INTERNATIONAL STANDARD

ISO/IEC 14496-3

Fourth edition
2009-09-01

AMENDMENT 2
2010-10-15

Information technology — Coding of audio-visual objects —

Part 3: **Audio**

AMENDMENT 2: ALS simple profile and transport of SAOC

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

AMENDEMENT 2: Profil simple ALS et transport de SAOC

Reference number
ISO/IEC 14496-3:2009/Amd.2:2010(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 14496-3:2009 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 — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 2: ALS simple profile and transport of SAOC

Changes in existing text and tables are highlighted by gray background.

In 1.2, Normative references, add:

ISO/IEC 23003-2, *Information technology — MPEG audio technologies — Part 2: Spatial Audio Object Coding (SAOC)*

In 1.3, Terms and definitions, alphabetically incorporate the following into the list and renumber the subsequent index-number-entries:

LD MPEG Surround: Low Delay MPEG Surround

SAOC: Spatial Audio Object Coding

In 1.5.1.1, Audio object type definition, amend Table 1.1 by incorporating the updates below:

Object Type ID	Audio Object Type	Remark
	…	
42	(reserved)	
43	SAOC	
44	LD MPEG Surround	
45-95	(reserved)	

After 1.5.1.2.37, add the following two new subclauses:

1.5.1.2.38 SAOC object type

The SAOC object type conveys Spatial Audio Object Coding side information (see ISO/IEC 23003-2) in the MPEG-4 Audio framework.

1.5.1.2.39 LD MPEG Surround object type

The LD MPEG Surround object type conveys Low Delay MPEG Surround Coding side information (see ISO/IEC 23003-2) in the MPEG-4 Audio framework.

In 1.5.2.1 (Profiles), add:

14. The **ALS Simple Profile** contains the audio object type 36 (ALS).

In 1.5.2.1 (Profiles), Table 1.3 (Audio Profiles definition), add:

Object Type ID	Audio Object Type	…	ALS Simple Profile
…	…	…	…
36	ALS	…	X
…	…	…	…
42	(reserved)		
43	SAOC		
44	LD MPEG Surround		

In 1.5.2.3 (Levels within the profiles), add:

- **Levels for the ALS Simple Profile**

Table AMD 2-1 – Level for the ALS Simple Profile

Level	Max. number of channels	Max. sampling rate [kHz]	Max. word length [bit]	Max. number of samples per frame	Max. prediction order	Max. BS* stages	Max. MCC** stages
1	2	48	16	4096	15	3	1

* BS: Block switching, ** MCC: Multi-channel coding

The BGMC tool and the RLS-LMS tool are not permitted. Floating-point audio data is not supported.

In 1.5.2.4 (audioProfileLevelIndication), insert the following new entries into Table 1.14 (audioProfileLevelIndication values) and adapt the "reserved for ISO use" range accordingly:

Value	Profile	Level
…		
0x3C	ALS Simple Profile	L1
0x3D	SAOC Baseline Profile	L1
0x3E	SAOC Baseline Profile	L2
0x3F	SAOC Baseline Profile	L3
0x40	SAOC Baseline Profile	L4
0x41	SAOC LD Profile	L1
0x42	SAOC LD Profile	L2
0x43	SAOC LD Profile	L3
0x44 - 0x7F	reserved for ISO use	-
…		

In 1.6.2.1, extend Table1.15 "AudioSpecificConfig()"as follows:

Table 1.15 – Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic

AudioSpecificConfig ()
{
 	
…
 	
sbrPresentFlag = -1;
psPresentFlag = -1;
mpsPresentFlag = -1;
saocPresentFlag = -1;
ldmpsPresentFlag = -1;
if (audioObjectType == 5 ||
audioObjectType == 29) {		

```
...

case 40:
case 41:
    SymbolicMusicSpecificConfig()
    break;
case 43:
    saocPresentFlag = 1;
    saocPayloadEmbedding;                                   1       uimsbf
    SaocSpecificConfig();
    break;
case 44:
    ldmpsPresentFlag = 1;
    ldsacPayloadEmbedding;                                  1       uimsbf
    LDSpatialSpecificConfig();
    break;
default:
    /* reserved */
}

...

                extensionChannelConfiguration;              4       uimsbf
            }
        }
    }
    if (extensionIdentifier == -1 && bits_to_decode() >= 11 ) {
        extensionIdentifier;                                11      bslbf
    }
    if ( extensionIdentifier == 0x76a ) {
        extensionIdentifier = -1;
        if ( audioObjectType != 30 && bits_to_decode() >= 1 ) {
            mpsPresentFlag;                                 1       uimsbf
            if ( mpsPresentFlag == 1 ) {
                sacPayloadEmbedding = 1;
                sscLen;                                     8       uimsbf
                if ( sscLen == 0xff ) {
                    sscLenExt;                              16      uimsbf
                    sscLen += sscLenExt;
                }
                SpatialSpecificConfig();
            }
        }
    }
    if (extensionIdentifier == -1 && bits_to_decode() >= 11 ) {
        extensionIdentifier;                                11      bslbf
    }
    if ( extensionIdentifier == 0x7cb ) {
        extensionIdentifier = -1;
        if ( audioObjectType != 43 && bits_to_decode() >= 1 ) {
            saocPresentFlag;                                1       uimsbf
            if ( saocPresentFlag == 1 ) {
                saocPayloadEmbedding = 1;
                saocscLen;                                  8       uimsbf
                if ( saocscLen == 0xff ) {
                    saocscLenExt;                           16      uimsbf
```

```
                        saocscLen += saocscLenExt;
                    }
                    SaocSpecificConfig();
                }
            }
        }
        if (extensionIdentifier == -1 && bits_to_decode() >= 11 ) {
            extensionIdentifier;                                    11      bslbf
        }
        if ( extensionIdentifier == 0x7cc ) {
            extensionIdentifier = -1;
            if ( audioObjectType != 44 && bits_to_decode() >= 1 ) {
                ldmpsPresentFlag;                                   1       uimsbf
                if ( ldmpsPresentFlag == 1 ) {
                    ldsacPayloadEmbedding = 1;
                    ldsscLen;                                       8       uimsbf
                    if ( ldsscLen == 0xff ) {
                        ldsscLenExt;                                16      uimsbf
                        ldsscLen += ldsscLenExt;
                    }
                    LDSpatialSpecificConfig();
                }
            }
        }
}
```

After 1.6.2.1.17, add 1.6.2.1.18 and 1.6.2.1.19 as follows:

1.6.2.1.18 SaocSpecificConfig

Defined in 6.1 of ISO/IEC 23003-2.

1.6.2.1.19 LDSpatialSpecificConfig

Defined in B.2.1 of ISO/IEC 23003-2.

In 1.6.2.2.1, extend Table 1.17 "Audio Object Types" as follows:

Table 1.17 — Audio Object Types

Object Type ID	Audio Object Type	definition of elementary stream payloads and detailed syntax	Mapping of audio payloads to access units and elementary streams
0	NULL		
…			
41	SMR Main	ISO/IEC 14496-23	
42	(reserved)		
43	SAOC	ISO/IEC 23003-2	
44	LD MPEG Surround	ISO/IEC 23003-2	

After 1.6.3.20, add the following new subclauses 1.6.3.21 until 1.6.3.28 as follows:

1.6.3.21 saocPayloadEmbedding

The audio Object Type ID 43 SAOC is used to convey spatial audio object coding side information for SAOC decoding as defined in ISO/IEC 23003-2. Depending on this flag, the SAOC data payload, i.e., SAOCFrame(), is available by different means:

Table AMD 2-2 – saocPayloadEmbedding

saocPayloadEmbedding	Meaning
0	One SAOCFrame() is mapped into one access unit. Subsequent access units form one elementary stream. That elementary stream will always depend on another elementary stream that contains the underlying (downmixed) audio data.
1	The top level payload is multiplexed into the underlying (downmixed) audio data. The actual multiplexing details depend on the presentation of the audio data (i.e., usually on the AOT). Note that this leads to an elementary stream with no real payload. That elementary stream will always depend on another elementary stream that contains both, the underlying (downmixed) audio data and the multiplexed spatial audio data.

1.6.3.22 saocPresentFlag

A one bit field indicating the presence or absence of SAOC data. The value –1 indicates that the saocPresentFlag was not conveyed in the AudioSpecificConfig().

1.6.3.23 saocscLen

A helper variable indicating the number of bytes of the subsequent SaocSpecificConfig() data function including possible fill bits.

1.6.3.24 saocscLenExt

A helper variable indicating the additional number of bytes of the subsequent SaocSpecificConfig() data function including possible fill bits.

1.6.3.25 ldsacPayloadEmbedding

The audio Object Type ID 44 LD MPEG Surround is used to convey low delay spatial audio coding side information for LD MPEG Surround decoding as defined in ISO/IEC 23003-2. Depending on this flag, the LD MPEG Surround data payload, i.e., LDSpatialFrame(), is available by different means:

Table AMD 2-3 – ldsacPayloadEmbedding

ldsacPayloadEmbedding	Meaning
0	One LDSpatialFrame() is mapped into one access unit. Subsequent access units form one elementary stream. That elementary stream will always depend on another elementary stream that contains the underlying (downmixed) audio data.
1	The top level payload is multiplexed into the underlying (downmixed) audio data. The actual multiplexing details depend on the presentation of the audio data (i.e., usually on the AOT). Note that this leads to an elementary stream with no real payload. That elementary stream will always depend on another elementary stream that contains both, the underlying (downmixed) audio data and the multiplexed spatial audio data.

1.6.3.26 ldmpsPresentFlag

A one bit field indicating the presence or absence of LD MPEG Surround data. The value –1 indicates that the ldmpsPresentFlag was not conveyed in the AudioSpecificConfig().

1.6.3.27 ldsscLen

A helper variable indicating the number of bytes of the subsequent LDSpatialSpecificConfig() data function including possible fill bits.

1.6.3.28 ldsscLenExt

A helper variable indicating the additional number of bytes of the subsequent LDSpatialSpecificConfig() data function including possible fill bits.

In 4.4.2.7, extend Table 4.57 "Syntax of extension_payload()" as follows:

Table 4.57 – Syntax of extension_payload()

Syntax	No. of bits	Mnemonic
extension_payload(cnt)		
{		
extension_type;	**4**	**uimsbf**
align = 4;		
switch(extension_type) {		
case EXT_DYNAMIC_RANGE:		
return dynamic_range_info();		
case EXT_SAC_DATA:		
return sac_extension_data(cnt);		
case EXT_SAOC_DATA:		
return saoc_extension_data(cnt);		
case EXT_LDSAC_DATA:		
return ldsac_extension_data(cnt);		
case EXT_SBR_DATA:		
return sbr_extension_data(id_aac, 0);		Note 1
case EXT_SBR_DATA_CRC:		

return sbr_extension_data(id_aac, 1);			Note 1
case EXT_DATA_LENGTH:			
hlp = 1;			
len;		**4**	**uimsbf**
if (len==15) {			
len += **add_len;**		**8**	**uimsbf**
hlp += 1;			
If (add_len==255) {			
len += **add_add_len**;		**16**	**uimsbf**
hlp += 2;			
}			
}			
return hlp+extension_payload(len);			Note 2
case EXT_FILL_DATA:			
fill_nibble; /* must be '0000' */		**4**	**uimsbf**
for (i=0; i<cnt-1; i++) {			
fill_byte[i]; /* must be '10100101' */		**8**	**uimsbf**
}			
return cnt;			
case EXT_DATA_ELEMENT:			
data_element_version;		**4**	**uimsbf**
switch(data_element_version) {			
case ANC_DATA:			
loopCounter = 0;			
dataElementLength = 0;			
do {			
dataElementLengthPart;		**8**	**uimsbf**
dataElementLength += dataElementLengthPart;			
loopCounter++;			
} while (dataElementLengthPart == 255);			
for (i=0; i<dataElementLength; i++) {			
data_element_byte[i];		**8**	**uimsbf**
}			
return (dataElementLength+loopCounter+1);			
default:			
align = 0;			
}			
case EXT_FIL:			
default:			
for (i=0; i<8*(cnt-1)+align; i++) {			
other_bits[i];		**1**	**uimsbf**
}			
return cnt;			
}			
}			

Note 1: id_aac is the id_syn_ele of the corresponding AAC element (ID_SCE or ID_CPE) or ID_SCE in case of CCE.

Note 2: The extension_payload() included here must not have extension_type == EXT_DATA_LENGTH.

In 4.4.2.7, after Table 4.61, add two new Tables, Table AMD 2-4 "saoc_extension_data()" and Table AMD 2-5 "ldsac_extension_data()" as given below:

Table AMD 2-4 – Syntax of saoc_extension_data()

Syntax	No. of bits	Mnemonic
saoc_extension_data(cnt)		
{		
ancType;	**2**	**uimsbf**
ancStart;	**1**	**uimsbf**
ancStop;	**1**	**uimsbf**
for (i=0; i<cnt-1; i++) {		
ancDataSegmentByte[i];	**8**	**bslbf**
}		
return (cnt);		
}		

Table AMD 2-5 – Syntax of ldsac_extension_data()

Syntax	No. of bits	Mnemonic
ldsac_extension_data(cnt)		
{		
ancType;	**2**	**uimsbf**
ancStart;	**1**	**uimsbf**
ancStop;	**1**	**uimsbf**
for (i=0; i<cnt-1; i++) {		
ancDataSegmentByte[i];	**8**	**bslbf**
}		
return (cnt);		
}		

In 4.5.2.9.3, extend Table 4.121 "Values of the extension_type field" as follows:

Table 4.121 – Values of the extension_type field

Symbol	Value of extension_type	Purpose
EXT_FILL	'0000'	bitstream payload filler
EXT_FILL_DATA	'0001'	bitstream payload data as filler
EXT_DATA_ELEMENT	'0010'	data element
EXT_DATA_LENGTH	'0011'	container with explicit length for extension_payload()
EXT_LDSAC_DATA	'1001'	LD MPEG Surround
EXT_SAOC_DATA	'1010'	SAOC
EXT_DYNAMIC_RANGE	'1011'	dynamic range control
EXT_SAC_DATA	'1100'	MPEG Surround
EXT_SBR_DATA	'1101'	SBR enhancement
EXT_SBR_DATA_CRC	'1110'	SBR enhancement with CRC
-	all other values	Reserved: These values can be used for a further extension of the syntax in a compatible way.
Note: Extension payloads of the type EXT_FILL or EXT_FILL_DATA have to be added to the bitstream payload if the total bits for all audio data together with all additional data are lower than the minimum allowed number of bits in this frame necessary to reach the target bitrate. Those extension payloads are avoided under normal conditions and free bits are used to fill up the bit reservoir. Those extension payloads are written only if the bit reservoir is full.		

After 4.5.2.11, add two new subclauses as given below:

4.5.2.12 Spatial Audio Object Coding (SAOC)

The syntax element saoc_extension_data() is used to embed spatial audio object coding side information for SAOC decoding as defined in ISO/IEC 23003-2. The semantics of the syntax elements ancType, ancStart, ancStop, and ancDataSegmentByte is defined in 8.2.4 of ISO/IEC 23003-2.

4.5.2.13 Low Delay MPEG Surround (LD MPEG Surround)

The syntax element ldsac_extension_data() is used to embed low delay spatial audio coding side information for LD MPEG Surround decoding as defined in ISO/IEC 23003-2. The semantics of the syntax elements ancType, ancStart, ancStop, and ancDataSegmentByte is defined in 8.2.4 and B.4 of ISO/IEC 23003-2.

In 4.5.4, add the following entries to Table 4.149 "AAC error sensitivity category assignment for extended payload and low delay sbr payload":

Table 4.149 – AAC error sensitivity category assignment for extended payload and low delay sbr payload

extension_payload	low delay sbr payload	data_element	function
5	-	**len**	extension_payload()
5	-	**add_len**	extension_payload()
5	-	**add_add_len**	extension_payload()
11	-	**ancType**	sac_extension_data()
11	-	**ancStart**	sac_extension_data()
11	-	**ancStop**	sac_extension_data()
11	-	**ancDataSegmentByte**	sac_extension_data()
12	-	**ancType**	saoc_extension_data()
12	-	**ancStart**	saoc_extension_data()
12	-	**ancStop**	saoc_extension_data()
12	-	**ancDataSegmentByte**	saoc_extension_data()
13	-	**ancType**	ldsac_extension_data()
13	-	**ancStart**	ldsac_extension_data()
13	-	**ancStop**	ldsac_extension_data()
13	-	**ancDataSegmentByte**	ldsac_extension_data()