ 70 & , ampRes = 1 \end{cases}$$

otherwise the frame will be considered corrupt.

The time grids are also verified according to the following rules (if any of the below is true the frame is considered to be corrupt):

- $L_E < 1$
- $L_E > 5$
- $L_Q > 2$
- $\mathbf{t}_E(0) < 0$
- $\mathbf{t}_E(0) \geq \mathbf{t}_E(L_E)$
- $\mathbf{t}_E(0) > 3$
- $\mathbf{t}_E(L_E) < 16$
- $\mathbf{t}_E(L_E) > 19$
- $\mathbf{t}_E(l) \geq \mathbf{t}_E(l+1), 0 \leq l < L_E$
- $l_A > L_E$
- $L_E = 1\ AND\ L_Q > 1$
- $\mathbf{t}_Q(0) \neq \mathbf{t}_E(0)$
- $\mathbf{t}_Q(L_Q) \neq \mathbf{t}_E(L_E)$
- $\mathbf{t}_Q(l) \geq \mathbf{t}_Q(l+1), 0 \leq l < L_Q$
- all elements of $\mathbf{t}_Q$ are not among the elements of $\mathbf{t}_E$

If the plausibility check fails, the frame error flag is set and the error concealment outlined above is applied.

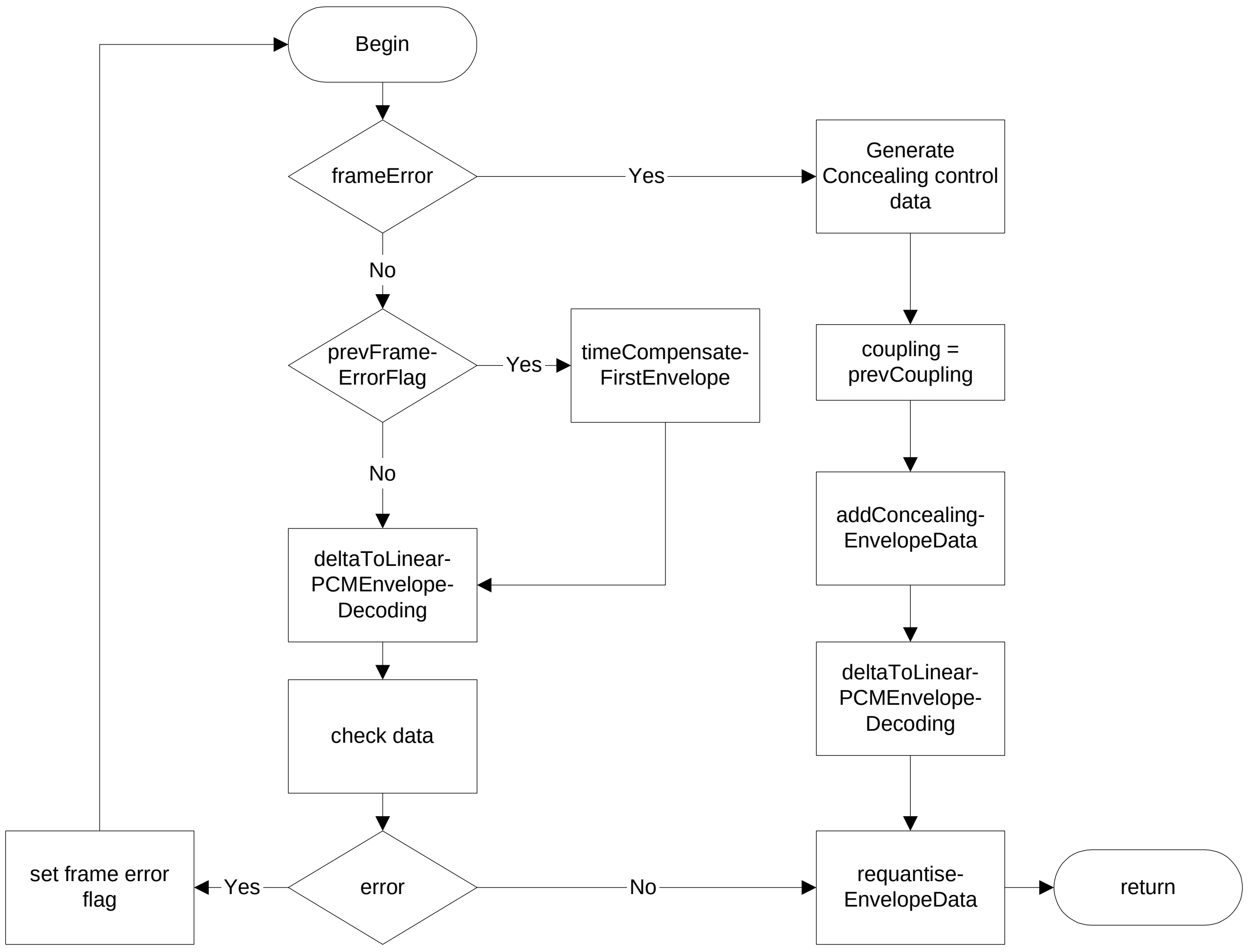
Begin
frameError
Yes
Generate Concealing control data
No
prevFrame-ErrorFlag
Yes
timeCompensate-FirstEnvelope
coupling = prevCoupling
No
addConcealing-EnvelopeData
deltaToLinear-PCMEnvelope-Decoding
deltaToLinear-PCMEnvelope-Decoding
check data
set frame error flag
Yes
error
No
requantise-EnvelopeData
return

Figure 1: SBR error concealment overview

6 SBR stereo parameter to mono parameter downmix

This module enables a decoder only capable of mono output to downmix stereo SBR parameters to mono parameters, hence mono decoding can be performed instead of a stereo decoding.

For all the descriptions in this section left and right applies to the two stereo channels that are being mapped to one channel from here on called merged.

6.1 Inverse filtering

The inverse filtering is mapped from stereo to mono by taking the maximum value from left or right as shown below:

$$\begin{aligned}&\text{for } \left(n \;=\; 0; n < N_Q; n++\right)\\&\quad bs\_sbr\_invf\_mode_{Merged}\left(n\right) = MAX\left(bs\_sbr\_invf\_mode_{Left}\left(n\right), bs\_sbr\_invf\_mode_{Right}\left(n\right)\right)\end{aligned}$$

6.2 Additional harmonics

The additional sinusoids in the merged mono bitstream are the union of the additional sinusoids present in left or right channel as shown below.

```
for $\left(n\ =\ 0; n < N_{High}; n++\right)$
    $bs\_add\_harmonic_{Merged}(n) = OR\left(bs\_add\_harmonic_{Left}(n), bs\_add\_harmonic_{Right}(n)\right)$
```

6.3 Envelope time borders

The merging of the time envelope grid is explained by using $\mathbf{t}_E$ which contains all start/stop borders for all SBR envelopes in the current SBR frame, and the creation of $\mathbf{t}_E$ from the bitstream is explained in [1].

Merging the two $\mathbf{t}_E$ into one is done in the following sequence

1. The start border value of the merged $\mathbf{t}_E$ is set to the largest of the start border values for the left and right channels.
2. The union of all borders for left and right channel values larger than the merged start border is added to the merged $\mathbf{t}_E$
3. All envelopes smaller than two time slots are removed from the merged $\mathbf{t}_E$. This is achieved by starting from the beginning of $\mathbf{t}_E$ and remove all stop borders that are closer than 2 from the start border.
4. If there are more than 5 envelopes the number of envelopes are reduced. This is achieved by starting from the end of $\mathbf{t}_E$ and search towards the beginning of $\mathbf{t}_E$ for a envelope smaller than 4 and remove the start border of that envelope, this continues until there are 5 envelopes left.

The index l_A defined in Table 4.119 in [1] has to be merged into one from two. The merging algorithm uses the following recursion:

```
If $\left(l_{A\_Left} == -1\right)$
    $l_{A\_Merged} = l_{A\_Right}$
else
    If $\left(l_{A\_Right} == -1\right)$
        $l_{A\_Merged} = l_{A\_Left}$
    else
        $l_{A\_Merged} = \min\left(l_{A\_Left}, l_{A\_Right}\right)$
```

6.4 Noise time borders

The number of noise floors L_Q and the start/stop borders for the noise $\mathbf{t}_Q$ is calculated according to [1] for left and right channel and then merged as shown below.

```
$L_{Q\_Merged} = MAX\left(L_{Q\_Left}, L_{Q\_Right}\right)$
$if\ \left(L_{Q\_Merged} > 1\right)$
    $\mathbf{t}_{Q\_Merged}(0) = \mathbf{t}_{E\_Merged}(0)$
    $\mathbf{t}_{Q\_Merged}(2) = \mathbf{t}_{E\_Merged}\left(L_{E\_Merged}\right)$
    $if\ \left(L_{Q\_Left} < L_{Q\_Right}\right)$
        $\mathbf{t}_{Q\_Merged}(1) = \mathbf{t}_{Q\_Right}(1)$
    $else$
        $If\ \left(L_{Q\_Right} < L_{Q\_Left}\right)$
            $\mathbf{t}_{Q\_Merged}(1) = \mathbf{t}_{Q\_Left}(1)$
        $else$
            $\mathbf{t}_{Q\_Merged}(1) = MIN\left(\mathbf{t}_{Q\_Left}(1), \mathbf{t}_{Q\_Right}(1)\right)$
$else$
    $\mathbf{t}_{Q\_Merged}(0) = \mathbf{t}_{E\_Merged}(0)$
    $\mathbf{t}_{Q\_Merged}(1) = \mathbf{t}_{E\_Merged}\left(L_{E\_Merged}\right)$
```

After this is done. It is possible that the middle noise floor border dose not coincide with any SBR envelope time border. If this is the case the middle noise floor border is set equal to the closest time envelope border in the upwards direction.

6.5 Envelope data

Before merging the actual data the per envelope frequency selection of high and low frequency resolution is merged as shown below.

$$\mathbf{r}_{Merged}(l) = \begin{cases} 0 & ,\mathbf{r}_{Left}\left(h_{Left}(l)\right) = 0 \text{ AND } \mathbf{r}_{Right}\left(h_{Right}(l)\right) = 0 \\ 1 & ,otherwise \end{cases} \quad ,0 \le l < L_{E\_Merged},$$

where $h_{Right}(l)$ is defined by $\mathbf{t}_{E\_Right}\left(h_{Right}(l)\right) \le \mathbf{t}_{E\_Merged}(l) < \mathbf{t}_{E\_Right}\left(h_{Right}(l)+1\right)$ and $h_{Left}(l)$ is defined by

$$\mathbf{t}_{E\_Left}\left(h_{Left}(l)\right) \le \mathbf{t}_{E\_Merged}(l) < \mathbf{t}_{E\_Left}\left(h_{Left}(l)+1\right)$$

The envelope data referred to as $\mathbf{E}_{Orig}$ in [1] for the left and right channels are merged into $\mathbf{E}_{Orig\_Merged}$ as shown below.

$$\mathbf{E}_{Orig\_Merged}(k,l) = \frac{\mathbf{E}_{LeftOrig}\left(g_{Left}(k), h_{Left}(l)\right) + \mathbf{E}_{RightOrig}\left(g_{Right}(k), h_{Right}(l)\right)}{2} \quad ,0 \le k < \mathbf{n}\left(\mathbf{r}_{Merged}(l)\right), 0 \le l < L_{E\_Merged}$$

where $h_{Right}(l)$ is defined by $\mathbf{t}_{E\_Right}\left(h_{Right}(l)\right) \le \mathbf{t}_{E\_Merged}(l) < \mathbf{t}_{E\_Right}\left(h_{Right}(l)+1\right)$ and

$g_{Right}(k)$ is defined by $\mathbf{F}\left(g_{Right}(k), \mathbf{r}_{Right}\left(h_{Right}(l)\right)\right) \le \mathbf{F}\left(k, \mathbf{r}_{Merged}(l)\right) < \mathbf{F}\left(g_{Right}(k)+1, \mathbf{r}_{Right}\left(h_{Right}(l)\right)\right)$ and

$h_{Left}(l)$ is defined by $\mathbf{t}_{E\_Left}\left(h_{Left}(l)\right) \le \mathbf{t}_{E\_Merged}(l) < \mathbf{t}_{E\_Left}\left(h_{Left}(l)+1\right)$ and

$g_{Left}(k)$ is defined by $\mathbf{F}\left(g_{Left}(k), \mathbf{r}_{Left}\left(h_{Left}(l)\right)\right) \le \mathbf{F}\left(k, \mathbf{r}_{Left}(l)\right) < \mathbf{F}\left(g_{Left}(k)+1, \mathbf{r}_{Left}\left(h_{Left}(l)\right)\right)$.

6.6 Noise floor data

The noise floor data is merged as the average between left and right channel data according to the function below.

$$\mathbf{E}_{Orig\_Merged}(k,l) = \frac{\mathbf{Q}_{LeftOrig}\left(k, h_{Left}(l)\right) + \mathbf{Q}_{RightOrig}\left(k, h_{Right}(l)\right)}{2} \quad ,0 \le k < N_Q, 0 \le l < L_{Q\_Merged} \ ,$$

where $h_{Right}(l)$ is defined by $\mathbf{t}_{Q\_Right}\left(h_{Right}(l)\right) \le \mathbf{t}_{Q\_Merged}(l) < \mathbf{t}_{Q\_Right}\left(h_{Right}(l)+1\right)$ and $h_{Left}(l)$ is defined by $\mathbf{t}_{Q\_Left}\left(h_{Left}(l)\right) \le \mathbf{t}_{Q\_Merged}(l) < \mathbf{t}_{Q\_Left}\left(h_{Left}(l)+1\right)$.

7 Output resampler tool

The audio output from an Enhanced aacPlus decoder using a downsampled synthesis filterbank (see 4.6.18.4.3 or 4.6.18.8.2.3 of [2]) is given at the AAC decoder sampling rate Fs. If an output with lower sampling rate is desired, a downsampler tool is required in order to convert the audio output from the source sampling frequency Fs_{out} to the target sampling frequency $Ft < Fs_{out}$.

The downsampler tool consists of three parts connected to the final operations of the SBR decoder as in Figure 1. (Refering to Figure 4.44 of [1]). The QMF bandlimiter operates on QMF samples and is inserted prior to the QMF synthesis bank, the spline resampler and the postfilter operate on synthesised PCM audio samples.

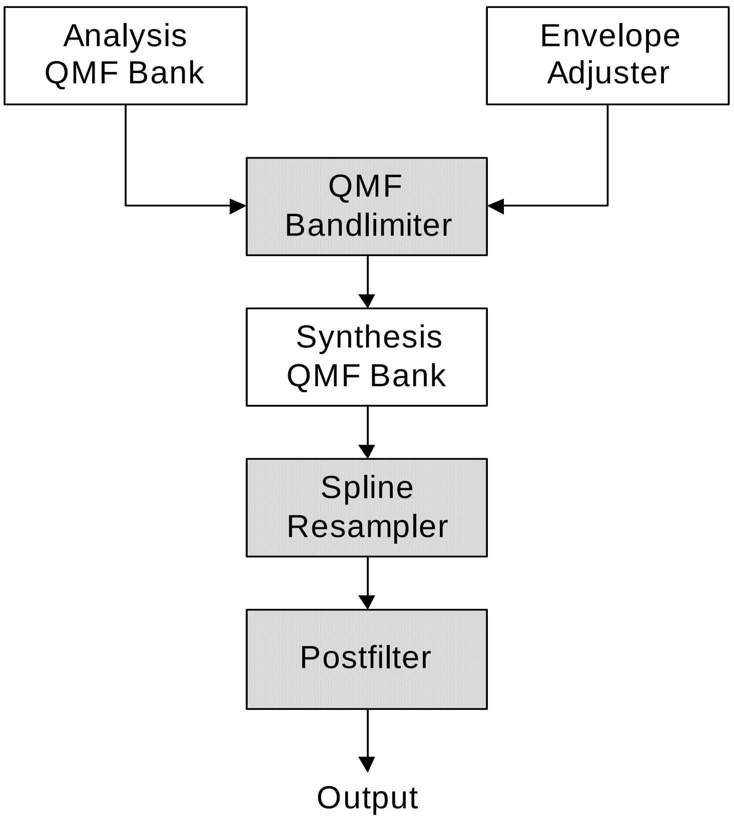

Figure 1: Downsampler tool

7.1 QMF bandlimiter

The QMF bandlimiter maps the final QMF matrix $\mathbf{Y}$ to $\mathbf{Y}_B$, according to the rule

$$\mathbf{Y}_B(k,l) = \begin{Bmatrix} \mathbf{Y}(k,l), & 0 \le k < k_B; \\ 0, & k_B \le k < 32, \end{Bmatrix}, \quad 0 \le l < numTimeSlots \cdot 2,$$

where the cutoff frequency subband index k_B is given by

$$k_B = INT\left(64 \frac{Ft}{Fs_{SBR}}\right).$$

7.2 Spline resampler

Given the discrete time output $x(l)$ from synthesis QMF bank the spline resampler relies on the continuous time signal representation,

$$s(t) = \sum_{l=0}^{\infty} x(l)\,\varphi(Fs_{out} \cdot t - l),$$

where $\varphi(t)$ is a cubic B-spline supported on the interval $[0,3]$.The discrete time output of the resampler is then defined, up to a suitable delay, by $y(n) = s(n/Ft)$. For each $n \ge 0$, define the integer $m(n)$ and the remainder $0 \le \rho(n) < 1$ by

$$\frac{Fs_{out}}{Ft} n = m(n) + \rho(n).$$

Then the spline evaluation results in the time varying causal filtering,

$$y(n) = \sum_{d=0}^{3} B\big(\rho(n), d\big)\, x\big(m(n) - d\big), \quad n = 0,1,\ldots,$$

with homogeneous initialization $x(l) = 0,\, l < 0,$ and filter coefficients

$$B(\rho,d) = \begin{cases} \frac{1}{6}\rho^3, & d = 0; \\ \frac{1}{2}(-\rho^3 + \rho^2 + \rho) + \frac{1}{6}, & d = 1; \\ \frac{1}{2}\rho^3 - \rho^2 + \frac{2}{3}, & d = 2; \\ \frac{1}{6}(1-\rho)^3, & d = 3. \end{cases}$$

7.3 Postfilter

In order to compensate for a small loss in signal power for high frequencies, a first order IIR postfilter is applied to the output $y(n)$ of the spline resampler, resulting in the final output $z(n)$, where $z(-1) = 0$ and

$$z(n) = y(n) + \alpha\big(y(n) - z(n-1)\big), \quad n = 0,1,\ldots.$$

The value of the parameter α is given by Table 1.

Table 1: Postfilter coefficient α

Fs_{out}	Ft	
	8 kHz	16 kHz
22.05 kHz	0.06	0.30
24 kHz	0.05	0.24

Annex A (informative): Change history

Change history							
Date	**TSG SA#**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
06-2004	SP-24	SP-040429	-	-	Presentation to TSG SA for information	-	1.0.0