INTERNATIONAL ORGANIZATION FOR STANDARDIZATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11

MPEG2021/N13955

29 七月 2021, Geneva, Switzerland

Title: WD of 14496-12 Amd.X Improved audio support

Editor: David Singer, Frank Baumgarte

1 Introduction

This is a revision of the proposed amendment that covers audio support. For ease of comprehension this is organized by topic.

<<Needs re-organizing in amendment style.>>

2 Audio

2.1 Introduction

(This section does not form part of the final amendment).

The audio support in the Part 12 file format (and hence MP4, 3GPP, etc.) has a few areas where improvement would be beneficial:

a) we need to improve support for marking the codec-independent characteristics of the audio stream; loudness measures are notably needed, not least to make supporting the Commercial Advertisement Loudness Mitigation (CALM) act in the USA easier. There is also increasing demand for dynamic range control, and some tagging in that area would also be advantageous;

b) we need to support declaring channel layout or speaker assignments; it's important to know what 'geometry' a signal is intended for before attempting to decode it, especially when alternatives are offered.

We separate 'dynamic' data – data which is sometimes or always derived from scanning the stream – from 'static' and setup data. The latter is appropriately placed in the sample entry, as it assists either decoding or presentation of the material, and is, or can be, known in advance. However, placing dynamic data in the sample entry is problematic – the sample entry must be created, and often stored, before the stream contents are known.

2.2 Static Data

2.2.1 Introduction

This places new containers in the "sample description extension" part of the MPEG-4 Systems framework with static data that describes the nature of the associated audio (similarly to the way that, for example, video color spaces can be tagged in the video sample entry).

2.2.2 Dynamic Range Control support (DRC)

A DRC is used in the encoder to generate gain values using one of the pre-defined DRC characteristics as defined in <<Coding-Independent Code-Points 23001-8>>; the coefficients are placed either in-stream or in an associated meta-data track.

For some content, such as some multi-channel content, it may be advantageous to use different DRC characteristics in different channels. For instance, if speech is exclusively present in the center channel, this feature can be very useful. It is supported by the assignment of DRC characteristics to audio channels.

It is possible to declare the loudness characteristics of the signal after DRC.

DRC support includes supporting in-stream DRC coefficients, and a separate track carrying them; the latter is particularly useful for legacy coding systems (including uncompressed audio) that have no provision for in-stream coefficients.

2.2.3 Downmix support

The downmix can be controlled by the production facility if necessary. For instance, some content may require more attenuation of the surround channels before downmixing to maintain intelligibility.

The downmix support is designed so that any downmix (e.g. from 7.1 to quad as well as to stereo) can be described; however, we expect initial support to be primarily for stereo downmixes.

It is possible to declare the loudness characteristics of the signal after downmix, and after DRC and downmix.

2.3 Dynamic Data

2.3.1 Introduction

We place 'dynamic data' in a user-data item, to enable its presence in movie fragments and thus storage after the stream (or a portion of it) has been seen.

In particular, in live scenarios, user-data in the initial movie atom may be a 'promise not to exceed' or 'best guess', and then user-data updates give better (but still generally valid) values. Thus, for example, a loudness range in this user data that is associated with a particular set of DRC instructions constitutes a 'promise' rather than a measurement, under these circumstances.

To do that, we introduce some measurements, and then define the additions to the sample entry that can carry those measurements.

The loudness measurements are drawn from the ITU BS 1770 documents, and from the EBU. There is an amendment to Coding-Independent Code Points to add formal names for these.

Several metadata values are available that describe aspects of the dynamic range. The size of the dynamic range can be useful in adjusting the DRC characteristic, e.g. the DRC is less aggressive if the dynamic range is small or the DRC can even be turned off.

True Peak and maximum loudness values can be useful for estimating the headroom, for instance when loudness normalization results in a positive gain [dB] or when headroom is needed to avoid clipping of the downmix. The DRC characteristic can then be adjusted to approach a headroom target.

2.3.2 Program Loudness and Anchor Loudness

Program loudness is as defined in ITU BS 1770 – applied to the whole signal. Anchor loudness is the loudness of the 'anchor' material; the anchor material is often (but not always) dialog, and Dialnorm is an example of an anchor loudness measure.

[[Ed: this is proposed but not yet agreed]] We also support the carriage of a mixing level, the audio sound pressure level that the content was mixed to. (If audio is listened to at a level other than the mixing level, this can affect the perceived tonal balance.)

2.3.3 Loudness Peak measurements

We include some measures derived from EBU-Tech 3341 and 3342 ([2] and [3]):

- Maximum of the Loudness Range [derived from [2])
- Maximum Momentary Loudness as defined in [3]
- Maximum Short-Term Loudness as defined in [3]

2.3.4 True Peak

The peak value of the associated content is represented here in a coding-independent way. It is as described in BS-1770-3.

2.3.5 Album Loudness

Under some circumstances it can be desirable to indicate the loudness characteristics of an album, in each song that the album contains. A separate box can be specified for that purpose. (It is assumed that the two ends agree on what the 'album' is – probably based on other meta-data – and that songs are unique to a specific album).

2.4 Box Design

2.4.1 Sample Entry

2.4.1.1 Channel Layout

If a V1 sample entry is used, the ChannelCount field in the AudioSampleEntry must be correct. The channel layout can be all or part of a standard layout (from an enumerated list), or a custom layout (which also allows a track to contribute part of an overall layout).

```
aligned(8) class ChannelLayout extends FullBox('chnl') {
      unsigned int(8) definedLayout;      // from Coding-independent code points
      if (definedLayout==0) {
            for (i = 1 ; i <= channelCount ; i++){
                                                                        //
channelCount comes from the sample entry
                  unsigned int(8) speaker_position; // also from CICP
            } else {
                  uint(64)    omittedChannelsMap; // a '1' bit indicates 'not in
this track'
      }     }
}
```

The definedLayout is a ChannelConfiguration from ISO/IEC 23001-8; similarly a speaker_position is an OutputChannelPosition from that specification.

1-bits in the channel map mean that a channel is absent. This assumes that 64 bits is enough for enumerated layouts (64 channels). A zero value of the map therefore always means that the given standard layout is fully present.

2.4.1.2 Downmix Instructions

We allow expression of how to downmix to any configuration, but we only expect to see this box for mono or stereo for the time being. If targetChannelCount is odd, the box is padded with 4 bits set to 0xF. The targetChannelCount must be consistent with the targetLayout (if given).

Each downmix is uniquely identified by an ID; there are two reserved values for the ID, 0 and 0xFF, which must not be used.

```
aligned(8) class DownMixInstruction extends FullBox('dmix') {
      unsigned int(8) targetLayout;             // from CICP
      unsigned int(8) targetChannelCount; // must be <= channelCount
      unsigned int(8) downmix_ID;
      int i, j;
      for (i = 1 ; i <= targetChannelCount; i++){
            for (j=1; j <= channelCount; j++) {
                  unsigned int(4) downmix_coefficient;
            }
      }
}
```

The downmix coefficient is encoded as defined in the following tables:

Value	Hex Encoding (4 bits)
0.00 dB	0x0
-0.50 dB	0x1
-1.00 dB	0x2
-1.50 dB	0x3
-2.00 dB	0x4
-2.50 dB	0x5
-3.00 dB	0x6
-3.50 dB	0x7
-4.00 dB	0x8
-4.50 dB	0x9
-5.00 dB	0xA
-5.50 dB	0xB
-6.00 dB	0xC
-7.50 dB	0xD
-9.00 dB	0xE
-∞ dB	0xF

Table 1: Downmix Coefficient Encoding for non-LFE channels

Value	Hex Encoding (4 bits)
0.00 dB	0x0
-0.50 dB	0x1
-1.00 dB	0x2
-1.50 dB	0x3
-2.00 dB	0x4
-2.50 dB	0x5
-3.00 dB	0x6
-4.00 dB	0x7
-5.00 dB	0x8
-6.00 dB	0x9
-7.50 dB	0xA
-9.00 dB	0xB
-12.00 dB	0xC
-16.00 dB	0xD
-20.00 dB	0xE
-∞ dB	0xF

Table 2: Downmix Coefficient Encoding for LFE channel

2.4.1.3 DRC Information

DRC information is typically present in the stream as sets of values at the stream level, not directly associated with a channel. In this design we therefore specify how to achieve a particular DRC effect (strength) by applying the coefficients given, to the given channels. Each box documents a consistent way to apply DRC to the stream, using in-channel or side-band DRC information, or pass-through (no processing). (How it is carried in-stream, and how multiple in-stream values are distinguished, are not defined at this level, but examples are given).

We expect various DRC characteristics to be defined, or identified from other standards bodies.

The DRC_characteristics may take a value from ISO/IEC 23001-8 (CICP):

0 : No DRC information applied to this channel (for this effect, this channel is un-modified)
1 – N : defined DRC characteristic 1 - N respectively
>N: reserved; should be treated as DRC information with unknown characteristics

The channel count must match the enclosing sample entry, or the applicable downmix box, as appropriate. Each channel is mapped to a coefficient (or set of coefficients for the frequency bands) of a given characteristic and location. The location is a 'coarse' identification of where that set is; sets in a given location are enumerated.

For each codec that can be carried in MP4 files and that also carry DRC information there is a specific definition of how the location is coded, using the fields DRC_location and the sequence_index.

A reserved range (negative values) of DRC_locations indicates that the DRC values are in an associated meta-data track. That track is the nth linked via a track reference of type 'adrc' (audio DRC) from the audio track, where n=abs(DRC_location), and the sample-entry type in the meta-data track indicates how the coefficients are stored. The sequence_index must be sufficient to locate them.

For example, for AAC streams, these fields are used as follows:

DRC_location		**Sequence_index**
1	***dynamic_range_info()***	0 (unused)
2	***MPEG4_ancillary_data()*** (defined in 14496-3DAM4)	0 (unused)

and the similar example for AC-3 streams:

DRC_location		**Sequence_index**
1	***dynrng, dynrng2***	0 (unused)
2	***compr, compr2***	0 (unused)

The DRC_set_ID is a positive integer that is unique for all the DRCInstructions in a given sample entry; it is used to identify this set of instructions in post-DRC loudness measurements.

Two boxes are used to declare DRC information; DRCCoefficients describes and locates the coefficient streams, and DRCInstruction applies them to channels in the audio. A DRC may be applicable before or after a downmix; the downmix_ID, if used, identifies which. In the case that the DRC is applied after downmix, it can indicate that a specific DRC must be applied before downmix by using the depends_on_DRC_set field. That DRC set must be applicable to the original, un-downmixed, audio.

The reserved value 0 for downmix_ID here indicates that the DRC applies to the program base material before any downmix.

The reserved value 0xFF for downmix_ID indicates that the DRC can be applied to any downmix or the base program material. In this case, the channel count must be 1, and the single coefficient stream is applied to all channels.

In the DRCInstructions, the sequence_index is the 1-based index in the DRCCoefficients box, of the 'sequence_count' loop for this DRC_location. If sequence_index is 0, this channel is not processed.

The crossover frequencies f_c and normalized crossover frequencies $f_{c,Norm}$ are defined as follows.

They are normalized by the audio sample rate f_s. The advantage of having normalized numbers as o pposed to nominal frequencies is that a decoder could have tables of static filter coefficients for each of the crossover frequencies. Hence it would not need to compute them on initialization. The normal ized crossover frequencies (crossover_freq_index) are transmitted in a 4 bit field as shown in Table 1. (The numbers are given as fractions because at least the larger numbers then correspond to crosso ver frequencies of common filter banks, such as index 4 and higher for QMF in dual-rate HE-AAC a nd MPEG-D). The values of cross_over_freq_index must increase with increasing band index.

crossover_freq_index	$f_{c,Norm}$
0	2/1024
1	3/1024
2	2/512
3	3/512
4	2/256
5	3/256
6	2/128
7	3/128
8	2/64
9	3/64
10	2/32
11	3/32
12	2/16
13	3/16
14	2/8
15	3/8

Table 1 Coding of normalized crossover frequencies

The crossover frequencies f_c in Hz are computed by:

$$f_c = f_s \times f_{c,Norm}.$$

In case of multi-rate decoder configurations such as HE-AAC, f_s is the sample rate of the final output signal.

The intended program effect of an entire DRC application is declared by DRC_effect. That field takes one of t he following values (we may need some way to allow unregistered effect names, e.g. by URN). Each value is a bit-position, starting from the least significant bit; multiple bits may be set. If no bits are set, the effect is un declared.

DRC Effect bit position	Meaning
1	Late night
2	Noisy environment
3	Limited playback range
4	Clipping protection
5	Sound/special effect
6	Fade-in/fade-out (when gap less content is played stand alone)
7	Dialog enhancement

[[Ed: This is a proposed field and not yet fully specified.]] The DRC_headroom declares the setting given to th e DRC, and can assist any following peak limiter by…

```
aligned(8) class DRCCoefficients extends FullBox('drcc') {
            // N copies of this box, one of these per DRC_location
      unsigned int(8) DRC_location;
      unsigned int(8) sequence_count;
      for (sequence_index=1 ; sequence_index<=sequence_count; sequence_index++){
            // each entry here in 1:1 correspondence with the
            //  gain coefficient sets (for the bands) in the given location
            unsigned int(8) band_count;   // must be >= 1
            unsigned int(8) band0_DRC_characteristics;
            for (j = 2; j <= band_count; j++)
            {
                  unsigned int(8) DRC_characteristics;
                  unsigned int(8) crossover_freq_index;
            }
      }
}
```

```
aligned(8) class DRCInstruction extends FullBox('drci') {
            // N copies, one for each overall combination DRC that can be
applied
      unsigned int(8) DRC_set_ID;   // must be non-zero and unique
      unsigned int(8) downmix_ID;   // if 0 - to base
                                                                  // if non-0,
applies after downmix
                                                                  // if 0xff,
applies before or after downmix
      unsigned int(8) DRC_headroom; // [[ed: proposed, under consideration]]
      unsigned int(32) DRC_effect;
      if (downmix_ID != 0)
      {
            unsigned int(8) depends_on_DRC_set;
            // if non-zero, must match a DRC that must be applied to the base
            //  stream before downmix
      }
      unsigned int(8) channel_count;      // must match channel count of the
signal
      unsigned int(8) DRC_location;
      for (i = 1 ; i <= channel_count; i++){
            unsigned int(8) sequence_index;
                  // if 0, then this channel is not processed
                  // else 1-based index into the DRC sequence array for this
location
      }
}
```

2.4.2 User-data

2.4.2.1 Loudness Information

We need a set of loudness measures; most are logically related (and measured similarly), and one (true peak) is slightly different:

Loudness:

1. Program Loudness [1]
2. Anchor Loudness
3. Loudness Range [2]
4. Maximum of the Loudness Range (derived from [2])
5. Maximum Momentary Loudness (0.4s window) [3]
6. Maximum Short-term Loudness (3s window) [3]
7. Mixing level

Peak value:

8. True Peak Value [1]

2.4.2.2 Loudness Boxes

```
aligned(8) class LoudnessInfo extends FullBox('loud') {
      unsigned int(8) downmix_ID;          // from CICP, matching downmix
      unsigned int(8) DRC_set_ID;          // to match a DRC box
      signed int(16)  peak_value;
      unsigned int(8) measurement_count;
      int i;
      for (i = 1 ; i <= measurement_count; i++){
            unsigned int(8) method_definition;
            unsigned int(8) method_value;
            unsigned int(4) measurement_system;
            unsigned int(4) reliability;
      }
}
aligned(8) class AlbumLoudnessInfo extends FullBox('alou') {
      unsigned int(8) downmix_ID;          // from CICP, matching downmix
      unsigned int(8) DRC_set_ID;          // to match a DRC box
      signed int(16)  peak_value;
      unsigned int(8) measurement_count;
      int i;
      for (i = 1 ; i <= measurement_count; i++){
            unsigned int(8) method_definition;
            unsigned int(8) method_value;
            unsigned int(4) measurement_system;
            unsigned int(4) reliability;
      }
}
```

These two boxes provide loudness information for the song, and for the entire album which contains the song, respectively.

If downmix_ID is zero, then this box declares the loudness characteristics of the layout without downmix. If DRC_set_ID is zero, it declares the characteristics without applying a DRC.

If downmix_ID and/or DRC_set_ID are non-zero, this box declares the loudness characteristics after either a downmix to the given layout, or the compression defined by the given DRC_set_ID, respectively, or both. These values, if non-zero, must match values in the sample entry. This assumes that there is only one sample entry present in the track.

Loudness values 0 through 255 indicate 6.0 dB LKFS to -57.75 dB LKFS, i.e. in -0.25 dB increments.

The loudness_range is calculated according to the following pseudo-code:

```
UInt8 CompressLoudnessRange(Float32 loudnessRange)
 {
      if (loudnessRange < 0.0f)
            return 0;
      else if(loudnessRange <= 32.0f)
            return (UInt8)(4.0f*loudnessRange + 0.5f);
      else if(loudnessRange <= 70.0f)
            return (UInt8)(2.0f*(loudnessRange – 32.0f) + 0.5f) + 128;
      else if(loudnessRange < 121.0f)
            return (UInt8)(loudnessRange – 70.0f) + 0.5f) + 204;
      else
            return 255;
 }
```

The program loudness is measured using [1] over the associated content; the 'anchor loudness' is the loudness of the anchor content, where what that content is, is determined by the content author; one suitable value (especially for content for which the main content is speech) is 'dialog normal level' or DialNorm.

The method_definition takes one of the following values; all others are reserved:

Index	Meaning	Value Type
1	program loudness (ProgramLoudness as defined in ISO/IEC 23001-8)	Loudness Value
2	anchor loudness (AnchorLoudness as defined in ISO/IEC 23001-8)	Loudness Value
3	loudness range	Loudness Range
4	maximum of the range, i.e. the 95th percentile of the loudness distribution according to [2];	Loudness Value
5	maximum momentary loudness, measured using a 0.4s window (see [3])	Loudness Value
6	maximum short-term loudness, measured using a 3s window (see [3])	Loudness Value
7	mixing level	Sound pressure level

The peak_value field is the true peak of the associated signal before encoding, as defined in ITU BS 1770-3 [1], in decibels relative to 100% full scale. It is represented as a signed value with 8 fractional bits; formally

$$L_{TP} = m(-1)^S 2^{-8}$$

where s is the sign bit and m is the 15 magnitude bits. The value of 0xFFFF FFFF is reserved and means "unknown value".

[[Ed: proposed but not yet fully specified or agreed]] The sound pressure level follows ATSC and is coded as a 5-bit code padded on the left with zero bits to 8 bits. This 5-bit code indicates the

absolute acoustic sound pressure level of an individual channel during the final audio mixing session. The 5-bit code represents a value in the range 0 to 31. The peak mixing level is 80 plus the value of mixlevel dB SPL, or 80 to 111 dB SPL. The peak mixing level is the acoustic level of a sine wave in a single channel whose peaks reach 100 percent in the PCM representation. The absolute SPL value is typically measured by means of pink noise with an RMS value of -20 or -30 dB with respect to the peak RMS sine wave level.

The ***measurement_system*** field takes one of the following values (all other values are reserved):

0: Unknown/other
1: EBU R-128
2: ITU-R BS.1770-3
3: User
4: Expert/panel

The value ‘User’ above indicates that an individual user has marked this content with what they perceive the loudness to be. Having this field allows a piece of content to be moved between the user’s playback devices and the user’s determination to persist.

Similarly, in some circumstances it may be advantageous to use a subjective loudness measure, as determined by an expert, panel, or similar method.

The ***reliability*** field takes one of the following values (all other values are reserved):

0: Reliability is unknown
1: Value is reported/imported but unverified
2: Value is a ‘not to exceed’ ceiling
3: Value is measured and accurate

2.4.3 Putting it all together

Now we can build the new AudioSampleEntry:

```
      // Audio Sequences

class AudioSampleEntry(codingname) extends SampleEntry (codingname){
      unsigned int(16) entry_version;     // must be 1,
                                                        // and must be in an
stsd with version ==1
      const unsigned int(16)[3] reserved = 0;
      template unsigned int(16) channelcount;   // must be correct
      template unsigned int(16) samplesize = 16;
      unsigned int(16) pre_defined = 0;
      const unsigned int(16) reserved = 0 ;
      template unsigned int(32) samplerate = 1<<16;
      // optional boxes follow
      SamplingRateBox();
      ChannelLayout();
      // we permit any number of DownMix or DRC boxes:
      DownMixInstruction() [];
      DRCCoefficients() [];
      DRCInstruction() [];
      Box ();           // further boxes as needed
}
```

And finally the box that goes in the user-data.

```
aligned(8) class LoudnessBox extends Box('linf')
      loudness                LoudnessInfo[];         // a set of one or more
loudness boxes
      albumLoudness     AlbumLoudnessInfo[];    // if applicable
}
```

2.5 Open Questions

1. We need to make sure that this design applies to other codecs as well.
2. We need to make sure that this design can be replicated in other transport environments (notably MPEG-2, and maybe LATM etc.)
3. Is the division, and consequent association, between what's in the setup information and what's in the bit-stream in the right place, clear, and workable in general (notably for DRC)?

3 References

[1] ITU-R, "Recommendation ITU-R BS.1770-3. Algorithm to measure audio programme loudness and true-peak audio level," 08/2012.

[2] EBU, "Loudness Range:A measure to supplement loudness normalisationin accordance with EBU R 128," EBU-Tech 3342, Geneva, August 2011

[3] EBU, "Loudness Metering: EBU mode metering to supplement loudness normalization in accordance with EBU R128", EBU – Tech 3341.

[4] ETSI, "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream," ETSI TS 101 154, 2011.

[5] ATSC "ATSC Recommended Practice: Techniques for Establishing and Maintaining Audio

Loudness for Digital Television," Document A/85:2011, 25 July 2011

[6] Dolby, "Dolby Metadata Guide", Issue 3.