INTERNATIONAL STANDARD

ISO/IEC 14496-3

Third edition
2005-12-01

AMENDMENT 2
2006-03-15

Information technology — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 2: Audio Lossless Coding (ALS), new audio profiles and BSAC extensions

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

AMENDEMENT 2: Codage audio sans perte (ALS), nouveaux profils audio et extensions BSAC

Reference number
ISO/IEC 14496-3:2005/Amd.2:2006(E)

Documents
Document Sharing Hub
© ISO/IEC 2006

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2006

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Documents
Document Sharing Hub

© ISO/IEC 2006 – All rights reserved

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

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 14496-3:2005 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 Audio Lossless Coding (ALS) scheme. The amendment further defines a new profile, the High Efficiency AAC v2 Profile, that incorporates all the features of the High Efficiency AAC Profile and in addition the Parametric Stereo tool. The amendment also specifies the way in which the audio object type ER BSAC is extended to support multi-channel format, providing backward compatibility.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

Documents
Document Sharing Hub

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

Information technology — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 2: Audio Lossless Coding (ALS), new audio profiles and BSAC extensions

In the Introduction, at the end of subclause "Lossless Audio Coding Tools", add:

MPEG-4 ALS (Audio Lossless Coding) provides lossless coding of digital audio signals. Input signals can be integer PCM data with 8 to 32-bit word length or 32-bit IEEE floating-point data. Up to 65536 channels are supported.

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

ALS: Audio Lossless Coding

and increase the index-number of subsequent entries.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

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 Type definition based on Tools/Modules

Object Type ID	Audio Object Type	gain control	block switching	window shapes - standard	window shapes – AAC LD	filterbank - standard	filterbank - SSR	TNS	LTP	intensity	coupling	frequency deomain 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 4kbit/s VR	SA tools	SASBF	MIDI	HILN	TTSI	SBR	Layer-1	Layer-2	Layer-3	SSC (Transient, Sinusoid, Noise)	Parametric stereo	DST	ALS	Remark
0	Null																																								
1	AAC main		X	X		X		X		X	X	X	X	X				X																							2)
2	AAC LC		X	X		X		X		X	X		X	X				X																							2)
3	AAC SSR	X	X	X			X	X		X	X		X	X				X																							
4	AAC LTP		X	X		X		X	X	X	X		X	X				X																							2)
5	SBR																																X								
6	AAC Scalable		X	X		X		X	X	X			X	X	X	X	X	X																							6)
7	TwinVQ		X	X		X		X	X					X					X																						
8	CELP																							X																	
9	HVXC																									X															
10	(reserved)																																								
11	(reserved)																																								
12	TTSI																															X									
13	Main synthetic																											X	X	X											3)
14	Wavetable synthesis																												X	X											4)
15	General MIDI																													X											
16	Algorithmic Synthesis and Audio FX																											X													
17	ER AAC LC		X	X		X		X		X			X	X				X			X	X	X																		
18	(reserved)																																								
19	ER AAC LTP		X	X		X		X	X	X			X	X				X			X	X	X																		5)
20	ER AAC scalable		X	X		X		X		X			X	X	X	X	X	X			X	X	X																		6)
21	ER TwinVQ		X	X		X		X						X					X			X	X																		
22	ER BSAC		X	X		X		X		X			X	X						X		X	X																		
23	ER AAC LD				X	X		X	X	X			X	X				X			X	X	X																		
24	ER CELP																					X	X	X	X																
25	ER HVXC																					X	X			X	X														
26	ER HILN																					X	X								X										
27	ER Parametric																					X	X			X	X				X										
28	SSC																																				X	X			
29	PS																																X					X			
30	(reserved)																																								
31	(escape)																																								
32	Layer-1																																	X							
33	Layer-2																																		X						
34	Layer-3																																			X					
35	DST																																						X		
36	ALS																																							X	
37 - 95	(reserved)																																								

In Part 3: Audio, Subpart 1, in subclause 1.5.1.2 Description, add:

1.5.1.2.30 ALS object type

The ALS object type is the counterpart of the Audio Lossless Coding (ALS) scheme and contains the corresponding ALS tools.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

In Part 3: Audio, Subpart 1, replace Table 1.3 (Audio Profiles definition) with the following table:

Table 1.3 – Audio Profiles definition

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											
15	General MIDI											
16	Algorithmic Synthesis and Audio FX											
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	(reserved)											
31	(escape)											
32	Layer-1											
33	Layer-2											
34	Layer-3											
35	DST											
36	ALS											

In Part 3: Audio, Subpart 1, subclause 1.5.2.3 (Levels within the profiles), add at the end:

- **Levels for the High Efficiency AAC v2 Profile**

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

Table 1.11A - Levels for the High Efficiency AAC v2 Profile

Level	Max. channels/ object	Max. AAC sampling rate, SBR not present [kHz]	Max. AAC sampling rate, SBR present [kHz]	Max. SBR sampling rate [kHz] (in/out)	Max. PCU	Max. RCU	Max. PCU HQ / LP SBR (Note 5)	Max. RCU HQ / LP SBR (Note 5)
1	NA	NA	NA	NA	NA	NA	NA	NA
2	2	48	24	24/48 (Note 1)	9	10	9	10
3	2	48	24/48 (Note 3)	48/48 (Note 2)	15	10	15	10
4	5	48	24/48 (Note 4)	48/48 (Note 2)	25	28	20	23
5	5	96	48	48/96	49	28	39	23

Note 1: A level 2 HE AAC v2 Profile decoder implements the baseline version of the parametric stereo tool. Higher level decoders shall not be limited to the baseline version of the parametric stereo tool.
Note 2: For level 3 and level 4 decoders, it is mandatory to operate the SBR tool in downsampled mode if the sampling rate of the AAC core is higher than 24kHz. Hence, if the SBR tool operates on a 48kHz AAC signal, the internal sampling rate of the SBR tool will be 96kHz, however, the output signal will be downsampled by the SBR tool to 48kHz.
Note 3: If Parametric Stereo data is present the maximum AAC sampling rate is 24kHz, if Parametric Stereo data is not present the maximum AAC sampling rate is 48kHz.
Note 4: For one or two channels the maximum AAC sampling rate, with SBR present, is 48kHz. For more than two channels the maximum AAC sampling rate, with SBR present, is 24kHz.
Note 5: The PCU/RCU number are given for a decoder operating the LP SBR tool whenever applicable.

A HE AAC v2 Profile decoder of a certain level shall operate the HQ SBR tool for streams containing Parametric Stereo data. For streams not containing Parametric Stereo data, the HE AAC v2 Profile decoder may operate the HQ SBR tool, or the LP SBR tool.

In Part 3: Audio, Subpart 1, subclause 1.5.2.4 (Table 1.12 - audioProfileLevelIndication Values), replace the row:

0x30-0x7F	reserved for ISO use	-

with:

0x28	AAC Profile	L1
0x29	AAC Profile	L2
0x2A	AAC Profile	L4
0x2B	AAC Profile	L5
0x2C	High Efficiency AAC Profile	L2
0x2D	High Efficiency AAC Profile	L3
0x2E	High Efficiency AAC Profile	L4
0x2F	High Efficiency AAC Profile	L5
0x30	High Efficiency AAC v2 Profile	L2
0x31	High Efficiency AAC v2 Profile	L3
0x32	High Efficiency AAC v2 Profile	L4
0x33	High Efficiency AAC v2 Profile	L5
0x34-0x7F	reserved for ISO use	-

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

In Part 3: Audio, Subpart 1, in subclause 1.6.2.1 AudioSpecificConfig, replace table 1.13 with the table below:

Table 1.13 — Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic
AudioSpecificConfig ()		
{		
audioObjectType = GetAudioObjectType();		
samplingFrequencyIndex;	**4**	**bslbf**
if (samplingFrequencyIndex == 0xf) {		
samplingFrequency;	**24**	**uimsbf**
}		
channelConfiguration;	**4**	**bslbf**
sbrPresentFlag = -1;		
psPresentFlag = -1;		
if (audioObjectType == 5 ||		
audioObjectType == 29) {		
extensionAudioObjectType = 5;		
sbrPresentFlag = 1;		
if (audioObjectType == 29) {		
psPresentFlag = 1;		
}		
extensionSamplingFrequencyIndex;	**4**	**uimsbf**
if (extensionSamplingFrequencyIndex == 0xf) {		
extensionSamplingFrequency;	**24**	**uimsbf**
}		
audioObjectType = GetAudioObjectType();		
}		
else {		
extensionAudioObjectType = 0;		
}		
switch (audioObjectType) {		
case 1:		
case 2:		
case 3:		
case 4:		
case 6:		
case 7:		
case 17:		
case 19:		
case 20:		
case 21:		
case 22:		
case 23:		
GASpecificConfig();		
break:		
case 8:		
CelpSpecificConfig();		
break;		
case 9:		
HvxcSpecificConfig();		
break:		
case 12:		
TTSSpecificConfig();		
break;		

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

```
    case 13:
    case 14:
    case 15:
    case 16:
        StructuredAudioSpecificConfig();
        break;
    case 24:
        ErrorResilientCelpSpecificConfig();
        break;
    case 25:
        ErrorResilientHvxcSpecificConfig();
        break;
    case 26:
    case 27:
        ParametricSpecificConfig();
        break;
    case 28:
        SSCSpecificConfig();
        break;
    case 32:
    case 33:
    case 34:
        MPEG_1_2_SpecificConfig();
        break;
    case 35:
        DSTSpecificConfig();
        break;
    case 36:
        ALSSpecificConfig();
        break;
    default:
        /* reserved */
    }
    switch (audioObjectType) {
    case 17:
    case 19:
    case 20:
    case 21:
    case 22:
    case 23:
    case 24:
    case 25:
    case 26:
    case 27:
        epConfig;                                   2       bslbf
        if ( epConfig == 2 || epConfig == 3 ) {
            ErrorProtectionSpecificConfig();
        }
        if ( epConfig == 3 ) {
            directMapping;                          1       bslbf
            if ( ! directMapping ) {
                /* tbd */
            }
        }
    }
    if ( extensionAudioObjectType != 5 && bits_to_decode() >= 16 ) {
        syncExtensionType;                          11      bslbf
        if (syncExtensionType == 0x2b7) {
        extensionAudioObjectType = GetAudioObjectType();
```


© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

```
            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
                        }
                    }
                }
            }
        }
    }
}
```

In Part 3: Audio, Subpart 1, in subclause 1.6.2.1 AudioSpecificConfig, add:

1.6.2.1.12 ALSSpecificConfig

Defined in ISO/IEC 14496-3 subpart 11.

In Part 3: Audio, Subpart 1, in subclause 1.6.2.2.1 Overview, replace table 1.15 by the following table:

Table 1.15 – 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	

Documents
Document Sharing Hub

© ISO/IEC 2006 – All rights reserved

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

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	
PS	29	ISO/IEC 14496-3 subpart 8	
(reserved)	30		
(escape)	31		
Layer-1	32	ISO/IEC 14496-3 subpart 9	
Layer-2	33	ISO/IEC 14496-3 subpart 9	
Layer-3	34	ISO/IEC 14496-3 subpart 9	
DST	35	ISO/IEC 14496-3 subpart 10	
ALS	36	ISO/IEC 14496-3 subpart 11	

In Part 3: Audio, Subpart 1, under 1.6.3 Semantics, after 1.6.3.13 extensionAudioObjectType add:

1.6.3.14 psPresentFlag

A one bit field indicating the presence or absence of Parametric Stereo data. The value –1 indicates that the psPresentFlag was not conveyed in the AudioSpecificConfig(). In this case, a High Efficiency AAC v2 Profile decoder shall support implicit signaling (see subclause 1.6.6).

In Part 3: Audio, Subpart 1, after 1.6.5 Signaling of SBR, add the following subclause:

1.6.6 Signaling of Parametric Stereo (PS)

1.6.6.1 Generating and Signaling HE AAC + PS Content

The PS tool in combination with the HE AAC coder enables good stereo quality at very low bitrates. At the same time it allows for compatibility with existing HE AAC-only decoders. However, the output from a HE AAC decoder will only be mono for a HE AAC v2 stream carrying PS data.

Therefore, depending on the application, a content provider or content creator may want to choose between the two alternatives given below. In general, the PS data is always embedded in the HE AAC stream in a HE AAC compatible way (in the sbr_extension element), and PS is a pure post processing step in the decoder. Therefore, compatibility can be achieved. However, by means of different signaling the content creator can select between the full-quality mode and the backward compatibility mode as outlined in 1.6.6.1.1 and 1.6.6.1.2.

For the hierarchical profiles, a profile higher in the profile hierarchy is of course able to decode the content of a profile lower in the profile hierarchy. In Figure 1.0A the hierarchical structure of the AAC, HE AAC and HE AAC v2 Profile is displayed. The figure shows that a HE AAC Profile decoder is fully capable of decoding any AAC-Profile stream, given that the HE AAC Profile decoder is of the same or a higher level as indicated in the AAC Profile stream. Similarly the HE AAC v2 decoder can handle all HE AAC Profile streams as well as all AAC Profile streams.

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

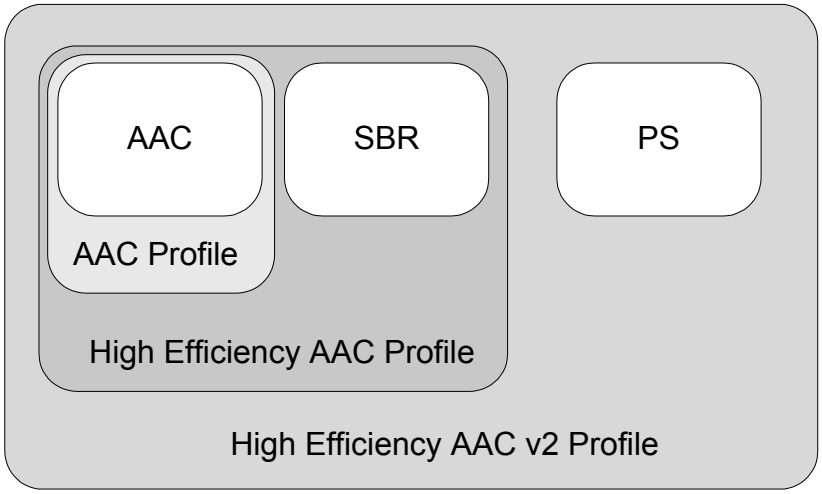

Figure 1.0A – Hierarchical structure of AAC, HE AAC and HE AAC v2 Profile, and compatibility between them.

1.6.6.1.1 Ensuring Full Audio Quality of AAC+SBR+PS for the Listener

To ensure that listeners get the full audio quality of AAC+SBR+PS, the stream should indicate the HE AAC v2 Profile and use the explicit, hierarchical signaling (signaling 2.A. as described below), so that it is played by HE AAC v2 Profile decoders, i.e., PS capable decoders. With regard to HE AAC-only streams or AAC-only streams, an HE AAC v2 Profile decoder will decode all HE AAC Profile streams and AAC Profile streams of the appropriate level, as the HE AAC v2 Profile is a superset of the HE AAC Profile and the AAC Profile.

1.6.6.1.2 Achieving Backward Compatibility with Existing HE AAC and AAC Decoders

The aim of this mode is to get all AAC-based and HE AAC-based decoders to play the stream, even if they do not support the PS tool. Compatible streams can be created using the following two signaling methods:

a) indicate a profile containing SBR (e.g. the HE AAC Profile), but not the HE AAC v2 Profile, and use the explicit backward compatible signalling (2.B. as described below). This method is recommended for all MPEG-4 based systems in which the length of the AudioSpecificConfig() is known in the decoder. As this is not the case for LATM with audioMuxVersion==0 (see clause 1.7), this method cannot be used for LATM with audioMuxVersion==0. In explicit backward compatible signaling, PS-specific configuration data is added at the end of the AudioSpecificConfig(). Decoders that do not know about PS will ignore these parts, while HE AAC v2 Profile decoders will detect its presence and configure the decoder accordingly.

b) indicate a profile containing SBR (e.g. the HE AAC Profile), but not the HE AAC v2 Profile, and use implicit signalling. In this mode, there is no explicit indication of the presence of PS data. Instead, HE AAC v2 Profile decoders shall open two output channels for a stream containing SBR data with channelConfiguration==1, i.e., a mono stream using a single channel element, and check the presence of PS data while decoding the stream and use the PS tool if PS data is found. This is possible because PS can be decoded without PS-specific configuration data if a certain way of handling decoder number of output channels is obeyed, as described below for HE AAC v2 Profile decoders.

Both methods lead to the result that, provided that the profile indication indicates a profile supported by the decoder, the AAC+SBR part of an AAC+SBR+PS streams will be decoded by HE AAC-only decoders, and the AAC part of an AAC+SBR+PS stream will be decoded by AAC-only decoders. HE AAC v2 decoders will detect the presence of PS and decode the full quality AAC+SBR+PS stream.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

1.6.6.2 Implicit and Explicit Signaling of Parametric Stereo

This subclause outlines the different signaling methods of PS, and the decoder behavior for different types of signaling.

There are several ways to signal the presence of PS data:

1. **implicit signaling:** If bs_extension_id equals EXTENSION_ID_PS, PS data is present in the sbr_extension element, and this implicitly signals the presence of PS data. The ability to detect and decode implicitly signaled PS is mandatory for all High Efficiency AAC v2 Profile (HE AAC v2 Profile) decoders.

2. **explicit signaling:** The presence of PS data is signaled explicitly by means of the PS Audio Object Type and the psPresentFlag in the AudioSpecificConfig(). When explicit signaling of PS is used, implicit signaling of PS shall not occur. Two different types of explicit signaling are available:

   2.A. **hierarchical signaling:** If the first audioObjectType (AOT) signaled is the PS AOT, the extensionAudioObjectType is set to SBR, and a second audio object type is signaled which indicates the underlying audio object type. This signaling method is not backward compatible. This method may be needed in systems that do not convey the length of the AudioSpecificConfig(), such as LATM with audioMuxVersion==0, and content authors are encouraged to use it only when thus needed.

   2.B. **backward compatible signaling:** If the extensionAudioObjectType SBR is signaled at the end of the AudioSpecificConfig(), a psPresentFlag is transmitted at the end of the backward compatible explicit SBR signaling, indicating the presence or absence of PS data. 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.

For all types of parametric stereo signaling, the channelConfiguration in the audioSpecifcConfig indicates the number of channels of the underlying AAC coded stream. Hence, if parametric stereo data is available, the channelConfiguration will be one, indicating a single channel element, while the parametric stereo tool will produce two output channels based on the single channel element and the parametric stereo data.

Table 1.22A shows the decoder behavior depending on profile and audio object type indication when implicit or explicit signaling is used.

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

Table 1.22A – PS Signaling and Corresponding Decoder Behavior

Bitstream characteristics				Decoder behavior	
Profile indication	**PS signaling**	**psPresent Flag**	**raw_data_block**	**HE AAC Profile Decoders**	**HE AAC v2 Profile Decoders**
High Efficiency AAC Profile	signaling 1, implicit signaling (first AOT != PS)	-1	AAC+SBR	Play AAC+SBR	Play AAC+SBR (Note 1)
			AAC+SBR+PS	Play AAC+SBR	Play at least AAC+SBR, should play AAC+SBR+PS (Note 1)
	signaling 2.B, backwards compatible explicit signaling (second AOT == SBR)	0	AAC+SBR	Play AAC+SBR	Play AAC+SBR (Note 2)
		1	AAC+SBR+PS	Play AAC+SBR	Play at least AAC+SBR, should play AAC+SBR+PS (Note 3)
High Efficiency AAC v2 Profile	signaling 2.A, non-backwards compatible signaling (first AOT == PS)	1	AAC+SBR+PS	Undefined	Play AAC+SBR+PS (Note 3)
	signaling 2.B, backwards compatible signling (second AOT == SBR)	1	AAC+SBR+PS	Undefined	Play AAC+SBR+PS (Note 3)
Note 1: Implicit signaling, assume the presence of PS data in the payload, giving two output channels for a single channel element. Note 2: Explicitly signals that there is no PS data, hence no implicit signaling is present. Note 3: Number of output channels is two for a single channel element containing AAC+SBR+PS data.					

The upper part of Table 1.22A displays bitstream characteristics and decoder behavior if the profile indication is the High Efficiency AAC Profile. The lower part displays bitstream characteristics and decoder behavior if the profile indication is the High Efficiency AAC v2 Profile.

1.6.6.3 HE AAC v2 Profile Decoder Behavior in Case of Implicit Signaling

If the presence of PS data is backward compatible implicitly signaled (signaling 1, in the list above) the first AudioObjectType signaled is not the PS AOT, and the psPresentFlag is not read from the AudioSpecificConfig(). Hence, the psPresentFlag is set to –1, indicating that implicit signaling of parametric stereo may occur.

Since a received mono stream will result in a stereo output if Parametric Stereo data is present in the stream, the HE AAC v2 Profile decoder shall assume that PS data is available and decide the number of output channels to be two for a single channel element containing SBR data, and thus also possibly PS data. If no PS data is found the mono output shall be mapped to the two opened channels for every single channel element.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

1.6.6.4 HE AAC v2 Profile Decoder Behavior in Case of Explicit Signaling

If the presence of PS data is explicitly signaled (signaling 2, in the list above) the presence of PS data is backward compatible explicitly signaled (signaling 2.B) or non-backward compatible explicitly signaled (signaling 2.A).

For the backward compatible explicit signaled (signaling 2.B) the extensionAudioObjectType signaled is the SBR AOT. The explicit signaling of PS is done by means of the psPresentFlag that can be either zero or one.

If the psPresentFlag is zero, this indicates that PS data is not present, and hence the HE AAC v2 Profile decoder should not make assumptions on the number of output channels in anticipation of PS data (as in case of implicit signaling of PS) and instead employ the original channelConfiguration. If the psPresentFlag is one, PS data is present and the HE AAC v2 Profile decoder shall operate the PS Tool.

For the non-backward compatible explicit signaling of PS (signaling 2.A) the first AudioObjectType signaled is the PS AOT. The extensionAudioObjectType is assigned the SBR AOT. For this hierarchical explicit signaling, the psPresentFlag is set to one if the first signaled AOT is the PS AOT. The psPresentFlag is not transmitted and hence it is not possible to explicitly signal the absence of implicit signaling. Hence, for the hierarchical explicit signaling of parametric stereo, PS data is always present and the HE AAC v2 Profile decoder shall operate the PS Tool.

In Part 3: Audio, Subpart 4, in subclause 4.4.2.6 Payloads for the audio object type ER BSAC, replace table 4.33 bsac_raw_data_block with the following table:

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
syncword
while(data_available())
extended_bsac_raw_data_block();
}
}
	
	32
	8
		
		bslbf
		bslbf

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

In Part 3: Audio, Subpart 4, in subclause 4.4.2.6 Payloads for the audio object type ER BSAC, after Table 4.43 Syntax of bsac_spectral_data, add the following two tables:

Table 4.35 – Syntax of extended_bsac_raw_data_block()

Syntax	No. of bits	Mnemonic

extended_bsac_raw_data_block()
{
extended_bsac_base_element();
layer=slayer_size;
while(data_available() && layer<(top_layer+slayer_size)) {
bsac_layer_element(layer);
layer++;
}
byte_alignment();
}		

Table 4.36 – Syntax of extended_bsac_base_element()

Syntax	No. of bits	Mnemonic
extended_bsac_base_element() {		
element_length	**11**	**uimbf**
channel_configuration_index	**3**	**uimbf**
reserved_bit	**1**	**uimbf**

bsac_header();
general_header();
byte_alignment();
for (slayer = 0; slayer < slayer_size; slayer++)
bsac_layer_element(slayer);
}		

In Part 3: Audio, Subpart 4, under Bitstream elements in subclause 4.5.2.6.2.1 Definitions, replace bsac_raw_data_block with the following:

bsac_raw_data_block() — block of raw data that contains coded audio data, related information and other data. A bsac_raw_data_block() basically consists of bsac_base_element() and several bsac_layer_element(). There exists a module that determines whether the BSAC bitstream has an extended part.

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

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

syncword — a eight bit code that identifies the start of the extended part. The bit string '1111 1111'.

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

In Part 3: Audio, Subpart 4, under Bitstream elements in subclause 4.5.2.6.2.1 Definitions, replace header_length with the following:

header_length — the length of the headers including frame_length, bsac_header() and general_header() in bytes. The actual length is (header_length+7) bytes. However if header_length is 0, it represents that the actual length is smaller than or equal to 7 bytes. And if header_length is 15, it represents that the actual length is larger than or equal to (15+7) bytes and should be calculated through the decoding of the headers. In case of extended_bsac_base_element(), header_length includes element_length, channel_configuration_index, reserved_bit, bsac_header and general_header().

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

extended_bsac_raw_data_block() — block of raw data that contains coded audio data, related information and other data for the extended part. A extended_bsac_ raw_data _block() basically consists of extended_ bsac _base_ element() and several bsac_layer_element().

extended_bsac_base_element() — syntactic element of the base layer bitstream containing coded audio data, related information and other data for the extended part of BSAC.

element_length — the length of the extended_bsac_raw_data_block() in bytes. This is used for proper arithmetic decoding.

channel_configuration_index — a three bit field that indicates the audio output channel configuration in the extended part. Each index specifies the number of channels given the channel to speaker mapping.

Table 4.68 – channel_configuration_index

Index	channel to speaker mapping	number of channels (nch)
0	center front speaker	1
1	left, right front speakers	2
2	rear surround speakers	1
3	left surround, right surround rear speakers	2
4	front low frequency effects speaker	1
5	left, right outside front speakers	2
6-7	reserved	-

reserved_bit — bit reserved for future use

In Part 3: Audio, Subpart 4, after subclause 4.5.2.6.2.2.13 Reconstruction of the decoded sample from bit-sliced data, add the subclause below:

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. New functions called extended_bsac_raw_data_block and extended_bsac_base_element are added for the extended BSAC.

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

4.5.2.6.2.2.14.1 extended_bsac_raw_data_block

An extended_bsac_raw_data_block also has the layered structure as bsac_raw_data_block. In case where data is still available after decoding the stereo part, **zero_code** and **syncword** are parsed. **zero_code** is used for the arithmetic termination of stereo part, and **syncword** is for the proper decoding of extended part.

4.5.2.6.2.2.14.2 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 1.8 (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 4.68. Each index indicates the number of channels given the channel to speaker mapping.

In Part 3: Audio, Subpart 4, at the end of subclause 4.B.17.8 Payload transmitted over Elementary Steam bit-sliced data, add the following subclause:

4.B.17.8.1 The functionality of fine-grain scalability in extended or multi-channel data

When the BSAC data extends to multi-channel data, each ES consists of large-step layers for a certain channel element. To provide the functionality of fine-grain scalability in the multi-channel data, one might use *streamPriority* specified in the ES descriptor in ISO/IEC 14496-1:2004. The values of *streamPriority* are assigned to elementary streams according to the priority of channel elements. Different numbers of layers per channel element can be truncated, because the extended BSAC bitstream consists of separate channel elements. The values of *streamPriority* and the number of layers to be truncated per channel element depend on application scenarios.

In Part 3: Audio, Subpart 8, in clause 8.A.1, replace:

The usage of this parametric stereo extension to HE AAC is signalled implicitly in the bitstream. Hence, if

with:

The usage of this parametric stereo extension to HE AAC is signalled either implicitly by the presence of parametric stereo data in the bitstream, or explicitly by signalling the corresponding AudioObjectType in the audioSpecificConfig. Hence, implicit signalling requires that, if

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

Create Part 3: Audio, Subpart 11:

Subpart 11: Technical description of Audio Lossless Coding for lossless coding of audio signals

11.1 Scope

This subpart of ISO/IEC 14496-3 describes the MPEG-4 Audio Lossless Coding (ALS) algorithm for lossless coding of audio signals.

MPEG-4 ALS is a lossless compression scheme for digital audio data, i.e. the decoded data is a bit-identical reconstruction of the original input data. Input signals can be integer PCM data with 8 to 32-bit word length or 32-bit IEEE floating-point data. MPEG-4 ALS provides a wide range of flexibility in terms of compression-complexity trade-off, since the combination of several tools allows for the definition of compression levels with different complexities.

11.2 Technical Overview

11.2.1 Encoder and Decoder Structure

The basic structure of the ALS encoder and decoder is shown in Figure 11.1.

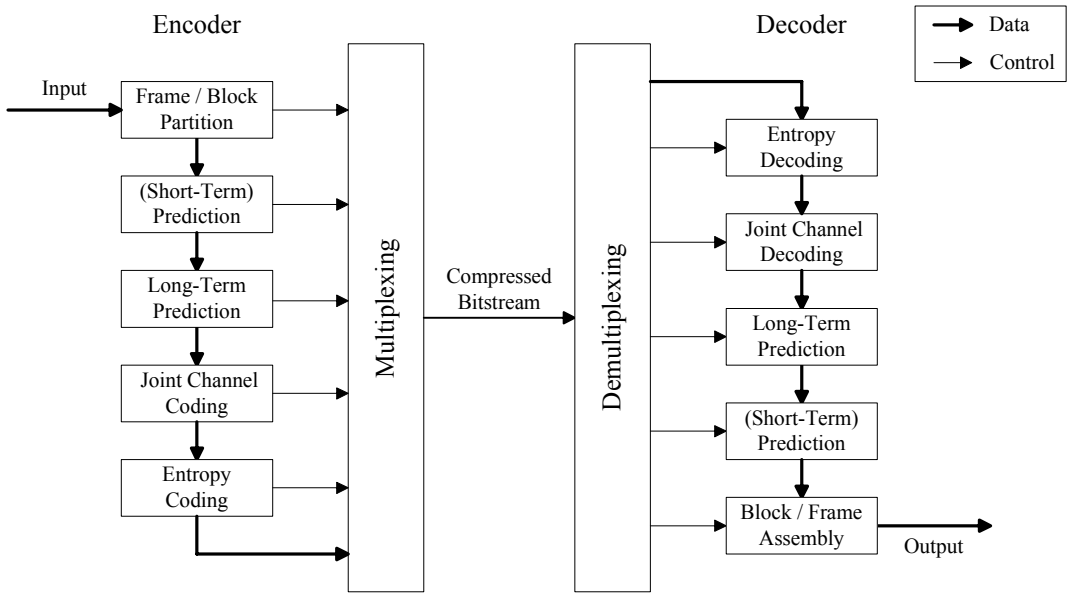

Figure 11.1 – Block diagram of the ALS encoder and decoder

The input audio data is partitioned into frames. Within a frame, each channel can be further subdivided into blocks of audio samples for further processing (*block switching*, see subclause 11.6.2). For each block, a prediction residual is calculated using short-term *prediction* (see subclauses 11.6.3 and 11.6.5) and optionally *long-term prediction* (*LTP*, see sublause 11.6.4). Inter-channel redundancy can be removed by *joint channel coding*, using either *difference coding of channel pairs* (see subclause 11.6.7) or *multi-channel coding* (*MCC*, see subclause 11.6.8). The remaining prediction residual is finally *entropy coded* (see subclause 11.6.6).

The encoder generates bitstream information allowing for random access at intervals of several frames. The encoder can also provide a CRC checksum, which the decoder may use to verify the decoded data.

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

11.2.2 Floating-Point Extensions

In addition to integer audio signals, MPEG-4 ALS also supports lossless compression of audio signals in the IEEE 32-bit floating-point format. The floating-point sequence is modeled by the sum of an integer sequence multiplied by a constant (ACF: Approximate Common Factor) and a residual sequence. The integer sequence is compressed using the basic ALS tools for integer data, while the residual sequence is separately compressed by the masked Lempel-Ziv tool. A detailed description of the floating-point extensions can be found in subclause 11.6.9.

11.3 Terms and Definitions

11.3.1 Definitions

The following definitions and abbreviations are used in this document.

Frame	Segment of the audio signal (containing all channels).
Block	Segment of one audio channel.
Sub-block	Subpart of a block that uses the same entropy coding parameters.
Random Access Frame	Frame that can be decoded without decoding previous frames.
Residual	Prediction error, i.e. original minus predicted signal.
Predictor/Prediction Filter	Linear FIR filter which computes an estimate of the input signal using previous samples.
Prediction order	Order of the prediction filter (number of predictor coefficients).
LPC coefficients	Coefficients of the direct form prediction filter.
Parcor coefficients	Parcor representation of the predictor coefficients.
Quantized coefficients	Quantized parcor coefficients.
LTP	Long-term prediction.
Rice code	Also known as Golomb-Rice code. In this document the short form is used.
BGMC	Block Gilbert-Moore Code (also known as Elias-Shannon-Fano code).
CRC	Cyclic Redundancy Check.
LPC	Linear Predictive Coding.
PCM	Pulse Code Modulation.
Mantissa	Fractional part of floating-point data
Exponent	Exponential part of floating-point data
ACFC	Approximate Common Factor Coding
Masked-LZ	Masked Lempel-Ziv Coding
MCC	Multi-Channel Coding
MSB	Most significant bit
LSB	Least significant bit

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

11.3.2 Mnemonics

uimsbf	Unsigned integer, most significant bit first
simsbf	Signed integer, most significant bit first
bslbf	Bit string, left bit first, where “left” is the order in which bits are written
IEEE32	IEEE 32-bit floating-point data (4 bytes), most significant bit first

The mnemonics Rice code and BGMC indicate that variable length codewords are used, which are described in subclause 11.6.6.

11.3.3 Data Types

The following data types are used in the pseudo code sections:

INT64	64-bit signed integer (two's complement)
long	32-bit signed integer (two's complement)
short	16-bit signed integer (two's complement)

If "unsigned" is added in front of the data type, then the type is unsigned instead of signed.

11.4 Syntax

11.4.1 Decoder Configuration

Table 11.1 – Syntax of ALSSpecificConfig

Syntax	No. of bits	Mnemonic
ALSSpecificConfig()		
{		
samp_freq;	**32**	**uimsbf**
samples;	**32**	**uimsbf**
channels;	**16**	**uimsbf**
file_type;	**3**	**uimsbf**
resolution;	**3**	**uimsbf**
floating;	**1**	**uimsbf**
msb_first;	**1**	**uimsbf**
frame_length;	**16**	**uimsbf**
random_access;	**8**	**uimsbf**
ra_flag;	**2**	**uimsbf**
adapt_order;	**1**	**uimsbf**
coef_table;	**2**	**uimsbf**
long_term_prediction;	**1**	**uimsbf**
max_order;	**10**	**uimsbf**
block_switching;	**2**	**uimsbf**
bgmc_mode;	**1**	**uimsbf**
sb_part;	**1**	**uimsbf**
joint_stereo;	**1**	**uimsbf**
mc_coding;	**1**	**uimsbf**
chan_config;	**1**	**uimsbf**
chan_sort;	**1**	**uimsbf**
crc_enabled;	**1**	**uimsbf**
RLSLMS	**1**	**uimsbf**

AVA Documents Document Sharing Hub

© ISO/IEC 2006 – All rights reserved

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

(reserved)	**5**	
aux_data_enabled;	**1**	**uimsbf**
if (chan_config) {		
chan_config_info;	**16**	**uimsbf**
}		
if (chan_sort) {		
for (c = 0; c < channels; c++)		
chan_pos[c];	**1..16**	**uimsbf**
}		
byte_align;		
header_size;	**16**	**uimsbf**
trailer_size;	**16**	**uimsbf**
orig_header[];	**header_size * 8**	**bslbf**
orig_trailer[];	**trailer_size * 8**	**bslbf**
if (crc_enabled) {		
crc;	**32**	**uimsbf**
}		
if ((ra_flag == 2) && (random_access > 0)) {		
for (f = 0; f < ((samples-1) / (frame_length+1)) + 1; f++) {		
ra_unit_size[f]	**32**	**uimsbf**
}		
}		
if (aux_data_enabled) {		
aux_size;	**16**	**uimsbf**
aux_data[];	**aux_size * 8**	**bslbf**
}		
}		

11.4.2 Bitstream Payloads

Table 11.2 – Syntax of top level payload (frame_data)

Syntax	No. of bits	Mnemonic
frame_data()		
{		
if ((ra_flag == 1) && (frame_id % random_access == 0)) {		
ra_unit_size	**32**	**uimsbf**
}		
if (mc_coding && joint_stereo) {		
js_switch;	**1**	**uimsbf**
byte_align;		
}		
if (!mc_coding || js_switch) {		
for (c = 0; c < channels; c++) {		
if (block_switching) {		
bs_info;	**8,16,32**	**uimsbf**
}		
if (independent_bs) {		
for (b = 0; b < blocks; b++) {		
block_data(c);		
}		
}		
else{		
for (b = 0; b < blocks; b++) {		
block_data(c);		
block_data(c+1);		
}		
c++;		

Documents
Document Sharing Hub

© ISO/IEC 2006 – All rights reserved

Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

}
}
else{
if (block_switching) {
bs_info;
}
for (b = 0; b < blocks; b++) {
for (c = 0; c < channels; c++) {
block_data(c);
channel_data(c);
}
}
}
if (floating)
{
num_bytes_diff_float;
diff_float_data();
}
}
	8,16,32
	
	32
		uimsbf
		
		uimsbf

Note: If joint_stereo is off, or if c is the last channel, independent_bs is true by default. If joint_stereo is on, independent_bs is false by default, but if block_switching is on as well, the independent_bs flag is explicitly signaled as the first bit of a channel pair's bs_info field (see subclause 11.6.2). The frame_id field indicates the consecutive frame number, starting at 0 for the first frame.

Table 11.3 – Syntax of block_data

Syntax	No. of bits	Mnemonic

block_data()
{
block_type;
if (block_type == 0) {
const_block;
js_block;
(reserved)
if (const_block == 1) {
{
if (resolution == 8) {
const_val;
}
else if (resolution == 16) {
const_val;
}
else if (resolution == 24) {
const_val;
}
else {
const_val;
}
}
}
else {
js_block;
if ((bgmc_mode == 0) && (sb_part == 0) {
sub_blocks = 1;
}
else if ((bgmc_mode == 1) && (sb_part ==1) {
ec_sub;
sub_blocks = 1 << ec_sub;
}
	
	1
	
	1
	1
	5
	
	8
	
	16
	
	24
	
	32
	
	1
	
	2
		
		uimsbf
		
		uimsbf
		uimsbf
		
		simsbf
		
		simsbf
		
		simsbf
		
		simsbf
		
		uimsbf
		
		uimsbf

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

else {		
ec_sub;	**1**	**uimsbf**
sub_blocks = (ec_sub == 1) ? 4 : 1;		
}		
if (bgmc_mode == 0) {		
for (k = 0; k < sub_blocks; k++) {		
s[k];	**varies**	**Rice code**
}		
}		
else {		
for (k = 0; k < sub_blocks; k++) {		
s[k],sx[k];	**varies**	**Rice code**
}		
}		
sb_length = block_length / sub_blocks;		
shift_lsbs;	**1**	**uimsbf**
if (shift_lsbs == 1) {		
shift_pos;	**4**	**uimsbf**
}		
if (!RLSLMS) {		
if (adapt_order == 1) {		
opt_order;	**1..10**	**uimsbf**
}		
for (p = 0; p < opt_order; p++) {		
quant_cof[p];	**varies**	**Rice code**
}		
}		
if (long_term_prediction) {		
LTPenable;	**1**	**uimsbf**
if (**LTPenable**) {		
for (i = -2; i <= 2; i++) {		
LTPgain[i];	**varies**	**Rice code**
}		
LTPlag;	**8,9,10**	**uimsbf**
}		
}		
start = 0;		
if (random_access_block) {		
if (opt_order > 0) {		
smp_val[0];	**varies**	**Rice code**
}		
if (opt_order > 1) {		
res[1];	**varies**	**Rice code**
}		
if (opt_order > 2) {		
res[2];	**varies**	**Rice code**
}		
if (opt_order < 3) {		
start = opt_order;		
}		
else {		
start = 3;		
}		
}		
if (bgmc_mode) {		
for (n = start; n < sb_length; n++) {		
msb[n];	**varies**	**BGMC**
}		
for (k=1; k < sub_blocks; k++) {		

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

for (n = k * sb_length; n < (k+1) * sb_length; n++) {		
msb[n];	**varies**	**BGMC**
}		
}		
for (n = start; n < sb_length; n++) {		
if (msb[n] != tail_code) {		
lsb[n];	**varies**	**uimsbf**
}		
else {		
tail[n];	**varies**	**Rice code**
}		
}		
for (k=1; k < sub_blocks; k++) {		
for (n = k * sb_length; n < (k+1) * sb_length; n++) {		
if (msb[n] != tail_code) {		
lsb[n];	**varies**	**uimsbf**
}		
else {		
tail[n];	**varies**	**Rice code**
}		
}		
}		
}		
else		
{		
for (n = start; n < block_length; n++) {		
res[n];	**varies**	**Rice code**
}		
}		
}		
if (RLSLMS) {		
RLSLMS_extension_data()		
}		
}		

Note: random_access_block is true if the current block belongs to a random access frame (frame_id % random_access == 0) and is the first (or only) block of a channel in this frame.

Table 11.4 – Syntax of channel_data

Syntax	No. of bits	Mnemonic
channel_data(c)		
{		
for(;;) {		
stop_flag;	**1**	**uimsbf**
if (stop_flag == 1) {		
break;		
}		
master_channel_index;	**1..16**	**uimsbf**
if (c != master_channel_index) {		
time_difference_flag	**1**	**uimsbf**
if (time_difference_flag == 0) {		
weighting_factor [0]	**varies**	**Rice code**
weighting_factor [1]	**varies**	**Rice code**
weighting_factor [2]	**varies**	**Rice code**
}		
else {		
weighting_factor [0]	**varies**	**Rice code**
weighting_factor [1]	**varies**	**Rice code**
weighting_factor [2]	**varies**	**Rice code**

© ISO/IEC 2006 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006

weighting_factor [3]	**varies**	**Rice code**
weighting_factor [4]	**varies**	**Rice code**
weighting_factor [5]	**varies**	**Rice code**
time_difference_sign	**1**	**uimsbf**
time_difference_index	**5,6,7**	**uimsbf**
}		
}		
}		
}		

Table 11.5 – Syntax of RLSLMS_extension_data

Syntax	No. of bits	Mnemonic
RLSLMS_extension()		
{		
mono_block	**1**	**uimsbf**
ext_mode	**1**	
if (ext_mode) {		
extension_bits	**3**	**uimsbf**
if (extension_bits&0x01) {		
RLS_order	**4**	**uimsbf**
LMS_stage	**3**	**uimsbf**
for(i=0; i<LMS_stage;i++){		
LMS_order[i]	**5**	**uimsbf**
}		
}		
if (extension_bits&0x02) {		
if (RLS_order) {		
RLS_lambda	**10**	**uimsbf**
if (RA)		
RLS_lambda_ra	**10**	**uimsbf**
}		
}		
if (extension_bits&04) {		
for(i=0; i<LMS_stage;i++) {		
LMS_mu[i]	**5**	**uimsbf**
}		
LMS_stepsize	**3**	
}		
}		
}		

11.4.3 Payloads for Floating-Point Data

Table 11.6 – Syntax of diff_float_data

Syntax	No. of bits	Mnemonic
diff_float_data()		
{		
use_acf;	**1**	**uimsbf**
if (random_access_block) {		
if (c=0; c < channels; c++) {		
last_acf_mantissa[c] = 0;		
last_shift_value[c] = 0;		
}		

© ISO/IEC 2006 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 14496-3_2005_Amd 2_2006