INTERNATIONAL STANDARD

ISO/IEC 14496-3

Second edition
2001-12-15

AMENDMENT 2
2004-08-01

Information technology — Coding of audio-visual objects —

Part 3: **Audio**

AMENDMENT 2: Parametric coding for high-quality audio

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

AMENDEMENT 2: Codage paramétrique pour le codage audio de haute qualité

Reference number
ISO/IEC 14496-3:2001/Amd.2:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to ISO/IEC 14496-3:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Introduction

This document specifies the second Amendment to ISO/IEC 14496-3:2001. The document specifies the normative syntax of the 'Parametric Coding for High Quality Audio' tool SSC and the decoding process. An informative encoder description is given as well.

Information technology — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 2: Parametric coding for high-quality audio

In ISO/IEC 14496-3:2001, Introduction, add:

MPEG-4 SSC, (SinuSoidal Coding) is a parametric coding tool that is capable of full bandwidth high quality audio coding. The coding tool dissects a monaural or stereo audio signal into a number of different objects that each can be parameterized efficiently and encoded at a low bit-rate. These objects are transients: representing dynamic changes in the temporal domain, sinusoids: representing deterministic components, and noise: representing components that do not have a clear temporal or spectral localisation. The fourth object, that is only relevant for stereo input signals, captures the stereo image. As the signal is represented in a parametric domain, independent, high quality pitch and tempo scaling are possible at low computational cost.

Amendment subpart 1

In Part 3: Audio, Subpart 1, in subclause 1.3 Terms and Definitions, add:

270. **SSC**: SinuSoidal Coding.

and increase the index-number of subsequent entries.

In Part 3: Audio, Subpart 1, in subclause 1.5.1.1 Audio object type definition, replace table 1.1 with the table below:

Table 1.1 – Audio object definition

Tools/ Modules Audio Object Type	gain control	block switching	window shapes - standard	window shapes – AAC LD	filterbank - standard	filterbank – SSR	TNS	LTP	intensity	coupling	MPEG-2 prediction	PNS	MS	SIAQ	FSS	upsampling filter tool	quantisation&coding - AAC	quantisation&coding - TwinVQ	quantisation&coding - BSAC	AAC ER Tools	ER payload syntax	EP Tool 1)	CELP	Silence Compression	HVXC	HVXC 4kbs VR	SA tools	SASBF	MIDI	HILN	TTSI	SBR	SSC	Remark	Object Type ID
Null																																			0
AAC main		X	X		X		X		X	X	X	X	X				X																	2)	1
AAC LC		X	X		X		X		X	X		X	X				X																		2
AAC SSR	X	X	X			X	X		X	X		X	X				X																		3
AAC LTP		X	X		X		X	X	X	X		X	X				X																	2)	4
SBR																																X			5
AAC Scalable		X	X		X		X	X	X			X	X	X	X	X	X																	6)	6
TwinVQ		X	X		X		X	X					X					X																	7
CELP																							X												8
HVXC																									X										9
(Reserved)																																			10
(Reserved)																																			11
TTSI																															X				12
Main synthetic																											X	X	X					3)	13
Wavetable synthesis																												X	X					4)	14
General MIDI																													X						15
Algorithmic Synthesis and Audio FX																											X								16
ER AAC LC		X	X		X		X		X			X	X				X			X	X	X													17
(Reserved)																																			18
ER AAC LTP		X	X		X		X	X	X			X	X				X			X	X	X												5)	19
ER AAC scalable		X	X		X		X		X			X	X	X	X	X	X			X	X	X												6)	20
ER TwinVQ		X	X		X		X						X					X			X	X													21
ER BSAC		X	X		X		X		X			X	X						X		X	X													22
ER AAC LD				X	X		X	X	X			X	X				X			X	X	X													23
ER CELP																					X	X	X	X											24
ER HVXC																					X	X			X	X									25
ER HILN																					X	X								X					26
ER Parametric																					X	X			X	X				X					27
SSC																																	X		28
(Reserved)																																			29
(Reserved)																																			30
(Reserved)																																			31

In Part 3: Audio, Subpart 1, replace Table 1.2 (Audio Profiles definition) with the following table:

Table 1.2 – Audio Profiles definition

Audio Object Type	Main Audio Profile	Scalable Audio Profile	Speech Audio Profile	Synthetic Audio Profile	High Quality Audio Profile	Low Delay Audio Profile	Natural Audio Profile	Mobile Audio Internet-working Profile	AAC Profile	High Efficiency AAC Profile	Object Type ID
Null											0
AAC main	X						X				1
AAC LC	X	X			X		X		X	X	2
AAC SSR	X						X				3
AAC LTP	X	X			X		X				4
SBR										X	5
AAC Scalable	X	X			X		X				6
TwinVQ	X	X					X				7
CELP	X	X	X		X	X	X				8
HVXC	X	X	X			X	X				9
(reserved)											10
(reserved)											11
TTSI	X	X	X	X		X	X				12
Main synthetic	X			X							13
Wavetable synthesis											14
General MIDI											15
Algorithmic Synthesis and Audio FX											16
ER AAC LC					X		X	X			17
(reserved)											18
ER AAC LTP					X		X				19
ER AAC Scalable					X		X	X			20
ER TwinVQ							X	X			21
ER BSAC							X	X			22
ER AAC LD						X	X	X			23
ER CELP					X	X	X				24
ER HVXC						X	X				25
ER HILN							X				26
ER Parametric							X				27
SSC											28
(reserved)											29
(reserved)											30
(reserved)											31

In Part 3: Audio, Subpart 1, in subclause 1.6.2.1 AudioSpecificConfig, replace table 1.8 with the table below:

Table 1.8 – Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic
AudioSpecificConfig ()		
{		
audioObjectType;	**5**	**uimsbf**
samplingFrequencyIndex;	**4**	**uimsbf**
if (samplingFrequencyIndex==0xf)		
samplingFrequency;	**24**	**uimsbf**
channelConfiguration;	**4**	**uimsbf**
sbrPresentFlag = -1;		
if (audioObjectType == 5) {		
extensionAudioObjectType = audioObjectType;		
sbrPresentFlag = 1;		
extensionSamplingFrequencyIndex;	**4**	**uimsbf**
if (extensionSamplingFrequencyIndex==0xf)		
extensionSamplingFrequency;	**24**	**uimsbf**
audioObjectType;	**5**	**uimsbf**
}		
else {		
extensionAudioObjectType = 0;		
}		
if (audioObjectType == 1 || audioObjectType == 2 ||		
audioObjectType == 3 || audioObjectType == 4 ||		
audioObjectType == 6 || audioObjectType == 7)		
GASpecificConfig();		
if (audioObjectType == 8)		
CelpSpecificConfig();		
if (audioObjectType == 9)		
HvxcSpecificConfig();		
if (audioObjectType == 12)		
TTSSpecificConfig();		
if (audioObjectType == 13 || audioObjectType == 14 ||		
audioObjectType == 15 || audioObjectType==16)		
StructuredAudioSpecificConfig();		
if (audioObjectType == 17 || audioObjectType == 19 ||		
audioObjectType == 20 || audioObjectType == 21 ||		
audioObjectType == 22 || audioObjectType == 23)		
GASpecificConfig();		
if (audioObjectType == 24)		
ErrorResilientCelpSpecificConfig();		
if (audioObjectType == 25)		
ErrorResilientHvxcSpecificConfig();		
if (audioObjectType == 26 || audioObjectType == 27)		
ParametricSpecificConfig();		
if (audioObjectType == 17 || audioObjectType == 19 ||		
audioObjectType == 20 || audioObjectType == 21 ||		
audioObjectType == 22 || audioObjectType == 23 ||		
audioObjectType == 24 || audioObjectType == 25 ||		
audioObjectType == 26 || audioObjectType == 27) {		
epConfig;	**2**	**uimsbf**
if (epConfig == 2 || epConfig == 3) {		
ErrorProtectionSpecificConfig();		
}		
if (epConfig == 3) {		

```
            directMapping;                                        1        uimsbf
            if ( ! directMapping ) {
                /* tbd */
            }
        }
    }
    if (  audioObjectType == 28)
        SSCSpecificConfig();

    if (  extensionAudioObjectType != 5 &&
        bits_to_decode() >= 16 ) {
        syncExtensionType;                                        11       bslbf
        if (syncExtensionType == 0x2b7) {
            extensionAudioObjectType;                             5        uimsbf
            if ( extensionAudioObjectType == 5 ) {
                sbrPresentFlag;                                   1        uimsbf
                If (sbrPresentFlag == 1) {
                    extensionSamplingFrequencyIndex;              4        uimsbf
                    if ( extensionSamplingFrequencyIndex == 0xf )
                        extensionSamplingFrequency;               24       uimsbf
                }
            }
        }
    }
}
```

In Part 3: Audio, Subpart 1, in subclause 1.6.2.2.1 Overview, replace table 1.9 by the following table:

Table 1.9 – Audio Object Types

Audio Object Type	Object Type ID	definition of elementary stream payloads and detailed syntax	Mapping of audio payloads to access units and elementary streams
AAC MAIN	1	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC LC	2	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC SSR	3	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC LTP	4	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
SBR	5	ISO/IEC 14496-3 subpart 4	
AAC scalable	6	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.3
TwinVQ	7	ISO/IEC 14496-3 subpart 4	
CELP	8	ISO/IEC 14496-3 subpart 3	
HVXC	9	ISO/IEC 14496-3 subpart 2	
TTSI	12	ISO/IEC 14496-3 subpart 6	
Main synthetic	13	ISO/IEC 14496-3 subpart 5	
Wavetable synthesis	14	ISO/IEC 14496-3 subpart 5	
General MIDI	15	ISO/IEC 14496-3 subpart 5	
Algorithmic Synthesis and Audio FX	16	ISO/IEC 14496-3 subpart 5	
ER AAC LC	17	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER AAC LTP	19	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER AAC scalable	20	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER Twin VQ	21	ISO/IEC 14496-3 subpart 4	
ER BSAC	22	ISO/IEC 14496-3 subpart 4	
ER AAC LD	23	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER CELP	24	ISO/IEC 14496-3 subpart 3	
ER HVXC	25	ISO/IEC 14496-3 subpart 2	
ER HILN	26	ISO/IEC 14496-3 subpart 7	
ER Parametric	27	ISO/IEC 14496-3 subpart 2 and 7	
SSC	28	ISO/IEC 14496-3 subpart 8	

Create Part 3: Audio, Subpart 8:

Subpart 8: Technical description of parametric coding for high quality audio

8.1 Scope

This part of ISO/IEC 14496 describes the MPEG-4 audio parametric coding scheme for compression of high quality audio. The short name is SSC (SinuSoidal Coding). At bit-rates around 24 kbit/s stereo and at a sampling rate of 44.1 kHz, the SSC coding scheme offers a quality that is interesting for a number of applications.

SSC employs four different tools that together parameterize an audio signal. These tools consist of transient modelling, sinusoidal modelling, noise modelling and stereo image modelling. One of the distinctive features of SSC is that it provides decoder support for independent tempo and pitch scaling at hardly any additional complexity.

Transient tool

The transient tool captures the highly dynamic events of the audio input signal. These events are efficiently modelled by means of a limited number of sinusoids that are shaped by means of an envelope.

Sinusoidal tool

The sinusoidal tool captures the deterministic events of the audio input signal. The slowly varying nature of sinusoidal components for typical audio signals is exploited by linking sinusoids over consecutive frames. By means of differential coding, the frequency, amplitude and phase parameters can be efficiently represented.

Noise tool

The noise tool captures the stochastic or non-deterministic events of the audio input signal. In the decoder, a white noise generator is used as excitation. A temporal and spectral envelope is applied to control the temporal and spectral properties of the noise in the audio signal.

Parametric stereo coding tool

The parametric stereo coding tool is able to capture the stereo image of the audio input signal into a limited number of parameters, requiring only a small overhead ranging from a few kbit/s for medium quality, up to about 9 kbit/s for higher quality. Together with a monaural downmix of the stereo input signal generated by the parametric stereo coding tool, the parametric stereo decoding tool is able to regenerate the stereo signal. It is a generic tool that in principle can operate in combination with any monaural coder. In Annex A of this document a normative description of the combination of HE-AAC with the parametric stereo coding tool is provided. SSC can also operate in dual mono mode. In that case the parametric stereo coding tool is not employed. The parametric stereo tool is intended for low bit-rates.

8.2 Terms and definitions

8.2.1
Frame
Basic unit that can be decoded on itself (file header information is required for general decoder settings).

8.2.2
Laguerre filter
Filter structure used in the noise analysis and synthesis.

8.2.3
Audio frame
Contains all data to decode an SSC-coded frame as a stand-alone unit (file header information is required for general decoder settings). For audio frames with refresh_sinusoids==%1 and refresh_noise==%1 the complete frame can always be reconstructed; otherwise it is possible in the case of random access that parts of the signal cannot be reconstructed (e.g. sinusoidal continuations, noise).

8.2.4
Sub-frame
Fine granularity within a frame.

8.2.5
f_s
The sampling frequency in Hertz.

8.2.6
Segment
An interval of samples that can be synthesized on the basis of the parameters that correspond to a sub-frame. The segment size is 2*S (see Table 8.11).

8.2.7
Window
A function that is used to weigh synthesized samples within a segment such that a valid synthesis is obtained.

8.2.8
LSF
Line Spectral Frequency.

8.2.9
Overlap and add
An additive method of combining overlapping intervals during signal synthesis.

8.2.10
Linking process
A method to keep track of sinusoidal components over time.

8.2.11
birth
The first component of a sinusoidal track.

8.2.12
Continuation
A sinusoidal track component that is not at the start or the end of a track.

8.2.13
Death
The last component of a sinusoidal track.

8.2.14
SMR
Signal-to-masking ratio.

8.2.15
Partial
Sinusoid of a limited duration.

8.2.16
IID
Inter-channel Intensity Differences.

8.2.17
IPD
Inter-channel Phase Differences.

8.2.18
OPD
Overall Phase Differences.

8.2.19
ICC
Inter-channel Coherence.

8.3 Symbols and abbreviations

8.3.1 Arithmetic operators

$\lfloor x \rfloor$ Round x towards minus infinity

$\lceil x \rceil$ Round x towards plus infinity.

mod Modulus operator: $\mathrm{mod}(x, y) = x - \left\lfloor \frac{x}{y} \right\rfloor y$. Defined only for positive values of *x* and *y*.

$\Gamma(\alpha)$ Gamma distribution function, defined as $\Gamma(\alpha) = \int_0^\infty e^{-t} \cdot t^{\alpha-1} dt$.

8.3.2 Relation operators

x?y:z If x is true then y else z.

8.3.3 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

uimsbf Unsigned integer, most significant bit first.

simsbf Signed integer, most significant bit first.

bslbf Bitstream left bit first.

8.3.4 Ranges

[0, 10] A number in the range of 0 up to and including 10.

[0, 10> A number in the range of 0 up to but excluding 10.

8.3.5 Number notation

%X Binary number representation (e.g. %01111100).

$X Hexadecimal number representation (e.g. $7C).

X Numbers with no prefix use decimal representation (e.g. 124).

8.3.6 Definitions

S Number of samples in a sub-frame (see Table 8.11).

L Number of samples in a segment; L = 2*S.

numQMFSlots Number of QMF subband samples per ps_data() element. For SSC, this parameter is fixed to 24.

8.4 Payloads for the audio object type SSC

8.4.1 Decoder configuration (SSCSpecificConfig)

Table 8.1 – Syntax of SSCSpecificConfig()

Syntax	Num. bits	Mnemonic
SSCSpecificConfig (channelConfiguration)		
{		
decoder_level	**2**	**uimsbf**
update_rate	**4**	**uimsbf**
synthesis_method	**2**	**uimsbf**
if (channelConfiguration != 1)		
{		
mode_ext	**2**	**uimsbf**
if ((channelConfiguration == 2) && (mode_ext == 1))		
{		
reserved	**2**	**uimsbf**
}		
}		
}		

8.4.2 SSC Bitstream Payload

Table 8.2 – Syntax of ssc_audio_frame()

Syntax	Num. bits	Mnemonic
ssc_audio_frame ()		
{		
ssc_audio_frame_header()		
ssc_audio_frame_data()		
}		

Table 8.3 – Syntax of ssc_audio_frame_header()

Syntax	Num. bits	Mnemonic
ssc_audio_frame_header ()		
{		
refresh_sinusoids	**1**	**uimsbf**
refresh_sinusoids_next_frame	**1**	**uimsbf**
refresh_noise	**1**	**uimsbf**
for (ch = 0; ch < nrof_channels; ch++)		
{		
s_nrof_continuations[0][ch]	Note 1	**uimsbf**
}		
n_nrof_den	**5**	**uimsbf**
n_nrof_lsf	Note 1	**uimsbf**
freq_granularity	**2**	**uimsbf**
amp_granularity	**2**	**uimsbf**
phase_jitter_present	**1**	**uimsbf**
if (phase_jitter_present == 1)		
{		
phase_jitter_percentage	**2**	**uimsbf**
phase_jitter_band	**2**	**uimsbf**
}		
}		
Note 1: See description of s_nrof_continuations and n_nrof_lsf in section 8.5.2.		

Table 8.4 – Syntax of ssc_audio_frame_data()

Syntax	Num. bits	Mnemonic
ssc_audio_frame_data()		
{		
for (sf = 0; sf < nrof_subframes; sf++)		
{		
for (ch = 0; ch < nrof_channels; ch++)		
{		
ssc_mono_subframe(sf,ch)		
if ((channelConfiguration == 2) && (mode_ext == 1)		
&& (mod(sf+1,4)==0))		
{		
ps_data()		
}		
}		
}		
}		

Table 8.5 – Syntax of ssc_mono_subframe()

Syntax	Num. bits	Mnemonic
ssc_mono_subframe (sf,ch)		
{		
subframe_transients(sf, ch)		
subframe_sinusoids(sf, ch)		
subframe_noise(sf, ch)		
}		