INTERNATIONAL STANDARD ISO/IEC 14496-3:1999
TECHNICAL CORRIGENDUM 1

Published 2001-08-01

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 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to International Standard ISO/IEC 14496-3:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ICS 35.040 **Ref. No. ISO/IEC 14496-3:1999/Cor.1:2001(E)**

Printed in Switzerland

Throughout the text of ISO/IEC 14496-3:1999 replace all occurrences of "AL PDU" *with* "SL packet" *and all occurrences of* "alPduPayload" *with* "slPacketPayload".

In subpart 1 replace all occurrences of " frameLength " *with* " frameLengthFlag ",

and in subpart 4 replace all occurrences of "FrameLengthFlag" *with* "frameLengthFlag".

In subclause 1.5.1, void tables 1.5.1 to 1.5 .4, and replace with

"

Table 1.5.1 — Audio Profiles

Audio Object Types	Speech Audio Profile	Synthesis Audio Profile	Scalable Audio Profile	Main Audio Profile
Null				
AAC LC			X	X
AAC main				X
AAC SSR				X
AAC LTP			X	X
AAC Scalable			X	X
TwinVQ			X	X
CELP	X		X	X
HVXC	X		X	X
TTSI	X	X	X	X
Main synthetic		X		X
Wavetable synthesis		(subset of Main synthetic)		(subset of Main synthetic)
General MIDI		(subset of Main synthetic)		(subset of Main synthetic)
Algorithmic Synthesis and Audio FX		(subset of Main synthetic)		(subset of Main synthetic)

".

In subclause 1.5.2, add "Audio" *to all profile names.*

In subclause 1.5.2, replace all " Synthesis Audio Profile" *with* " Synthetic Audio Profile".

In subclause 1.5.2.2, replace the first row of Table 1.5.6 – Complexity of Object Types with

"

Object Type	Parameters	PCU (MOPS per channel)	RCU (kWords per channel)	Remarks

".

Replace subclause 1.5.2.2 with

"

- **Levels for Synthetic Audio Profile**

Three levels are defined:

Synthetic Audio 1: All bitstream elements may be used with:

"Low processing" (exact numbers in ISO/IEC 14496-4:2000)

Only core sample rates may be used

No more than one TTSI object

Synthetic Audio 2: All bitstream elements may be used with:

"Medium processing" (exact numbers in ISO/IEC 14496-4:2000).

Only core sample rates may be used.

no more than four TTSI objects.

Synthetic Audio 3: All bitstream elements may be used with:

"High processing" (exact numbers in ISO/IEC 14496-4:2000).

No more than twelve TTSI objects.

- **Levels for Main Audio Profile**

Main Audio Profile contains all natural and synthetic object types. Levels are then defined as a combination of the two different types of levels from the two different metrics defined for natural tools (computation-based metrics) and synthetic tools (macro-oriented metrics).

For Object Types not belonging to the Synthetic Profile four levels are defined:

Natural Audio 1: PCU < 40, RCU < 20

Natural Audio 2: PCU < 80, RCU < 64

Natural Audio 3: PCU < 160, RCU < 128

Natural Audio 4: PCU < 320, RCU < 256

For Object Types belonging to the Synthetic Profile the same three Levels are defined as above, i.e. Synthetic Audio 1, Synthetic Audio 2 and Synthetic Audio 3.

Four Levels are then defined for Main Profile:

Natural Audio 1 + Synthetic Audio 1

Natural Audio 2 + Synthetic Audio 1

Natural Audio 3 + Synthetic Audio 2

Natural Audio 4 + Synthetic Audio 3

".

In subclause 1.5.2, add "Algorithmic synthesis and AudioFX object type "*in Object Type definitions for Audio and in the Profiles and Levels table (Table 1.5.6 Complexity of Object Types).*

Replace Table 1.5.6 with

"

The following table gives complexity estimates for the different object types and Sampling Rate conversion:

Table 1. 5. 6 - Complexity of Object Types and SR conversion

Object Type	Parameters	PCU (MOPS per channel)	RCU (kWords per channel)	Remarks
AAC Main	fs = 48 kHz	5	5	1)
AAC LC	fs = 48 kHz	3	3	1)
AAC SSR	fs = 48 kHz	4	3	1)
AAC LTP	fs = 48 kHz	4	4	1)
AAC Scalable	fs = 48 kHz	5	4	1), 2)
TwinVQ	fs = 24 kHz	2	3	1)
CELP	fs = 8 kHz	1	1	
CELP	fs = 16 kHz	2	1	
CELP	fs = 8/16 kHz (bandwidth scalable)	3	1	
HVXC	fs = 8 kHz	2	1	
TTSI		-	-	4)
General MIDI		4	1	
Wavetable Synthesis	fs = 22.05 kHz	depends on bitstreams (3)	depends on bitstreams (3)	
Main Synthetic		depends on bitstreams (3)	depends on bitstreams (3)	
Algorithmic Synthesis and AudioFX		depends on bitstreams (3)	depends on bitstreams (3)	
Sampling Rate Conversion	rf = 2, 3, 4, 6	2	0.5	

Definitions:

fs = sampling frequency

rf = ratio of sampling rates

Notes -

1) PCU proportional to sampling frequency.

2) Includes core decoder.

3) See ISO/IEC 14496-4:2000.

4) The complexity for speech synthesis is not taken into account.

".

In subclause 1.6.2, replace all "AudioSpecificInfo()" *with* "AudioSpecificConfig()".

To the end of subclause 1.6.2.7, add

"

Payloads that are not byte aligned should be zero-padded at the end for transport schemes which require byte alignment.

".

In subclause 1.6.3.3, replace the table header of Table 1.6.2 with

"

samplingFrequencyIndex	Value

".

Remove subclause 1.A.2.3 "MPEG-4 Audio Transport Stream (MATS)" .

In subclause 2.3.1, replace

"

HVXC Base Layer –Configuration

For HVXC object type in unscalable mode or as the base layer in scalable mode requires the following HvxcSpecificConfig() required:

```
        HvxcSpecificConfig() {
                HVXCconfig();
        }
```

HVXC Enhancement Layer –Configuration

HVXC object type provides a 2kbit/s base layer plus a 2kbit/s enhancement layer scalable mode. In this scalable mode the basic layer configuration must be as follows:

```
        HVXCvarMode = 0                HVXC fixed bit rate
        HVXCrateMode = 0               HVXC 2kbps
```

For the enhancement layer, there is no HvxcSpecificConfig() required:

```
        HvxcSpecificConfig() {
        }
```

"

with

"

The following HvxcSpecificConfig() is required:

```
HvxcSpecificConfig ( ) {
        isBaseLayer                                    1       uimsbf
        if (isBaseLayer) {
                HVXCconfig()
        }
}
```

HVXC object type provides unscalable modes and a 2kbit/s base layer plus a 2kbit/s enhancement layer scalable mode. In this scalable mode the basic layer configuration must be as follows:

```
        HVXCvarMode = 0                HVXC fixed bit rate
        HVXCrateMode = 0               HVXC 2kbps
        isBaseLayer=1                  base layer
```

",

and at the end of subclause 2.4.1, add

"

isBaseLayer A one-bit identifier representing whether the corresponding layer is the base layer (1) or the enhancement layer (0).

".

In subclause 2.5.3.3, add

"

If the pitch modification is controlled by the **pitch** field in the AudioSource BIFS node, the modification factor is:

$$pch\_mod = pitch$$

"

after

"

Pitch modification can be done by dividing pch by pitch modification factor pch_mod:

$$pch = pch / pch\_mod$$

".

In subclause 2.5.5.3, add the sentence

"

If the speed is controlled by the time scaling factor in the **speed** field of the AudioSource BIFS node, the speed change ratio is:

$$spd = 1 / speed$$

"

after

"

where N_1 is the duration of the original speech and N_2 is the duration of the speed controlled speech. Therefore,

$0 \le n < N_1$ and $0 \le m < N_2$.

".

In subclause 3.3, replace the following paragraphs:

"

CelpSpecificConfig()

CELP Base Layer

The CELP core in the unscalable mode or as the base layer in the scalable mode requires the following CelpSpecificConfig():

```
class CelpSpecificConfig (uint(4) samplingFrequencyIndex ) {
        CelpHeader (samplingFrequencyIndex);
}
```

CELP Enhancement Layer

The CELP core is used for both bitrate and bandwidth scalable modes. In the bitrate scalable mode, the enhancement layer requires no CelpSpecificConfig(). In the bandwidth scalable mode, the enhancement layer has the following CelpSpecificConfig():

```
class CelpSpecificConfig() {
        CelpBWSenhHeader();
}
```

"

with

"

CelpSpecificConfig()

The following CelpSpecificConfig() is required:

```
class CelpSpecificConfig (uint(4) samplingFrequencyIndex ) {
    isBaseLayer                1   uimsbf
    if (isBaseLayer) {
        CelpHeader(samplingFrequencyIndex)
    } else {
        isBWSLayer             1   uimsbf
        if (isBWSLayer) {
            CelpBWSenhHeader()
        } else {
            CELP-BRS-id        2   uimsbf
        }
    }
}
```

",

and at the end of subclause 3.3.4, add

"

isBaseLayer see subclause 2.4.1 of subpart 2.

isBWSLayer A one-bit identifier representing whether the corresponding layer is the bandwidth scalable enhancement layer (1) or the bitrate scalable enhancement layer (0).

CELP-BRS-id A two-bit identifier representing the order of the bitrate scalable enhancement layers, where the first enhancement layer has the value of '1'. The value of '0' should not be used.

".

In subclause 4.4.2 replace Table 4.4.28

"

Syntax	No. of bits	Mnemonic
ltp_data()		
{		
ltp_lag	**11**	**uimsbf**
ltp_coef	**3**	**uimsbf**
if(window_sequence==EIGHT_SHORT_SEQUENCE) {		
for (w=0; w<num_windows; w++) {		
ltp_short_used[w]	**1**	**uimsbf**
if (ltp_short_used [w]) {		
ltp_short_lag_present[w]	**1**	**uimsbf**
}		
if (ltp_short_lag_present[w]) {		
ltp_short_lag[w]	**4**	**uimsbf**
}		
}		
} else {		
for (sfb=0; sfb<max_sfb; sfb++) {		
ltp_long_used[sfb]	**1**	**uimsbf**
}		
}		
}		

"

with

"

Syntax	No. of bits	Mnemonic
ltp_data()		
{		
ltp_lag	**11**	**uimsbf**
ltp_coef	**3**	**uimsbf**
if(window_sequence==EIGHT_SHORT_SEQUENCE) {		
for (w=0; w<num_windows; w++) {		
ltp_short_used[w]	**1**	**uimsbf**
if (ltp_short_used [w]) {		
ltp_short_lag_present[w]	**1**	**uimsbf**
if (ltp_short_lag_present[w]) {		
ltp_short_lag[w]	**4**	**uimsbf**
}		
}		
}		
} else {		
for (sfb=0; sfb<max_sfb; sfb++) {		
ltp_long_used[sfb]	**1**	**uimsbf**
}		
}		
}		

".

In subclause 4.4, remove

"

Two types of data are part of the MPEG-4 GA coder syntax. These are

1. Configuration information
2. Actual Payload

The payload is intended to be transported via the MPEG-4 Systems layer. These data contain all information variing on a frame to frame basis, and therefore carry the actual audio information.

The Configuration information is also transported via MPEG-4 systems. These elements contain configuration information, which is necessary for the decoding process and parsing of the Payload. However, an update is only necessary if there are changes in the configuration.

The configuation information and the payload are abstract elements which define all information for the decoding and parsing of the bitstream. However, for real applications these streams need a transport layer which cares for the delivery of these elements. Normally, this transport mechanism will be handled by MPEG-4 Systems. However, the interface format streams defined in the Annex A of subpart 1 define a simple way of multiplexing the header and the raw data streams.

".

In subclause 4.4.1, 4.5.1, 4.5.2 and 4.6.14, replace

"GASpecificConfiguration()", "GA_SpecificConfig", *and* "GA_SpecificConfig()"

with

"GASpecificConfig()".

In subclause 4.4.1, replace the heading

"GA Specific configuration"

with

"Decoder configuration (GASpecificConfig)".

In table 4.15 (Syntax of aac_scalable_main_header()) in subclause 4.4.2,

replace the term "tvq_layer_pesent"

with

"tvq_layer_present".

In subclause 4.5.1.1, replace the description

"

ExtensionFlag: Set to '0' in MPEG-4 Phase 1. Set to ' 1' in MPEG-4 Phase 2.

"

with

"

ExtensionFlag: Shall be '0' for audio object types 1, 2, 3, 4, 6, 7. Shall be '1' for audio object types 17, 19, 20, 21, 22, 23.

".

At the end of subclause 4.5.1.1, add

"

Restriction:

An MPEG-4 Audio decoder is only required to follow the Program Configuration Element in GASpecificConfig(). The decoder shall ignore any Program Configuration Elements that may occur in raw data blocks. PCEs transmitted in raw data blocks cannot be used to convey decoder configuration information.

",

and in subclause 4.5.1.2.1, replace

"

For more complicated configurations a **Program Configuration Element** (PCE) is defined. There are 16 available PCE' s, and each one can specify a distinct program that is present in the raw data stream. All available PCE' s within a raw_data_block must come before all other syntactic elements. Programs may or may not share audio syntactic elements, for example, programs could share a channel_pair_element and use distinct coupling channels for voice over in different languages. A given program configuration element contains information pertaining to only one program out of many that may be included in the raw data stream. Included in the PCE are „list of front channels", again using the rule center outwards, left before right. In this list, a center channel SCE, if any, must come first, and any other SCE’ s must appear in pairs, constituting an LR pair. If only two SCE’ s are specified, this signifies one LR stereophonic pair.

After the list of front channels, there is a list of "side channels" consisting of CPE’ s, or of pairs of SCE’ s. These are listed in the order of front to back. Again, in the case of a pair of SCE’ s, the first is a left channel, the second a right channel.

After the list of side channels, a list of back channels is available, listed from outside in. Any SCE’ s except the last

SCE must be paired, and the presence of exactly two SCE’ s (alone or preceeded by a CPE) indicates that the two

SCE’ s are Left and Right Rear center, respectively.

The configuration indicated by the PCE takes effect at the raw_data_block containing the PCE. The number of front, side and back channels as specified in the PCE must be present in that block and all subsequent raw_data_blocks until a raw_data_block containing a new PCE is transmitted.

Other elements are also specified. A list of one or more LFE’ s is specified for application to this program. A list of one or more CCE’ s is also provided, in order to allow for dialog management as well as different intensity coupling streams for different channels using the same main channels. A list of data streams associated with the program can also associate one or more data streams with a program. The program configuration element also allows for the specification of one monophonic and one stereophonic simulcast mixdown channel for a program.

Note that the MPEG-4 Systems standard supports alternate methods of simulcast.

The PCE element is not intended to allow for rapid program changes. At any time when a given PCE, as selected by its element_instance_tag, defines a new (as opposed to repeated) program, the decoder is not obliged to provide audio signal continuity.

″

with

″

For more complicated configurations a **Program Configuration Element** (PCE) is defined. The same restrictions apply with respect to the PCE as defined in ISO/IEC 14496-3:1999. However, an MPEG-4 decoder is only required to parse PCEs in raw_data_blocks(), without interpreting them. Only the PCE provided within GASpecificConfig() describes the decoder configuration for the elementary stream under consideration. This implies that only one program can be configured at a certain time.

″.

In subclause 4.5.1.1, replace

″

If the sampling rate is not one of the rates listed in the right column in the table below, the sampling frequency dependent tables (code tables, scale factor band tables etc.) must be deduced in order for the bit stream to be parsed. Since a given sampling frequency is associated with only one sampling frequency table, and since maximum flexibility is desired in the range of possible sampling frequencies, the following table shall be used to associate an implied sampling frequency with the desired sampling frequency dependent tables. However, there is one exception to this rule, which is described in subclause 4.6.13.1 for Table 4.6.12.

Table 4.5.1

Frequency range	use tables for sampling frequency
f >= 92017	96000
92017 > f >= 75132	88200
75132 > f >= 55426	64000
55426 > f >= 46009	48000
46009 > f >= 37566	44100
37566 > f >= 27713	32000
27713 > f >= 23004	24000
23004 > f >= 18783	22050
18783 > f >= 13856	16000
13856 > f >= 11502	12000
11502 > f >= 9391	11025
9391 > f	8000

″

with

″

If the sampling rate is not one of the rates listed in the right column in Table 4.5.1, the sampling frequency dependent tables (code tables, scale factor band tables etc.) must be deduced in order for the bit stream to be parsed. Since a given sampling frequency is associated with only one sampling frequency table, and since maximum flexibility is desired in the range of possible sampling frequencies, the following table shall be used to associate an implied sampling frequency with the desired sampling frequency dependent tables. However, there is one exception to this rule, which is described in subclause 4.6.13.1 for Table 4.6.12.

Table 4.5.1 Sampling frequency mapping

Frequency range (in Hz)	Use tables for sampling frequency (in Hz)
f >= 92017	96000
92017 > f >= 75132	88200
75132 > f >= 55426	64000
55426 > f >= 46009	48000
46009 > f >= 37566	44100
37566 > f >= 27713	32000
27713 > f >= 23004	24000
23004 > f >= 18783	22050
18783 > f >= 13856	16000
13856 > f >= 11502	12000
11502 > f >= 9391	11025
9391 > f	8000

If a certain sampling frequency dependent table stated in the right column of Table 4.5.1 is not defined, the nearest defined table shall be used.

″.

In subclause 4.5.2.1.1, replace

"

raw_data_block(:) block of raw data that contains audio data for a time period of 1024 or 960 samples, related information and other data. There are 8 bitstream elements, identified as bitstream element id_syn_ele. The audio elements in one raw data stream and one raw data block must have one and only one sampling rate. In the raw data block, several instances of the same id_syn_ele may occur, but each such instance of an id_syn_ele except for a data_stream_element must have a different 4-bit element_instance_tag. Therefore, in one raw data block, there can be from 0 to at most 16 of any id_syn_ele. The exceptions to this are the data_stream_element, the fill_element and the terminator element. If multiple data stream elements occur which have unique element_instance_tags then they are part of distinct data streams. If multiple data stream elements occur which have the same element_instance_tag then they are part of the same data stream. The fill_element has no element_instance_tag (since the content does not require subsequent reference) and can occur any number of times. The terminator element has no element_instance_tag and must occur exactly once, as it marks the end of the raw_data_

"

with

"

raw_data_block(): block of raw data that contains audio data for a time period of 1024 or 960 samples, related information and other data. There are 8 syntactic elements, identified as syntactic element id_syn_ele. The audio elements in one raw data block must have one and only one sampling rate. In the raw data block, several instances of the same id_syn_ele may occur, but each such instance of an id_syn_ele except for a data_stream_element must have a different 4-bit element_instance_tag. Therefore, in one raw data block, there can be from 0 to at most 16 of any id_syn_ele. The exceptions to this are the data_stream_element, the fill_element and the terminator element. If multiple data stream elements occur which have unique element_instance_tags then they are part of distinct data streams. If multiple data stream elements occur which have the same element_instance_tag then they are part of the same data stream. The fill_element has no element_instance_tag (since the content does not require subsequent reference) and can occur any number of times. The terminator element has no element_instance_tag and must occur exactly once, as it marks the end of the raw_data_block**.**

".

In subclause 4.5.2.2.4, replace

"

For all scale factor bands where M/S or Intensity coding is selected, the M''-Signal is calculated by adding M'' and M' (The restrictions given in subclause 5.2.2.7 have to be followed which prohibit the addition under certain circumstances).

"

with

"

For all scale factor bands where M/S coding is selected, the M-Signal is calculated by adding M'' and M' (The restrictions given in subclause 5.2.2.7 have to be followed which prohibit the addition under certain circumstances).

".

In Table 5.7 (raw 1, column 1) in subclause 4.5.2.2.5.2, replace

"Sampling rate (Hz) " *with* " Sampling rate (kHz) ".