INTERNATIONAL STANDARD

ISO/IEC 23003-3

First edition
2012-04-01

AMENDMENT 3
2016-08-01

Information technology — MPEG audio technologies —

Part 3: Unified speech and audio coding

AMENDMENT 3: Support of MPEG-D DRC, audio pre-roll and immediate play-out frame

Technologies de l'information — Technologies audio MPEG —

Partie 3: Discours unifié et codage audio

AMENDEMENT 3: Support de DRC MPEG-D, message préliminaire audio et cadre de lecture immédiat

Reference number
ISO/IEC 23003-3:2012/Amd.3:2016(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

Amendment 3 to ISO/IEC 23003-3:2012 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — MPEG audio technologies —

Part 3:
Unified speech and audio coding

AMENDMENT 3: Support of MPEG-D DRC, audio pre-roll and immediate play-out frame

Page 1, Normative references

Add the following reference:

ISO/IEC 23003-4, *Information technology — MPEG audio technologies — Part 4: Dynamic Range Control*

Page 4, 4.4

Add new subclause at the end of 4.4:

4.4.1 Decoder behaviour

4.4.1.1 General decoding process

The decoder shall operate in such a way that the decoding of one access unit shall always and immediately produce one full composition unit of audio signal data (one audio frame with outputFrameLength number of samples).

The decoder shall not discard any audio samples. In particular the decoder shall make no assumptions about encoder delay and shall also not attempt to compensate assumed encoder processing delay by removing audio samples from the composition unit buffer.

Discarding of audio samples due to the presence of an EditListBox as described in Annex F is not part of the normative USAC decoder but shall be applied by the MPEG-4 Systems infrastructure.

4.4.1.2 Initialization and re-initialization of the USAC decoder

Upon (re-) initialization all decoder internal signal buffers shall be set to zero.

Due to the initialized state of the decoder internal buffers, the decoder output may contain “start-up samples” when decoding the first access units of a given compressed data stream.

These start-up samples are samples that do not have a direct relation to the audio input data and are typically zero-valued and may be discarded by the Systems infrastructure.

The number of start-up samples to be discarded may for example be transmitted by means of the media_time field in the EditListbox in an ISO Base Media file format environment. Note that this must be done by the encoder.

If a given USAC decoder implementation produces more than the minimum number of start-up samples (i.e. it creates additional decoder delay), the number of additional samples must be reported by the decoder to the Systems infrastructure. Systems infrastructure shall then correctly apply delay compensation or time-alignment.

4.4.1.3 Decoding process of access unit with audio pre-roll

The decoding process of access units with embedded audio pre-roll frames is identical to the above description.

The presence of audio pre-roll in the first access unit prepares the decoder internal signal buffers. This allows an encoder to produce a compressed data stream, that will cause the decoder output buffer to contain less or no start-up samples.

The decoding description when changing from one configuration to another while employing audio pre-roll is described in 7.18.3.3.

If a given decoder implementation produces additional start-up samples (additional decoder delay), then the flushing of the old configuration (FlushDecoder()) shall be increased by the same amount of samples. The signal crossfade must be delayed accordingly. The decoder must ensure that the number of additional start-up samples (additional decoder delay) does not change when switching to another stream in the adaptation set.

Page 11, 4.5.3

Add the following paragraph at the end of 4.5.3:

Furthermore the following requirements apply:

— The number of pre-roll frames, numPreRollFrames, in an AudioPreRoll() extension payload shall not exceed 3.

— Decoders conforming to the Baseline USAC profile shall support the full decoding and correct handling of the AudioPreRoll() extension.

NOTE The number of pre-roll frames required for seamless operation of the audio codec may be lower than the above mentioned number. See B.26 for encoder implementation guide lines.

Page 12, Clause 4

Add new subclause at the end of Clause 4:

4.6 Combination of USAC with MPEG-D DRC

The output of the USAC decoder can be further processed by MPEG-D DRC (ISO/IEC 23003-4). If the SBR tool in USAC is active, a USAC decoder can typically be efficiently combined with a subsequent MPEG-D DRC decoder by connecting them in the QMF domain in the same way as it is described in ISO/IEC 23003-4. If a connection in the QMF domain is not possible they shall be connected in the time domain.

The MPEG-D DRC payload shall be embedded into a USAC bitstream by means of the usacExtElement mechanism, with usacExtElementType of type ID_EXT_ELE_UNI_DRC. The loudness metadata shall be embedded by means of the usacConfigExt mechanism with usacConfigExtType of type ID_CONFIG_EXT_LOUDNESS_INFO. The time-alignment between the USAC data and the MPEG-D DRC data assumes the most efficient connection between the USAC decoder and the MPEG-D DRC decoder. If the SBR tool in USAC is active, the most efficient connection is in the QMF domain. Otherwise, the most efficient connection is in the time domain. The DRC tool is operated in regular delay mode and the DRC frame size has the same duration as the USAC frame size. The same holds for the DRC sampling rate, which is synchronized to the USAC sampling rate.

The time resolution of the DRC tool is specified by *deltaTmin* in units of the audio sample interval. It is calculated as specified in ISO/IEC 23003-4. Specific values are provided here as examples based on the following formula:

$$\text{deltaTmin} = 2^{M}$$

The applicable exponent M is found by looking up the audio sample rate range that fulfils:

$$f_{s,\min} \le f_s < f_{s,\max}$$

Table — AMD3.1 — Lookup table for the exponent *M*

***fs*,min [Hz]**	***fs*,max [Hz]**	***M***
8000	16000	3
16000	32000	4
32000	64000	5
64000	128000	6

Given the codec frame size N_{Codec} (==outputFrameLength), the DRC frame size in units of DRC samples at a rate of *deltaTmin* is:

$$N_{DRC} = N_{Codec}\,2^{-M}$$

For USAC, MPEG-D DRC offers mandatory decoding capability of up to four DRC subbands using the time-domain DRC filter bank. More DRC subbands can be supported by operating in the QMF-domain. DRC sets that contain more than four DRC subbands must contain gain sequences that are all aligned with the QMF-domain used for SBR. If the SBR tool in USAC is active, MPEG-D DRC shall always operate in the QMF-domain. The gain sequences are all aligned with the QMF domain in that case.

If no additional filter bank is required for the application of multiband DRC gains, MPEG-D DRC doesn't introduce any additional decoding delay.

The drcLocation parameter shall be encoded according to Table AMD3.2.

Table — AMD3.2 — Encoding of drcLocation parameter

drcLocation n	**Payload**
1	uniDrcConfig() / uniDrcGain() (see ISO/IEC 23003-4)
2	*reserved*
3	*reserved*
4	*reserved*

Page 16, Table 14

Replace Table 14 with the following table:

Table 14 — Syntax of UsacExtElementConfig()

Syntax	No. of bits	Mnemonic
UsacExtElementConfig()		
{		
usacExtElementType = escapedValue(4,8,16);		
usacExtElementConfigLength = escapedValue(4,8,16);		
usacExtElementDefaultLengthPresent;	**1**	**uimsbf**
if (usacExtElementDefaultLengthPresent) {		
usacExtElementDefaultLength = escapedValue(8,16,0) + 1;		
} else {		
usacExtElementDefaultLength = 0;		
}		
usacExtElementPayloadFrag;	**1**	**uimsbf**
switch (usacExtElementType) {		
case ID_EXT_ELE_FILL:		
break;		
case ID_EXT_ELE_MPEGS:		
SpatialSpecificConfig();		
break;		
case ID_EXT_ELE_SAOC:		
SaocSpecificConfig();		
break;		
case ID_EXT_ELE_AUDIOPREROLL:		
/* No configuration element */		
break;		
case ID_EXT_ELE_UNI_DRC:		
uniDrcConfig();		
break;		
default:	NOTE	
while (usacExtElementConfigLength--) {		
tmp;	**8**	**uimsbf**
}		
break;		
}		
}		
NOTE: The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions.		

Page 16, Table 15

Replace Table 15 with the following table:

Table 15 — Syntax of UsacConfigExtension()

Syntax	No. of bits	Mnemonic
UsacConfigExtension()		
{		
numConfigExtensions = escapedValue(2,4,8) + 1;		
for (confExtIdx=0; confExtIdx<numConfigExtensions; confExtIdx++) {		
usacConfigExtType[confExtIdx] = escapedValue(4,8,16);		
usacConfigExtLength[confExtIdx] = escapedValue(4,8,16);		
switch (usacConfigExtType[confExtIdx]) {		
case ID_CONFIG_EXT_FILL:		
while (usacConfigExtLength[confExtIdx]--) {		
fill_byte[i]; /* should be '10100101' */	**8**	**uimsbf**
}		
break;		
case ID_CONFIG_EXT_LOUDNESS_INFO:		
loudnessInfoSet()		
break;		
default:		
while (usacConfigExtLength[confExtIdx]--) {		
tmp;	**8**	**uimsbf**
}		
break;		
}		
}		
}		

Page 50, Clause 5

Add new subclause at the end of Clause 5:

5.3.5 Payload of extension elements

Table — AMD3.3 — Syntax of AudioPreRoll()

Syntax	No. of bits	Mnemonic
AudioPreRoll()		
{		
configLen = escapedValue(4,4,8);	**4..16**	
Config()	**8*configLen**	
applyCrossfade;	**1**	**bool**
reserved;	**1**	**bool**
numPreRollFrames = escapedValue(2,4,0);	**2..6**	
for (frameIdx=0; frameIdx < numPreRollFrames; ++frameIdx) {		
auLen = escapedValued(16,16,0)	**16..32**	**uimsbf**
AccessUnit()	**8*auLen**	
}		
}		

Page 58, Table 73

Replace Table 73 with the following table:

Table 73 — Value of usacExtElementType

usacExtElementType	Value
ID_EXT_ELE_FILL	0
ID_EXT_ELE_MPEGS	1
ID_EXT_ELE_SAOC	2
ID_EXT_ELE_AUDIOPREROLL	3
ID_EXT_ELE_UNI_DRC	4
/* reserved for ISO use */	5-127
/* reserved for use outside of ISO scope */	128 and higher
NOTE Application-specific usacExtElementType values are mandated to be in the space reserved for use outside of ISO scope. These are skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions.	

Page 58, Table 74

Replace Table 74 with the following table:

Table 74 — Value of usacConfigExtType

usacConfigExtType	Value
ID_CONFIG_EXT_FILL	0
/* reserved for ISO use */	1
ID_CONFIG_EXT_LOUDNESS_INFO	2
/* reserved for ISO use */	3-127
/* reserved for use outside of ISO scope */	128 and higher

Page 64, Table 81

Replace Table 81 with the following table:

Table 81 — Interpretation of data blocks for USAC extension payload decoding

usacExtElementType	The concatenated usacExtElementSegmentData represents:
ID_EXT_ELE_FIL	Series of **fill_byte**
ID_EXT_ELE_MPEGS	SpatialFrame()
ID_EXT_ELE_SAOC	SaocFrame()
ID_EXT_ELE_AUDIOPREROLL	AudioPreRoll()
ID_EXT_ELE_UNI_DRC	uniDrcGain() as defined in ISO/IEC 23003-4
unknown	unknown data. The data block shall be discarded.

Page 210, Clause 7

Add new subclause at the end of Clause 7:

7.18 Audio Pre-Roll

7.18.1 General

The *AudioPreRoll()* syntax element is used to transmit audio information of previous frames along with the data of the present frame. The additional audio data can be used to compensate the decoder startup delay (pre-roll), thus enabling random access at stream access points (SAP) that make use of *AudioPreRoll()*.

A *UsacExtElement()* with the usacExtElementType of ID_EXT_ELE_AUDIOPREROLL shall be used to transmit the *AudioPreRoll()*.

7.18.2 Semantics

configLen	Size of the configuration syntax element in bytes.
Config()	The decoder configuration syntax element. In the context of this standard this shall be the UsacConfig() as defined in 5.2. The Config() field may be transmitted to be able to respond to changes in the audio configuration (e.g. switching of streams).
applyCrossfade	If this flag is set to 1, a linear crossfade shall be applied in case of configuration change, as defined in 7.18.3.3.

reserved reserved bit shall be zero.

numPreRollFrames The number of pre-roll access units (AUs) transmitted as audio pre-roll data. The reasonable number of AUs depends on the decoder start-up delay.

auLen AU length in bytes.

AccessUnit() The pre-roll AU(s).

NOTE The pre-roll data carried in the extension element may be excluded from buffer requirement restrictions, i. e. the buffer requirements may not be satisfied

In order to use *AudioPreRoll()* for both random access and bitrate adaptation the following restrictions apply:

- The first element of every frame shall be an extension element (UsacExtElement) of type ID_EXT_ELE_AUDIOPREROLL.
- The corresponding UsacExtElement() shall be configured as specified in Table AMD3.4.
- Consequently, if pre-roll data is present, this UsacFrame() shall start with the following bit sequence:
  - "1": usacIndependencyFlag.
  - "1": usacExtElementPresent (referring to audio pre-roll extension element).
  - "0": usacExtElementUseDefaultLength (referring to audio pre-roll extension element).
- If no *AudioPreRoll()* is transmitted, the extension payload shall not be present (usacExtElementPresent = 0).
- The pre-roll frames with index "0" shall be independently decodable, i.e. *usacIndependencyFlag* shall be set to "1".
- In access units that are embedded as pre-roll in an AudioPreRoll() extension the usacExtElementPresent field for extensions of type ID_EXT_ELE_AUDIOPREROLL shall be 0.

Table — AMD3.4 — Setup of UsacExtElementConfig() for AudioPreRoll()

Bitstream Field	Value
usacExtElementType	ID_EXT_ELE_AUDIOPREROLL
usacExtElementConfigLength	0
usacExtElementDefaultLengthPresent	0
usacExtElementPayloadFrag	0

7.18.3 Decoding process

7.18.3.1 General

This section describes the decoding process for both random access/immediate play-out and bitrate adoption scenarios.

7.18.3.2 Random access and immediate play-out

Random access and immediate play-out is possible at every frame that utilizes the *AudioPreRoll*() structure as specified in this subclause. The following pseudo-code describes the decoding process:

```
if(usacIndependencyFlag == 1){
  if(usacExtElementPresent == 1){

    /* In this case usacExtElementUseDefaultLength must be 0! */
    if(usacExtElementUseDefaultLength != 0) goto error;
```



```
      /* Not used */
      getUsacExtElementPayloadLength();

      /* Check for presence of config and re-initialize if necessary */
      int configLen = getConfigLen();
      if(configLen > 0){
        config c = getConfig(configLen);
        ReConfigureDecoder(c);
      }

      /* Get pre-roll AUs and decode, discard output samples */
      int numPreRollFrames = getNumPreRollFrames();
      for(auIdx = 0; auIdx < numPreRollFrames; auIdx++) {
        int auLen = getAuLen();
        AU nextAU = getPreRollAU(auLen);
        DecodeAU(nextAU);
      }
    }
  }
```

/* Internal decoder states are initialized at this point. Continue normal decoding */

7.18.3.3 Bitrate adaption

Bitrate adaption may be utilized by switching between different encoded representations of the same audio content. The *AudioPreRoll*() structure as specified in this subclause may be used for that purpose. The decoding process in case of bitrate adaption is specified by the following pseudo-code:

```
if(usacIndependencyFlag == 1){
  if(usacExtElementPresent == 1{

    /* In this case usacExtElementUseDefaultLength must be 0! */
    if(usacExtElementUseDefaultLength != 0) goto error;

    /* Not used */
    getUsacExtElementPayloadLength();

    int configLen = getConfigLen();
    if(configLen > 0){
      config newConfig = getConfig(configLen);

      /* Configuration did not change, skip AudioPreRoll and continue decoding as normal */
      if(newConfig == currentConfig){
        SkipAudioPreRoll();
        goto finish;
      }

      /* Configuration changed, prepare for bitstream switching*/
      outSamplesFlush = FlushDecoder();
      ReConfigureDecoder(c);

      /* Get pre-roll AUs and decode, discard output samples */
      int numPreRollFrames = getNumPreRollFrames();
      for(auIdx = 0; auIdx < numPreRollFrames; auIdx++) {
        int auLen = getAuLen();
        AU nextAU = getPreRollAU(auLen);
        DecodeAU(nextAU);

      }

      /* Get "regular" AU and decode */
      AU au = UsacFrame();
      outSamplesFrame = DecodeAU(au);

      /* Apply crossfade only on the output samples*/
      If(applyCrossfade) {
       for(i = 0; i < 128; i++){
         outSamples[i] = outSamplesFlush[i] * (1-i/127) +
                 outSamplesFrame[i] * (i/127)
       }
```



```
      } else {
       for(i = 0; i < 128; i++) {
         outSamples[i] = outSamplesFrame[i];
       }
      }
      for(i = 128; i < outputFrameLength; i++){
       outSamples[i] = outSamplesFrame[i];
      }
    }
  }
}
```

If a configuration change is detected by the decoder the following steps shall be applied:

— Flush the internal decoder states and buffers (FlushDecoder()), i.e. decode a hypothetical access unit composed of all zero samples. Store the resulting output samples (outSamplesFlush) in a temporary buffer.

— Re-initialize the decoder with the new configuration (ReConfigureDecoder()).

— Decode all contained pre-roll AUs and discard the resulting output.

— Decode the current AU (UsacFrame()). Store the resulting output samples (outSamplesFrame) in a temporary buffer.

— In case **applyCrossfade** is set to 1 and operates in the time domain, a linear cross-fade of length 128 on outSamplesFlush and outSamplesFrame shall be applied to avoid switching artifacts.

Page 253, Annex B

Add new subclause at the end of Annex B:

B.26 Delay considerations for adaptive streaming

B.26.1 General

The following paragraphs discuss only those potential delay sources in the USAC encoding and decoding process that can be observed at the decoder output and need to be considered there. Delay sources that affect real-time encoding and decoding behaviour are intentionally ignored here (e.g. framing delay, transmission delay, processing time etc.).

B.26.2 Additional encoder delay

The USAC encoder implementation is not normative and thus may introduce further sources of delay, for example additional look-ahead for parameter estimation or delay due to certain framing constraints. In the following this number of additional encoder generated samples of delay shall be called “additional encoder delay”, D_{addenc}. It is important to understand that this “additional encoder delay” is not known to the USAC decoder and depends on the specific encoder implementation. This delay can be completely compensated by the encoder, if it processes sufficient samples before sending the first access unit. Note that this only means that the delay disappears from the decoder output buffer. A theoretical “real-time” end-to-end delay is not affected.

B.26.3 Additional decoder delay

A given USAC decoder implementation may produce additional delay. For example with extension payloads or additional tools whose application is not mandatory (e.g. MPEG Surround extension). In the following this number of additional decoder generated samples of delay shall be called “additional decoder delay”, Dadddec. This delay is decoder specific and cannot be known by the encoder. This delay is not included in a potential media time field in the EditListbox in an ISO Base Media file format environment. Therefore this additional decoder delay must be reported by the decoder to the Systems infrastructure which shall compensate this additional decoder delay.