Free Speech (and Music) for Digital Radio Mondiale Plus (DRM+)

v1.1

*A proposal for using the open-source audio codec **CELT** inside a DRM+ transmission chain.*

Motivation

Among other voice coding algorithms DRM uses Advanced Audio Coding plus SBR (AAC+) as main audio coding scheme[1]. The usage of the AAC+ encoder in DRM broadcast systems requires a license, thus complicating the life of enthusiastic hobbyists which want to play around with this great piece of digital radio technology. In the past, modifications of the free advanced audio coder (FAAC) have been made, but only with monaural streaming and without SBR. However, the integration of FAAC into the DRM transmission chain required a license if the codec was used for broadcasting.

In 2008 a new high-quality audio codec named CELT[2] entered the world of audio coding. The codec is well suited for the usage in a digital radio transmission chain due to the following features:

- Very low-latency
- Low computational complexity (low memory and computational requirements)
- Bit-error tolerant
- Support for both, speech and music
- A quality/bitrate trade-off
- Packet loss concealment
- Free of patents

Although CELT owns all these remarkable properties, in terms of quality AAC+SBR outperforms CELT at bitrates below 30 kBits per second, especially for music broadcasting. Therefore, it cannot be recommended to use CELT for stereo transmissions in the DRM modes A, B, C and D as the achievable bitrates in these modes might not deliver acceptable quality. However, in DRM mode E where robust broadcasting can be established at bitrates above 30 kBits/s CELT can play out its strengths.

The following section describes how CELT can be integrated into a DRM mode E transmission chain.

Technical Integration of CELT into DRM

Audio coding is commonly performed on a frame-wise basis, i.e. by acquiring a certain number of audio samples at a certain sampling frequency and putting them into an audio coder. Clearly, the longer the audio encoding takes place, the higher the number of audio frames (AF) produced by the

[1] DRM System Specification - ETSI ES 201 980 V3.1.1
[2] www.celt-codec.org

encoder where each frame corresponds to certain duration of audio in time determined by the sampling frequency.

In DRM the encoded audio data is transmitted in the Multiplex Service Channel (MSC). Inside the MSC an integer number of AFs is wrapped inside a so called audio super frame (ASF). DRM uses a modified version of AAC+ which supports 960 audio input samples per AF. With an audio input sampling frequency of 48 kHz one AF corresponds to 960 / 48 kHz = 20 ms in time. In mode E one ASF corresponds to 200 ms in time, and thus it carries exactly 200 ms / 20 ms = 10 audio frames.

This relationship is expressed by the following formula, where `fs` denotes the sampling frequency:

```
nFrames = fs [samples/s] * duration of ASF [s] / 960 [samples/frame]
```

If CELT is used in DRM, processing audio frames with 960 samples per frame, or any other number where the number of audio frames per ASF becomes integer, would be preferable to ease encoding and decoding. Furthermore, when using 960 samples per frame, the CELT encoding and decoding chain becomes similar to the AAC+ encoding and decoding chain in the DRM standard which eases the implementation and migration effort.

In summary, the following CELT codec parameters are recommended for using CELT in DRM mode E:

Samplerate: 48 kHz

Framesize: 960 samples per frame

For AAC encoding, the DRM standard requires a header to be signaled inside the ASF. The header contains the AAC frame borders, i.e. the number of bytes of the encoded frame as a 12 bit value. This value is encoded incrementally. When using CELT, the number of bytes per encoded frame must also be transmitted in order for the receiver to feed the CELT decoder with the correct data. The number of header bytes can be calculated from the number of frames as follows:

```
nHeaderBytes = ceil ((nFrames – 1) * 12 bits / 8.0 bits per byte)
```

For more information on the header encoding please refer to the AAC encoding part of the DRM specification. In DRM mode E, the value `nFrames` is always equal to 10 since we agreed on a sampling rate of 48 kHz. Hence, we obtain `nHeaderBytes`= 14.

In order to calculate the number of bytes per ASF one has to know the number of bytes per transmission frame (TF), i.e. `nBytesPerTF`, which is signaled in the multiplex data entity in the service description channel (SDC). In DRM+ one ASF is spanned over 2 transmission frames, which is why `nBytesPerTF` must be multiplied by a factor of two in order to get the number of bytes per ASF. Furthermore, we have to know if a text message is being transmitted inside the TF which is also signaled in the SDC. If a text message is included in the stream we have to subtract 4 bytes from the number of bytes per transmission frame and the number of bytes per ASF is calculated as follows:

```
nASFBytes = (nBytesPerTF – 4) * 2
```

If no text message is transmitted, the number of ASF bytes is two times the number of bytes per TF:

```
nASFBytes = nBytesPerTF * 2
```

In DRM30, where one ASF is embedded in a single TF, the multiplication by a factor of two can be omitted. A DRM+ content server could now partition the CELT audio frames as follows:

```
nBytesPerFrameInteger = floor((nASFBytes - nHeaderBytes) / nFrames)
```

With our assumptions this formula becomes:

```
nBytesPerFrameInteger = floor((nASFBytes - 14) / 10)
```

In order to fill the whole ASF with data the following data partitioning is proposed:

```
for (iFrame = 0; iFrame < nFrames; iFrame++)
{
      if (iFrame < nFrames - 1)
            nBytesPerFrame[iFrame] = nBytesPerFrameInteger
      else
            nBytesPerFrame[iFrame] =
                  nASFBytes - (nFrames - 1) * nBytesPerFrameInteger
}
```

With that partitioning, the last CELT frame is greater or equal to the previous CELT frames in order to fill the complete ASF with encoded audio data such that the frame borders in the transmitted header can be correctly interpreted by the decoder.

Although the encoder and decoder could be used in DRM mode E with these settings, in order to be able to detect if is CELT frame is valid or not, we will spend 1 byte for a CRC sum which we calculate over the encoded bytes. The 8-bit CRC calculation for one CELT frame is equal to the CRC calculation for the Fast Access Channel (FAC). The polynomial and the calculation procedure can be taken from the Annex D of the DRM system specification.

Since we have to transmit the CRC inside the ASF we have to subtract the number of bytes per CRC from the number of bytes per frame to obtain the actual number of payload bytes per CELT frame as follows:

```
nBytesPerCELTFrame[iFrame] = nBytesPerFrame[iFrame] - 1
```

Thus, we encode the `iFrame`-th audio frame and obtain `nBytesPerCELTFrame[iFrame]` bytes by the CELT encoder and calculate the 8-bit CRC sum over the encoded frame. This CRC value will be transmitted inside the higher protected part of the CELT audio stream equal to the AAC frame CRC transmission.

In conclusion, the mapping of the ASF components such as header, CRC and audio data, and hence the framing of the lower- and higher protected parts are equal to the AAC ASF encoding given in the DRM specification. Thus, migrating from an AAC to a CELT design requires minimal implementation effort.

Finally, the receiver needs to know that the CELT encoder instead of the AAC encoder has been used by the transmitter. In the next section a standard-compatible way of doing such a signaling will be proposed.

Audio Encoder Signaling for CELT in DRM

In order for the decoder to know which audio encoder has been used, some side-information has to be included in the transmitted data stream. This side-information is transmitted in the SDC and included in so called data entities. ~~The data entity which carries the audio encoder information is called "Audio information data entity", or simply data entity type 9.~~

~~For CELT, the audio information data entity fields have to be interpreted as follows:~~

~~Field~~	~~Num. Bits~~	~~Value~~	~~Description~~
~~audio coding~~	~~2 bits~~	~~3 = 11~~	~~CELT codec used~~
~~SBR flag~~	~~1 bit~~	~~0 = 0~~	~~no SBR~~
~~audio mode~~	~~2 bits~~	~~0,1 = 00, 01~~	~~mono = 0, stereo = 1~~
~~audio sampling rate~~	~~3 bits~~	~~6 = 101~~	~~48 kHz~~
~~coder field~~	~~5 bits~~	~~0 = 00000~~	~~reserved~~

~~These are all the necessary changes to make for integrating CELT into the DRM transmission chain without violating the DRM specification.~~

The signaling is subject to change! A future version will use the Data Entity Type 5!

CELT Performance in DRM+

The fixed-point CELT codec implementation version 0.7.1 was successfully implemented in the Spark[3] signal generator by Michael Feilen and the SoDiRa[4] multi-standard software receiver by Bernd Reiser. In this setup the codec was tested in terms of computational complexity, stability and audio quality. Runtime experiments under Linux have shown that in terms of computational complexity the floating-point CELT implementation outperforms the latest floating-point version of the free AAC encoder (FAAC) by a factor of two, whereas both codecs where compiled using the open-source GNU compiler gcc with all optimizations enabled. Although no explicit tests have been made with highly optimized AAC+ encoder libraries, it can be suspected that due to the simpler codec design a highly optimized CELT implementation requires less computational resources.

The perceived audio quality of CELT with respect to a certain DRM+ modulator configuration is depicted by the following table:

Audio Bitrate [kBit/s]	Code Rate (EEP)	Mapping	Transmission-Robustness	Perceived Audio Quality
74.48	1/2	QPSK	Medium	Very high
49.68	1/3	QPSK	High	High
37.20	1/4	QPSK	Highest	Medium

For more information on the integration of CELT into DRM please contact the author at mail@drm-sender.de.

[3] http://www.drm-sender.de
[4] http://www.dsp4swls.de/sodira/sodira.html

Document History

Revision	Last changed	Author	Description
v1.0	26.05.2010	Michael Feilen	Publication
v1.1	31.05.2010		Corrected text message calculation