INTERNATIONAL STANDARD

ISO/IEC 14496-3

Third edition
2005-12-01

AMENDMENT 5
2007-07-15

Information technology — Coding of audio-visual objects —

Part 3: **Audio**

AMENDMENT 5: BSAC extensions and transport of MPEG Surround

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

AMENDEMENT 5: Extensions BSAC et transport d'ambiance MPEG

Reference number
ISO/IEC 14496-3:2005/Amd.5:2007(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 5 to ISO/IEC 14496-3:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This Amendment specifies the normative syntax of the integration of ER BSAC and the SBR tool and the corresponding decoding process. An informative encoder description is given in the Annex. Furthermore, this Amendment specifies the transport and embedding of MPEG Surround (ISO/IEC 23003-1) side information in MPEG-4 AAC bitstreams and MPEG-4 Audio environments.

Information technology — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 5: BSAC extensions and transport of MPEG Surround

In the following, changes to existing text and Tables are highlighted by gray background.

In the fourth paragraph of subclause 0.3.2.2 General audio coding tools, replace:

It operates at up to 48 kHz sampling rate and uses a frame length of 512 or 480 samples, compared to the 1024 or 960 samples used in standard MPEG-2/4 AAC to enable coding of general audio signals with an algorithmic delay not exceeding 20 ms.

with:

To enable coding of general audio signals with an algorithmic delay not exceeding 20 ms at 48 kHz, it uses a frame length of 512 or 480 samples (compared to the 1024 or 960 samples used in standard MPEG-2/4 AAC).

In Part 3: Audio, Subpart 1, in subclause 1.2 Normative references, add:

ISO/IEC 14496-23, *Information technology — Coding of audio-visual objects — Part 23: Symbolic Music Representation*

ISO/IEC 23003-1, *Information technology — MPEG audio technologies — Part 1: MPEG Surround*

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

PS: Parametric Stereo

SMR: Symbolic Music Representation

and increase the index-number of subsequent entries.

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

Tools/ Modules Audio Object Type	Remark	Object Type ID
…		
MPEG Surround		30
…		
(escape)	X	31
…		
(reserved)		39
SMR Simple		40
SMR Main		41
...		

In Part 3: Audio, Subpart 1, after subclause 1.5.1.2.32 add the following subclauses:

1.5.1.2.33 PS object type

The PS object type contains the PS tool and can be combined with the SBR tool.

1.5.1.2.34 MPEG Surround object type

The MPEG Surround object type conveys MPEG Surround side information (see ISO/IEC 23003-1) in the MPEG-4 Audio framework.

1.5.1.2.35 SMR Simple object type

The SMR simple object type it is used to transport musical scores to be rendered both audibly and visually. The encoded data contains information on the main score, the parts (i.e. instrument specific scores), the possible multilingual lyrics associated with the parts, the visual formatting rules to be used for visual rendering, fonts to be used for custom musical symbols and synchronization information. The fonts and the synchronization information are represented as binary data, all the other as XML data. The SMR simple object type can convey the XML data as plain XML text or as gzip-ped XML (see ISO/IEC 14496-23).

1.5.1.2.36 SMR Main object type

The SMR main object type can transport the musical scores exactly in the same way as the SMR Simple object type but in this case the XML data can be encoded also using the MPEG-B tools defined in ISO/IEC 23001-1 (see ISO/IEC 14496-23).

In Part 3: Audio, Subpart 1, in subclause 1.5.1.2.6 SBR object type, replace Table 1.2 with the table below:

Table 1.2 — Audio object types that can be combined with SBR

Audio Object Type	Object Type ID
AAC main	1
AAC LC	2
AAC SSR	3
AAC LTP	4
AAC Scalable	6
ER AAC LC	17
ER AAC LTP	19
ER AAC scalable	20
ER BSAC	22

In subclause 1.5.2.1, replace Table 1.3 Audio Profiles definition with the table below:

Object Type ID	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	High Efficiency AAC v2 Profile
0	Null											
1	AAC main	X						X				
2	AAC LC	X	X			X		X		X	X	X
3	AAC SSR	X						X				
4	AAC LTP	X	X			X		X				
5	SBR										X	X
6	AAC Scalable	X	X			X		X				
7	TwinVQ	X	X					X				
8	CELP	X	X	X		X	X	X				
9	HVXC	X	X	X			X	X				
10	(reserved)											
11	(reserved)											
12	TTSI	X	X	X	X		X	X				
13	Main synthetic	X			X							
14	Wavetable synthesis	X*			X*							
15	General MIDI	X*			X*							
16	Algorithmic Synthesis and Audio FX	X*			X*							
17	ER AAC LC					X		X	X			
18	(reserved)											
19	ER AAC LTP					X		X				
20	ER AAC Scalable					X		X	X			
21	ER TwinVQ							X	X			
22	ER BSAC							X	X			
23	ER AAC LD						X	X	X			
24	ER CELP					X	X	X				
25	ER HVXC						X	X				
26	ER HILN							X				
27	ER Parametric							X				
28	SSC											
29	PS											X
30	MPEG Surround											
31	(escape)											
32	Layer-1											
33	Layer-2											
34	Layer-3											
35	DST											

36	ALS											
37	SLS											
38	SLS non-core											
39	(reserved)											
40	SMR Simple											
41	SMR Main											

*) Subset of Main Synthetic

In subclause 1.6.2.1, extend Table 1.13 — AudioSpecificConfig() as follows:

Table 1.13 — Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic
AudioSpecificConfig ()		
{		
audioObjectType = GetAudioObjectType();		
samplingFrequencyIndex;	**4**	**uimsbf**
if (samplingFrequencyIndex==0xf)		
samplingFrequency;	**24**	**uimsbf**
channelConfiguration;	**4**	**uimsbf**
sbrPresentFlag = -1;		
psPresentFlag = -1;		
if (audioObjectType == 5 ||		
audioObjectType == 29) {		
extensionAudioObjectType = audioObjectType;		
sbrPresentFlag = 1;		
if (audioObjectType == 29) {		
psPresentFlag = 1;		
}		
extensionSamplingFrequencyIndex;	**4**	**uimsbf**
if (extensionSamplingFrequencyIndex==0xf)		
extensionSamplingFrequency;	**24**	**uimsbf**
audioObjectType = GetAudioObjectType();		
if (audioObjectType == 22)		
extensionChannelConfiguration;	**4**	**uimsbf**
}		
else {		
extensionAudioObjectType = 0;		
}		
switch (audioObjectType) {		
...		
case 28:		
SSCSpecificConfig();		
break;		
case 30:		
sacPayloadEmbedding;	**1**	**uimsbf**
SpatialSpecificConfig();		
break;		
case 32:		
case 33:		
case 34:		
...		

```
    case 40:
    case 41:
        SymbolicMusicSpecificConfig()
        break;
    default:
        /* reserved */
    }

    ...

    if (  extensionAudioObjectType != 5 &&
        bits_to_decode() >= 16 ) {
        syncExtensionType;                                  11      bslbf
        if (syncExtensionType == 0x2b7) {
            extensionAudioObjectType = GetAudioObjectType();
            if ( extensionAudioObjectType == 5 ) {
                sbrPresentFlag;                             1       uimsbf
                if (sbrPresentFlag == 1) {
                    extensionSamplingFrequencyIndex;        4       uimsbf
                    if ( extensionSamplingFrequencyIndex == 0xf ) {
                        extensionSamplingFrequency;         24      uimsbf
                    }
                    if ( bits_to_decode() >= 12 ) {
                        syncExtensionType;                  11      bslbf
                        if (syncExtensionType == 0x548) {
                            psPresentFlag;                  1       uimsbf
                        }
                    }
                }
            }
            if ( extensionAudioObjectType == 22 ) {
                sbrPresentFlag;                             1       uimsbf
                if (sbrPresentFlag == 1) {
                    extensionSamplingFrequencyIndex;        4       uimsbf
                    if ( extensionSamplingFrequencyIndex == 0xf )
                        extensionSamplingFrequency;         24      uimsbf
                }
                extensionChannelConfiguration;              4       uimsbf
            }
        }
    }
}
```

After subclause 1.6.2.1.13, add the following new subclauses:

1.6.2.1.14 SpatialSpecificConfig

Defined in ISO/IEC 23003-1:2007, 5.1.

1.6.2.1.15 SymbolicMusicSpecificConfig

Defined in ISO/IEC 14496-23:—[1)], 7.2.1.

1) To be published.

In subclause 1.6.2.2.1, extend Table 1.15 — Audio Object Types as follows:

Table 1.15 — 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		
…			
29	PS	ISO/IEC 14496-3 subpart 8	
30	MPEG Surround	ISO/IEC 23003-1	
31	(escape)		
…			
39	(reserved)		
40	SMR Simple	ISO/IEC 14496-23	
41	SMR Main	ISO/IEC 14496-23	

After subclause 1.6.3.15 fillBits, add the following new subclauses:

1.6.3.16 extensionChannelConfiguration

A four bit field indicating the channel configuration of the BSAC extensions. This configuration data is available in case of explicit signalling for the BSAC extensions. The number of audio output channels is determined according to Table 1.17 — Channel Configuration.

1.6.3.17 sacPayloadEmbedding

The audio Object Type ID 30 MPEG Surround is used to convey spatial audio coding side information for MPEG Surround decoding as defined in ISO/IEC 23003-1. Depending on this flag, the MPEG surround data payload, i.e. SpatialFrame(), is available by different means:

Table 1.19A — sacPayloadEmbedding

sacPayloadEmbedding	Meaning
0	One SpatialFrame() 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.

In Part 3: Audio, Subpart 1, after 1.6.6 Signaling of Parametric Stereo (PS), add the following new sublcause:

1.6.7 Signalling of BSAC Extension payloads

The implicit signalling method for BSAC extension payloads is similar to that of SBR tool. The BSAC decoder that can decode the BSAC extension payloads checks if there is an extension type for the SBR tool such as 'EXT_BSAC_SBR_DATA' in the bsac_raw_data_block(). The sampling frequency should be updated, when the extension type is detected and the SBR tool is operated in dual-rate mode.

The BSAC extension decoder checks if there is the extension type for the BSAC channel extension such as 'EXT_BSAC_CHANNEL' in the bsac_raw_data_block(). In case where the extension type for multichannel, the number of channel from the AudioSpecificConfig() for BSAC Audio Object Type is updated depending on the 'channel_configu-ration_index' of each extended_bsac_base_element().

When explicit signalling is used, implicit signalling shall not occur. Two different types of explicit signalling are available:

1. Explicit Signalling Method 1: hierarchical signalling
   If the first audioObjectType (AOT) signalled is the SBR AOT, a second audio object type is signalled which indicates the ER BSAC AOT. The extensionChannelConfiguration indicates the total number of channels in the bsac_raw_data_block().

2. Explicit Signalling Method 2 : backward compatible signalling
   The extensionAudioObjectType is signalled at the end of the AudioSpecificConfig(). If the extensionAudioObjectType is the ER BSAC AOT, the extensionChannelConfiguration indicates the total number of channels in the bsac_raw_data_block(). This method shall only be used in systems that convey the length of the AudioSpecificConfig(). Hence, it shall not be used for LATM with audioMuxVersion==0.

Table 1.22B explains the decoder behaviour with SBR and BSAC channel extension signalling.

Table 1.22B – SBR and BSAC channel extension signalling and Corresponding Decoder Behaviour

Bitstream characteristics				Decoder behaviour	
Extension AudioObjectType	**sbrPresentFlag**	**extensionChannelConfiguration**	**raw_data_block**	**BSAC decoder**	**BSAC Extensions decoder**
!= ER_BSAC (Implicit Signalling)	-1 (Note 1)	Not available	BSAC	Play BSAC	Play BSAC
			BSAC+SBR	Play BSAC	Play at least BSAC, should play BSAC+SBR
			BSAC+MC	Play BSAC	Play at least BSAC, should play BSAC+MC
			BSAC+SBR+MC	Play BSAC	Play at least BSAC, should play BSAC+SBR +MC
== ER_BSAC (Explicit Signalling)	0 (Note 2)	== channelConfiguration (Note 4)	BSAC	Play BSAC	Play BSAC
		!= channelConfiguration	BSAC + MC	Play BSAC	Play BSAC+MC
	1 (Note 3)	== channelConfiguration (Note 4)	BSAC+SBR	Play BSAC	Play BSAC+SBR
		!= channelConfiguration	BSAC+SBR+MC	Play BSAC	Play BSAC+SBR +MC

Note 1: Implicit signalling, check payload in order to determine output sampling frequency, or assume the presence of SBR data in the payload, giving an output sampling frequency of twice the sampling frequency indicated by samplingFrequency in the AudioSpecificConfig() (unless the down sampled SBR Tool is operated, or twice the sampling frequency indicated by samplingFrequency exceeds the maximum allowed output sampling frequency of the current level, in which case the output sampling frequency is the same as indicated by samplingFrequency).

Note 2: Explicitly signals that there is no SBR data, hence no implicit signalling is present, and the output sampling frequency is given by samplingFrequency in the AudioSpecificConfig().

Note 3: Output sampling frequency is the extensionSamplingFrequency in AudioSpecificConfig().

Note 4: Explicitly signals that there is no BSAC channel extension data, and the number of output channel is given by channelConfiguration in the AudioSpecificConfig().

Replace the definition of bsac_raw_data_block() in Part 3: Audio, Subpart 4, subclause 4.4.2.6 Payloads for the audio object type ER BSAC, Table 4.33:

Table 4.33 — Syntax of bsac_raw_data_block()

Syntax	No. of bits	Mnemonic
bsac_raw_data_block()		
{		
bsac_base_element();		
layer=slayer_size;		
while(data_available() && layer<(top_layer+slayer_size)) {		
bsac_layer_element(layer);		
layer++;		
}		
byte_alignment();		
if (data_available()) {		
zero_code	**32**	**bslbf**
sync_word	**4**	**bslbf**
while(bits_to_decode() > 4) {		
extension_type	**4**	**bslbf**
switch(extension_type) {		
case EXT_BSAC_CHANNEL :		
extended_bsac_raw_data_block();		Note 1
break;		
case EXT_BSAC_SBR_DATA :		
extended_bsac_sbr_data(nch, 0);		
break;		
case EXT_BSAC_SBR_DATA_CRC :		
extended_bsac_sbr_data(nch, 1);		
break;		
case EXT_BSAC_CHANNEL_SBR :		
extended_bsac_raw_data_block();		Note 1
extended_bsac_sbr_data(nch, 0);		
break;		
case EXT_BSAC_CHANNEL_SBR_CRC :		
extended_bsac_raw_data_block();		Note 1
extended_bsac_sbr_data(nch, 1);		
break;		
case EXT_BSAC_SAC_DATA :		
extended_bsac_sac_data();		
break;		
default :		
extended_bsac_data();		
break;		
}		
byte_alignment();		
}		
}		
}		
Note 1: The byte_alignment() functions in extended_bsac_raw_data_block() are relative to the start of extended_bsac_raw_data_block().		

Renumber Table 4.35 – Syntax of extended_bsac_raw_data_block() and Table 4.36 – Syntax of extended_bsac_base_element() which were introduced in ISO/IEC 14496-3:2005/Amd.2 as Table 4.43A – Syntax of extended_bsac_raw_data_block() and Table 4.43B – Syntax of extended_bsac_base_element(), respectively.

Add the following tables after Table 4.43B – Syntax of extended_bsac_base_element() in Part 3: Audio, Subpart 4, at the end of subclause 4.4.2.6:

Table 4.43C — Syntax of extended_bsac_sbr_data ()

Syntax	No. of bits	Mnemonic
extended_bsac_sbr_data(nch, crc_flag)		
{		
num_sbr_bits = 0;		
cnt = **count;**	**4**	**uimsbf**
num_sbr_bits += 4;		
if (cnt == 15) {		
cnt += **esc_count** - 1;	**8**	**uimsbf**
num_sbr_bits += 8;		
}		
if (crc_flag) {		
bs_sbr_crc_bits;	**10**	**uimsbf**
num_sbr_bits += 10;		
}		
num_sbr_bits += 1;		
if (**bs_header_flag**)	**1**	**uimsbf**
num_sbr_bits += sbr_header();		
num_sbr_bits += bsac_sbr_data(nch, bs_amp_res);		
num_align_bits = (8*cnt - num_sbr_bits);		
bs_fill_bits;	num_align_ bits	**uimsbf**
}		

Table 4.43D — Syntax of bsac_sbr_data()

Syntax	No. of bits	Mnemonic
bsac_sbr_data(nch, bs_amp_res)		
{		
switch (nch) {		
case 1 :		
sbr_single_channel_element(bs_amp_res)		
break;		
case 2 :		
sbr_channel_pair_element(bs_amp_res)		
break;		
}		
}		

Table 4.43E — Syntax of extended_bsac_data()

Syntax	No. of bits	Mnemonic
extended_bsac_data()		
{		
cnt = **count;**	**8**	**uimsbf**
if (cnt == 255) {		
cnt += **esc_count** - 1;	**8**	**uimsbf**
}		
for (i=0 ; i < cnt-1 ; i++) {		
byte_payload	**8**	**uimsbf**
}		
}		

Table 4.43F — Syntax of extended_bsac_sac_data()

Syntax	No. of bits	Mnemonic
extended_bsac_sac_data()		
{		
cnt = **count;**	**8**	**uimsbf**
if (cnt == 255) {		
cnt += **esc_count** - 1;	**8**	**uimsbf**
}		
ancType;	**2**	**uimsbf**
ancStart;	**1**	**uimsbf**
ancStop;	**1**	**uimsbf**
bs_crc_flag	**1**	**uimsbf**
bs_fill_bits	**3**	**uimsbf**
if (bs_crc_flag) {		
ancCrcWord;	**8**	**uimsbf**
cnt = cnt - 1;		
}		
for (i=0; i<cnt-2; i++) {		
ancDataSegmentByte[i];	**8**	**bslbf**
}		
}		

In subclause 4.4.2.7, extend Table 4.51 — Syntax of extension_payload() as follows:

Table 4.51 — 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_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
…		
}		

In subclause 4.4.2.7 after Table 4.54 add a new Table 4.54A — Syntax of sac_extension_data() as given below:

Table 4.54A — Syntax of sac_extension_data()

Syntax	No. of bits	Mnemonic
sac_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 subclause 4.5.1 Decoding of the GA specific configuration, replace:

If a certain sampling frequency dependent table stated in the right column of Table 4.68 is not defined, the nearest defined table shall be used.

with:

If in a certain sampling frequency dependent table a sampling frequency stated in the right column of Table 4.68 is not defined, the nearest defined table shall be used.

In Part 3: Audio, Subpart 4, in subclause 4.5.2.6.2.1.1 Data elements, after bsac_raw_data_block(), remove the text related to semantics of zero_code, syncword, and extension type and add the following after bsac_spectral_data():

zero_code 32-bit zero values in order to terminate the arithmetic decoding for the stereo part.

sync_word a four bit code that identifies the start of the extended part. The bit string '1111'.

bits_to_decode() A helper function; returns the number of bits not yet decoded in the current bsac_raw_data_block ().

extension_type a four bit code that identifies the extension type according to Table 4.84A.

Table 4.84A — BSAC extension_type

Symbol	Value of extension_type	Purpose
EXT_BSAC_CHANNEL	'1111'	BSAC channel extension
EXT_BSAC_SBR_DATA	'0000'	BSAC SBR enhancement
EXT_BSAC_SBR_DATA_CRC	'0001'	SBR enhancement with CRC
EXT_BSAC_SAC_DATA	'0010'	MPEG Surround enhancement
EXT_BSAC_CHANNEL_SBR	'1110'	BSAC channel extension with SBR
EXT_BSAC_CHANNEL_SBR_CRC	'1101'	BSAC channel extension with SBR_CRC
RESERVED	'0011' ~ '1100'	reserved

In Part 3: Audio, Subpart 4, under Bitstream elements in subclause 4.5.2.6.2.1 Definitions, after reserved_bit, add the following:

extended_bsac_sbr_data () Syntactic element that contains the SBR extension data for ER BSAC

count Initial length of extended_bsac_sbr_data() or extended_bsac_data()

esc_count Incremental length of extended_bsac_sbr_data() or extended_bsac_data()

bs_sbr_crc_bits Cyclic redundancy checksum for the SBR extension data. The CRC code is defined by the generator polynomial G10(x) = x10 + x9 + x5 + x4 + x + 1 and the initial value for the CRC calculation is zero

bs_header_flag Indicates if an SBR header is present

sbr_header () Syntactic element that contains the SBR header. See Table 4.56 — Syntax of sbr_header()

bsac_sbr_data() Syntactic element that contains the SBR data for ER BSAC

bs_fill_bits Byte alignment bits

sbr_single_channel_element() Syntactic element that contains data for an SBR single channel element. See Table 4.58-Syntax of sbr_single_channel_element()

sbr_channel_pair_element() Syntactic element that contains data for an SBR channel pair element. See Table 4.59-Syntax of sbr_channel_pair_element()

extended_bsac_data() Syntactic element that contains the payload to be discarded by decoder. This is for the further extensions.

byte_payload Byte to be discarded by the decoder

extended_bsac_sac_data() The syntax element extended_bsac_sac_data() contains spatial audio coding side information for MPEG Surround decoding as defined in ISO/IEC 23003-1. The semantics of the syntax elements ancType, ancStart, ancStop, ancCrcWord, and ancDataSegmentByte are defined in ISO/IEC 23003-1:2007, 7.2.4.

bs_crc_flag Indicates if CRC word is present

In Part 3: Audio, Subpart 4, replace subclause 4.5.2.6.2.2.14 with:

4.5.2.6.2.2.14 Decoding the extended part

The structure of the extended part of BSAC is a simple replica of mono or stereo BSAC bitstream. The functions extended_bsac_raw_data_block and extended_bsac_base_element are defined in subclauses 4.5.2.11.2.1 and 4.5.2.11.2.2.

In Part 3: Audio, Subpart 4, remove subclauses 4.5.2.6.2.2.14.1 and 4.5.2.6.2.2.14.2.

In subclause 4.5.2.9.3 extend Table 4.105 — Values of the extension_type field as follows:

Table 4.105 — 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_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 subclause 4.5.2.9 add the following new subclauses:

4.5.2.10 MPEG Surround (Spatial Audio Coding)

The syntax element sac_extension_data() is used to embed spatial audio coding side information for MPEG Surround decoding as defined in ISO/IEC 23003-1. The semantics of the syntax elements ancType, ancStart, ancStop, and ancDataSegmentByte is defined in ISO/IEC 23003-1:2007, 7.2.4.

4.5.2.11 Payloads for the audio object type ER BSAC Extensions

4.5.2.11.1 Introduction

ER BSAC is extended to enhance the performance with extension payloads such as multichannel, SBR and MPEG Surround payloads. This extension is designed to maintain the functionality of FGS when a bitstream is transmitted over MPEG-4 system. ER-BSAC can provide several operating modes for various application scenarios, as shown in Table 4.106A.

Table 4.106A — Operating modes of BSAC Extensions

Operating Mode	Extension payload
BSAC mono/stereo	Not available
BSAC multichannel	BSAC multichannel extension payload
BSAC mono/stereo with SBR	BSAC SBR payload
BSAC multichannel with SBR	BSAC multichannel extension payload and BSAC SBR payload
BSAC with MPEG Surround	MPEG Surround payload

The integration of ER-BSAC and SBR is to provide fine grain scalable reproduction with bandwidth extension for stereo and multi-channel. In the preferred modes of operating the integration of ER-BSAC and SBR, the high frequency sound components can be either an ER-BSAC enhancement layer signal or synthesized SBR signal.

4.5.2.11.2 Decoding process

4.5.2.11.2.1 zero_code and sync_word

Syntax elements zero_code and sync_word are parsed while data is available after decoding a BSAC mono or stereo part. Syntax element zero_code is used for the arithmetic termination of the BSAC mono or stereo part, and sync_word is used to indicate the beginning of an extended part.

4.5.2.11.2.2 extended_bsac_raw_data_block

An extended_bsac_raw_data_block contains multichannel information and has the layered structure as bsac_raw_data_block.

4.5.2.11.2.3 extended_bsac_base_element

An extended bsac_base_element consists of **element_length**, **channel_configuration_index**, **reserved_bit**, bsac_header, general_header and bsac_layer_element. For the stereo part, the value of *nch* is obtained from **channelConfiguration** in Table 1.13 (Syntax of AudioSpecificConfig) and it is limited to either 1 or 2 (left and right front speakers). For the extended part, the parameter, *nch*, is concerned with the rest of speakers, and the exact value is determined by **channel_configuration_index** specified in Table 4.84B. Each index indicates the number of channels given the channel to speaker mapping.

4.5.2.11.2.4 extended_bsac_sbr_data

SBR elements are inserted into the bsac_raw_data_block() after layer element data. By combining a BSAC core coder and the SBR bandwidth extension tool, the functionality of fine grain scalability is supported with full bandwidth for all layers. The start frequency for the SBR frequency range is the base_band frequency of the BSAC base layer. The SBR part covers high frequency regions beyond the base layer frequency range. If enhancement layers of the BSAC core are transmitted, the overlapped region of the SBR data is replaced with core data.

4.5.2.11.2.5 extended_bsac_sac_data

MPEG Surround payload is embedded into the bsac_raw_data_block() after layer element data. For the MPEG Surround payload, the extension type 'EXT_BSAC_SAC_DATA' is defined. The syntax extended_bsac_sac_data() is shown in Table 4.43F. The semantics of the syntax elements *ancType, ancStart, ancStop, ancCrcWord*, and *ancDataSegmentByte* are defined in ISO/IEC 23003-1:2007, 7.2.4. In case that CRC error checking is necessary for the MPEG Surround payload, the *bs_crc_flag* is set to 1 and the *ancCrcWord* is transmitted. As specified in ISO/IEC 23003-1:2007, 7.2.3, the interface between ER BSAC and the MPEG Surround decoder can be either the time-domain or the QMF domain. In case of ER BSAC combined with SBR tool, the QMF representation is available as an input of the MPEG Surround decoder.

4.5.2.11.3 Technical overview of the integration of ER-BSAC and SBR tool

The basic structure of the integration of ER-BSAC and SBR tool is shown in Figure 4.21A. This subclause describes the decoding process of SBR tool integrated with ER BSAC in a scalable manner. The scalable SBR tool is made possible by adding a module referred to as the HF-Overlap. The decoding process of the scalable SBR tool is identical to that of SBR tool except the HF-Overlap module. The scalable SBR decoding block diagram is shown in the following Figure 4.21A. It displays how the HF-Overlap module is interconnected into the SBR decoder.

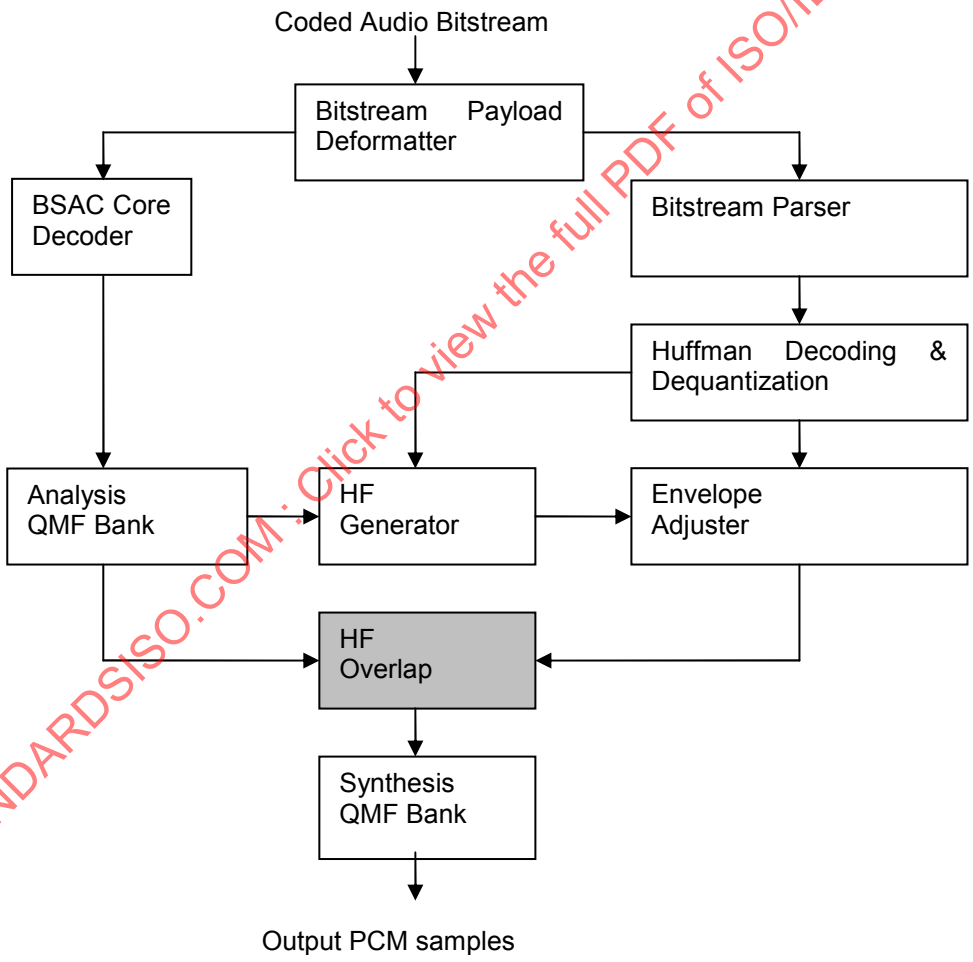

Figure 4.21A — Overview of the integration of ER-BSAC and SBR decoder