LIBXAAC Encoder

API Document

Document Number	LIBXAAC-Enc-API
Version	1.0
Date	June 29, 2023

Ittiam Systems (P) Ltd,
The Consulate, 1 Richmond Road,
Bangalore 560 025, India

Notice

Revision History

Version	Date	Changes
1.0	June 29, 2023	Original version

Contents

Tables

Figures

1. Introduction

1.1 Motivation

Extended HE-AAC, the latest innovation member of the MPEG AAC codec family, is ideally suited for adaptive bit rate streaming and digital radio applications. Extended HE-AAC bridges the gap between speech and audio coding and ensures consistent high-quality audio for all signal types, including speech, music, and mixed material. It is the required audio codec for DRM (Digital Radio Mondiale). When it comes to coding, the codec is incredibly effective, generating high-quality audio for music and speech at bitrates as low as 6 kbit/s for mono and 12 kbit/s for stereo services. By switching to extremely low bitrate streams, Extended HE-AAC streaming apps and streaming radio players can provide uninterrupted playback even during very congested network conditions.

As the Extended High Efficiency AAC Profile is a logical evolution of the MPEG Audio's popular AAC Family profiles, the codec supports AAC-LC, HE-AACv1 (AAC+) and HE-AACv2 (eAAC+) audio object type encoding. The bitrate that was saved with AAC family tools can be used to enhance video quality. Extended HE-AAC is a well-liked option for a number of applications since it is a strong and effective audio codec that provides high-quality audio at low bitrates.

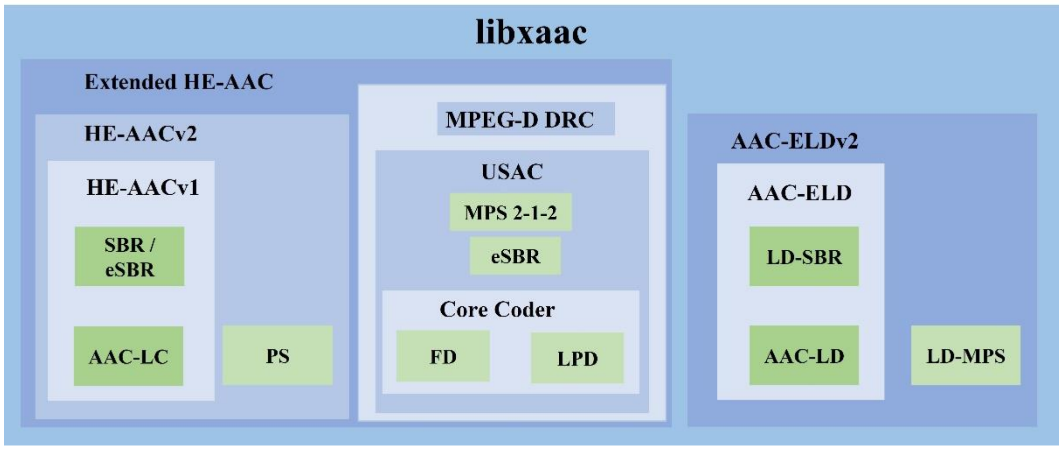

Figure 1-1 Block Diagram of libxaac

One of the key features of libxaac encoder (refer to above image) is that it has support for AAC-LD (Low Delay), AAC-ELD (Enhanced Low Delay), and AAC-ELDv2 (Enhanced Low Delay version 2) modes. AAC-LD mode provides low latency encoding, making it suitable for applications such as interactive communication and live audio streaming. It helps to reduce the delay in the encoding process to improve the real-time performance of the system. AAC-ELD mode improves the low-delay performance of HE-AAC by reducing the coding delay while maintaining high audio quality. It was observed that minimum delay it can achieve is 15ms. In order to achieve low delay coding scheme and low bitrate, it uses the Low Delay SBR tool. AAC-ELDv2 is the most advanced version of AAC-based low delay coding. It provides an enhanced version of AAC-ELD, which provides even lower coding delay and higher audio quality.

Overall, libxaac encoder, with support for AAC-LD, AAC-ELD, and AAC-ELDv2 modes, is a versatile audio coding technology that can be used for a wide range of applications, such as broadcasting, streaming, and teleconferencing which requires high-quality audio compression with minimal delay.

Also, libxaac decoder supports MPEG-D DRC(Dynamic Range Control) for USAC profile. DRC offers a bitrate efficient representation of dynamically compressed versions of an audio signal. This is achieved by adding a low-bitrate DRC metadata stream to the audio signal. DRC includes dedicated sections for clipping prevention, ducking, and for generating a fade-in and fade-out to supplement the main dynamic range compression functionality. The DRC effects available at the DRC decoder are generated at the DRC encoder side. At the DRC decoder side, the audio signal may be played back without applying DRC, or an appropriate DRC effect is selected and applied based on the given playback scenario. It offers flexible solutions to efficiently support the widespread demand for technologies such as loudness normalization and dynamic range compression for various playback scenarios.

MPEG-D USAC (along with MPEG-D DRC) support for libxaac encoder will be coming soon, which will help in improved audio quality at reduced bitrates. USAC technology will provide efficient and high quality compression of speech and audio signals, making it a valuable addition to our libxaac capabilities and features.

This document describes the **A**pplication **P**rogram **I**nterface for the libxaac encoder. It also addresses the knowledge requirements of developers to integrate different components of their system with libxaac encoder software solution.

1.2 Scope

This document discusses the following:

- Overview of API (Chapter 2)
  - This chapter gives a complete overview of the API.
  - Overview of error codes.
  - It contains some information useful for system integrator.

1.3 Glossary

Term	Explanation
API	Application Program Interface (Interface through which an application talks to functional blocks)
MPEG	Moving Picture Experts Group
AAC	Advanced Audio Coding
HE-AAC	High Efficiency Advanced Audio Coding
SBR	Spectral Bandwidth Replication
PS	Parametric Stereo
ADTS	Audio Data Transport Stream
ADIF	Audio Data Interchange Format
LOAS	Low Overhead Audio Stream
LC	Low Complexity
HQ	High Quality (SBR Encoder)
LP	Low Power (SBR Encoder)
LTP	Long Term Prediction
CELP	Code Excited Linear Prediction
IS	Intensity Stereo
MS	Mid-side Stereo
TNS	Temporal Noise Shaping
PCE	Program Configuration Element
PNS	Perceptual Noise Substitution
USAC	Unified Speech and Audio Coding
eSBR	Enhanced SBR
MPS	MPEG Surround

2. ‘C’ Application Program Interface

This chapter describes the API for the libxaac encoder implementation.

2.1 Memory Management

Ittiam audio software implementation supports a flexible memory scheme and a simple, easy to use C interface that eases the integration of the software into a larger system.

Data memory (RAM memory) usage of the audio software consists primarily of the scratch and persistent memory. The algorithm also uses an input buffer and an output buffer for communication with the external world.

Persistent Memory

This is also known as static or context. This is the state or history information that is stored across algorithm invocations. The algorithm expects that the contents of the persistent memory be unchanged by the system for the complete lifetime of the algorithm.

Scratch Memory

This is the temporary buffer used by the algorithm for processing. The contents of this memory region should be unchanged if the actual encode process is active. This region can be used freely by the system between successive calls to encode.

Input Buffer

This is the buffer used by the algorithm for accepting input. This memory region is treated as read-only by the algorithm. Before the call to the Encoder, the input buffer needs to be filled with the input data.

Output Buffer

This is the buffer to which the algorithm writes the output. This buffer needs to be made available for usage of the Encoder before its call.

2.2 ‘C’ APIs

This section lists the APIs used in the libxaac encoder implementation

2.2.1 Library Information API

The following function should be called to get the library name and version number details.

`ixheaace_get_lib_id_strings`	
Description	This API gets the encoder library name and version number details.

Syntax	`ixheaace_get_lib_id_strings (`
	`      pVOID pv_output);`

Parameters	`pv_output` Pointer to the output structure variable. The library updates the relevant output parameters of this structure.
Returns	IA_NO_ERROR.

Table 2-1 Library information API

2.2.2 Create API

The following function should be called to create the encoder instance.

`ixheaace_create`	
Description	This API gets the memory requirements size of the API. It sets the configuration parameters of the libxaac encoder. It sets the attributes of all memory types required by the application onto the memory structure. It creates necessary memories (discussed in the previous section) and sets the pointer to the memory being referred to by the index to the input value. This API also encodes header/initialization bytes as per set parameters and initializes state, configuration structure and output configuration structure.
Syntax	`ixheaace_create(pVOID pv_input, pVOID pv_output);`

Parameters	`pv_input`
	Pointer to the input structure variable, `ia_input_config`. The library gets its necessary input parameters from this structure.
	`pv_output`
	Pointer to the output structure variable, `ia_output_config`. The library updates the relevant output parameters of this structure.

`ixheaace_create`	
Returns	Error Code based on the success/failure of encoder instance creation.

Table 2-2 Create API

2.2.3 Processing API

The following function should be called for encoding.

`ixheaace_process`	
Description	This API encodes the input frame data

Syntax	`ixheaace_process (`
	`      pVOID p_ia_module_obj,`
	`      pVOID pv_input,`
	`      pVOID pv_output);`

Parameters	`p_ia_module_obj`
	Pointer to API structure object.
	`pv_input`
	Pointer to the input structure variable. The library gets its necessary input parameters from this structure.
	`pv_output`
	Pointer to the output structure variable. The library updates the relevant output parameters of this structure.

Returns	Error Code based on the success/failure of encoder processing.

Table 2-3 Processing API

2.2.4 Delete API

The following function should be called to delete the encoder instance

`ixheaace_delete`	
Description	This API frees the allocated memories for the encoder.
Syntax	`ixheaace_delete(pVOID pv_output);`
Parameters	`pv_output` Pointer to the output structure variable. The library updates the relevant output parameters of this structure.
Returns	IA_NO_ERROR.

Table 2-4 Delete API

2.3 Input Data

The input source is a stored file.

Stored File Input

File is read for input data into the input buffer. The process loop will not produce output until a valid input is received.

2.4 Configuration parameters

The encode algorithm accepts the following parameters from the user.

- PCM word size – Can only be set to 16. Default value is also 16. This value specifies the PCM bit width at the input. Please refer to **Section 2.5** for more details on error handling.
- Sampling Frequency – Can be set to values from 8000 to 96000. Default value is 44100. Please refer to **Section 2.5** for more details on error handling.
- Number of Channels – AAC-LC, HE-AACv1, HE-AACv2, AAC-LD, AAC-ELD and AAC-ELDv2 support maximum 6 channels. Please refer to **Section 2.5** for more details on error handling.
- Bitrate – Can be set to values from 8000 to 576000. Please refer to **Section 2.5** for more details on error handling.
- AOT – Audio object type, Can be set to 2 for AAC-LC, 5 for HE-AACv1(Legacy SBR), 29 for HE-AACv2, 23 for AAC-LD, 39 for AAC-ELD. Default value is 2.
- eSBR flag – Can be set to 0 or 1. When set to 1, enables eSBR in HE-AACv1 encoding. Default value is 0 (legacy SBR is used).
- ADTS flag – Can be set to 0 or 1. If set to 1, ADTS bitstream is generated and by default ADTS flag is disabled. This flag is applicable only for AAC-LC/HE-AACv1/HE-AACv2 profiles. Default value is 0.
- TNS flag - Can be set to 0 or 1. If set to 0, Temporal Noise Shaping is enabled. Default value is 1.
- MPS flag – Can be set to 0 or 1. If set to 1, MPEG-Surround is enabled. Default value is 0. This flag is applicable only when the AOT is set to 39 (AAC-ELD).
- Tree Configuration – Denotes the tree configuration for MPS. Can be set to 0 for 212 configuration, 1 for 5151 configuration, 2 for 5152 configuration and 3 for 525 configuration. Default value is 0 for stereo input and 1 for 6-channel input.
- Frame-size – Denotes the frame size (in samples) to be used by the core coder for AAC-LC / HE-AACv1 / HE-AACv2 and AAC-LD / AAC-ELD / AAC-ELDv2 profiles. Can be set to 960 or 1024 for AAC-LC / HE-AACv1 / HE-AACv2 and 480 or 512 for AAC-LD / AAC-ELD / AAC-ELDv2. Default value is 1024 for AAC-LC / HE-AACv1 / HE-AACv2 and 512 for AAC-LD / AAC-ELD / AAC-ELDv2.

- Bit-reservoir size – Denotes the maximum size of the bit-reservoir to be used for non-USAC profiles. Valid values are from -1 to 6144. Should be set to -1 to omit use of bit reservoir. Default value is 384.

2.5 Error Handling

The Encoder algorithm signals error conditions to the sample application through error-codes. The complete listing of error codes and the error handling procedure are listed down in the following sections.

2.5.1 API fatal error codes

The Encoder must be re-instantiated with appropriate correction in case of fatal errors.

Error Number	Error Code
0xFFFF8000	IA_EXHEAACE_API_FATAL_MEM_ALLOC
0xFFFF8001	IA_EXHEAACE_API_FATAL_UNSUPPORTED_AOT

Table 2-5 API fatal error codes

2.5.2 Configuration non-fatal error codes

Non-fatal error codes are generated by Encoder for invalid configuration parameters. Please refer to **Section 2.4** for information on configuration parameters.

Error Number	Error Code
0x00000800	IA_EXHEAACE_CONFIG_NONFATAL_INVALID_CONFIG
0x00000801	IA_EXHEAACE_CONFIG_NONFATAL_BITRES_SIZE_TOO_SMALL
0x00000900	IA_EXHEAACE_CONFIG_NONFATAL_MPS_INVALID_CONFIG
0x00000901	IA_EXHEAACE_CONFIG_NONFATAL_MPS_PARAM_ERROR

Table 2-6 Configuration non-fatal error codes

2.5.3 Configuration fatal error codes

The possible fatal error codes generated as a part of the Encoder indicating invalid configuration parameter are listed below . Please refer to **Section 2.4** for information on configuration parameters.

Error Number	Error Code
0xFFFF8800	IA_EXHEAACE_CONFIG_FATAL_SAMP_FREQ
0xFFFF8801	IA_EXHEAACE_CONFIG_FATAL_NUM_CHANNELS
0xFFFF8802	IA_EXHEAACE_CONFIG_FATAL_USE_STEREO_PRE_PROC
0xFFFF8803	IA_EXHEAACE_CONFIG_FATAL_QUALITY_LEVEL
0xFFFF8804	IA_EXHEAACE_CONFIG_FATAL_PCM_WDSZ
0xFFFF8805	IA_EXHEAACE_CONFIG_FATAL_AAC_CLASSIC_WITH_PS
0xFFFF8806	IA_EXHEAACE_CONFIG_FATAL_AAC_CLASSIC
0xFFFF8807	IA_EXHEAACE_CONFIG_FATAL_USE_TNS
0xFFFF8808	IA_EXHEAACE_CONFIG_FATAL_CHANNELS_MASK
0xFFFF8809	IA_EXHEAACE_CONFIG_FATAL_WRITE_PCE
0xFFFF880A	IA_EXHEAACE_CONFIG_FATAL_USE_FULL_BANDWIDTH
0xFFFF880B	IA_EXHEAACE_CONFIG_FATAL_USE_SPEECH_CONF

Table 2-7 Configuration fatal error codes

2.5.4 Initialization fatal error codes

These are possible fatal error codes generated at the time of initialization of encoder. The encoder must be re-instantiated with appropriate correction in case of fatal errors.

Error Number	Error Code
0xFFFF9000	IA_EXHEAACE_INIT_FATAL_RESAMPLER_INIT_FAILED
0xFFFF9001	IA_EXHEAACE_INIT_FATAL_AAC_INIT_FAILED
0xFFFF9002	IA_EXHEAAACE_INIT_FATAL_AACPLUS_NOT_AVAIL
0xFFFF9003	IA_EXHEAACE_INIT_FATAL_BITRATE_NOT_SUPPORTED
0xFFFF9004	IA_EXHEAACE_INIT_FATAL_INVALID_TNS_PARAM
0xFFFF9005	IA_EXHEAACE_INIT_FATAL_SCALE_FACTOR_BAND_NOT_SUPPORTED
0xFFFF9006	IA_EXHEAACE_INIT_FATAL_INVALID_CORE_SAMPLE_RATE
0xFFFF9007	IA_EXHEAACE_INIT_FATAL_INVALID_BIT_RATE
0xFFFF9008	IA_EXHEAACE_INIT_FATAL_INVALID_ELEMENT_TYPE
0xFFFF9009	IA_EXHEAACE_INIT_FATAL_NUM_CHANNELS_NOT_SUPPORTED

0xFFFF900A	IA_EXHEAACE_INIT_FATAL_INVALID_NUM_CHANNELS_IN_ELE
0xFFFF900B	IA_EXHEAACE_INIT_FATAL_SFB_TABLE_INIT_FAILED
0xFFFF9100	IA_EXHEAACE_INIT_FATAL_MPS_INIT_FAILED
0xFFFF9400	IA_EXHEAACE_INIT_FATAL_SBR_INVALID_NUM_CHANNELS
0xFFFF9401	IA_EXHEAACE_INIT_FATAL_SBR_INVALID_SAMPLERATE_MODE
0xFFFF9402	IA_EXHEAACE_INIT_FATAL_SBR_INVALID_FREQ_COEFFS
0xFFFF9403	IA_EXHEAACE_INIT_FATAL_SBR_INVALID_NUM_BANDS
0xFFFF9404	IA_EXHEAACE_INIT_FATAL_SBR_INVALID_BUFFER_LENGTH
0xFFFF9405	IA_EXEHAACE_INIT_FATAL_SBR_NOISE_BAND_NOT_SUPPORTED

Table 2-8 Initialization fatal error codes

2.5.5 Execution non-fatal error codes

These are possible non-fatal error codes generated at the time of process call of encoder. The content of the output buffer shall not be valid when these error codes are returned. Next input data can be provided to encoder without any corrective actions.

Error Number	Error Code
0x0001900	IA_EXHEAACE_EXE_NONFATAL_MPS_ENCODE_ERROR
0x0001901	IA_EXHEAACE_EXE_NONFATAL_MPS_INVALID_DATA_BANDS
0x0001C00	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_BANDWIDTH_INDEX
0x0001C01	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_NUM_PATCH
0x0001C02	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_VOCOD_BUF
0x0001C03	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_PVC_MODE
0x0001C04	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_FFT
0x0001C05	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_START_BAND
0x0001C06	IA_EXHEAACE_EXE_NONFATAL_ESBR_INVALID_VALUE

Table 2-9 Execution non-fatal error codes

2.5.6 Execution fatal error codes

These are possible fatal error codes generated at the time of process call of encoder. The encoder needs to be re-initialised or re-instantiated once this error is reported.

Error Number	Error Code
0xFFFF9800	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_TIME_SLOTS
0xFFFF9801	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_IN_CHANNELS

0xFFFF9802	IA_EXHEAACE_EXE_FATAL_PS_INVALID_HYBRID_RES_VAL
0xFFFF9803	IA_EXHEAACE_EXE_FATAL_UNSUPPORTED_AOT
0xFFFF9804	IA_EXHEAACE_EXE_FATAL_INVALID_BLOCK_TYPE
0xFFFF9805	IA_EXHEAACE_EXE_FATAL_INVALID_SBR_FRAME_TYPE
0xFFFF9806	IA_EXHEAACE_EXE_FATAL_INVALID_SBR_NUM_ENVELOPES
0xFFFF9807	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_NUM_BANDS
0xFFFF9808	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_BS
0xFFFF9809	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_CODEBOOK
0xFFFF980A	IA_EXHEAACE_EXE_FATAL_INVALID_SCALE_FACTOR_GAIN
0xFFFF980B	IA_EXHEAACE_EXE_FATAL_INVALID_BIT_RES_LEVEL
0xFFFF980C	IA_EXHEAACE_EXE_FATAL_INVALID_BIT_CONSUMPTION
0xFFFF980D	IA_EXHEAACE_EXE_FATAL_INVALID_SIDE_INFO_BITS
0xFFFF980E	IA_EXHEAACE_EXE_FATAL_INVALID_HUFFMAN_BITS
0xFFFF980F	IA_EXHEAACE_EXE_FATAL_INVALID_SCALE_FACTOR_BITS
0xFFFF9810	IA_EXHAACE_EXE_FATAL_SBR_INVALID_AMP_RES
0xFFFF9811	IA_EXHEAACE_EXE_FATAL_INVALID_OUT_BYTES
0xFFFF9812	IA_EXHEAACE_EXE_FATAL_INVALID_TNS_FILT_ORDER
0xFFFF9813	IA_EXHEAACE_EXE_FATAL_SBR_INVALID_SAMP_FREQ
0xFFFF9900	IA_EXHEAACE_EXE_FATAL_MPS_NULL_DATA_HANDLE
0xFFFF9901	IA_EXHEAACE_EXE_FATAL_MPS_INVALID_HUFF_DATA_TYPE
0xFFFF9902	IA_EXHEAACE_EXE_FATAL_MPS_INVALID_NUM_PARAM_SETS
0xFFFF9903	IA_EXHEAACE_EXE_FATAL_MPS_UNSUPPORTED_GUIDED_ENV_SHAPE
0xFFFF9904	IA_EXHEAACE_EXE_FATAL_MPS_3D_STEREO_MODE_NOT_SUPPORTED
0xFFFF9905	IA_EXHEAACE_EXE_FATAL_MPS_UNSUPPORTED_RESIDUAL_CODING
0xFFFF9906	IA_EXHEAACE_EXE_FATAL_MPS_UNSUPPORTED_ARBITARY_DOWNMIX_CODING
0xFFFF9907	IA_EXHEAACE_EXE_FATAL_MPS_ARBITARY_TREE_NOT_SUPPORTED
0xFFFF9908	IA_EXHEAACE_EXE_FATAL_MPS_INVALID_QUANT_COARSE
0xFFFF9909	IA_EXHEAACE_EXE_FATAL_MPS_INVALID_RES_STRIDE
0xFFFF990A	IA_EXHEAACE_EXE_FATAL_MPS_INVALID_LEVELS
0xFFFF990B	IA_EXHEAACE_EXE_FATAL_MPS_CFFT_PROCESS

Table 2-10 Execution fatal error codes

3. Input and Output configuration structure

This section describes the definitions of the elements of the input and output configuration structures used in the API call.

3.1 Input Configuration Structure

Data Type	Element Name	Description
`UWORD32`	`ui_pcm_wd_sz`	Word size of PCM input.
`WORD32`	`i_bitrate`	Bitrate to be used for encoding.
`WORD32`	`frame_length`	Frame length to be used for encoding.
`WORD32`	`frame_cmd_flag`	Frame length command flag.
`WORD32`	`out_bytes_flag`	Flag to signal the library to use default or user-set bit reservoir size.
`WORD32`	`user_tns_flag`	Flag to indicate to tns is enabled.
`WORD32`	`aot`	Audio Object Type specifier
`WORD32`	`i_mps_tree_config`	MPS tree configuration
`WORD32`	`esbr_flag`	Flag to enable eSBR for HE-AACv1 streams
`WORD32`	`i_channels`	Number of channels of PCM input.
`WORD32`	`i_samp_freq`	Sampling frequency of PCM input.

WORD32	i_native_samp_freq	Native sampling frequency.
WORD32	i_channels_mask	Channel mask of PCM input data.
WORD32	i_num_coupling_chan	Number of coupling channels.
WORD32	i_use_mps	Enable/Disable MPS encoding when AOT is AAC-ELD (AAC-ELDv2 profile).
WORD32	i_use_adts	Flag that indicates to use ADTS header. Applicable only for HE-AACv2 and its subset profiles.
WORD32	i_use_es	Flag that indicates to encode as elementary stream. Suitable for feeding as input to MP4
FLAG	write_program_config_element	Flag to indicate PCE writing.
ixheaace_aac_enc_config	aac_config	AAC parameter configuration structure

Table 3-1 ixheaace_input_config structure description

Data Type	Element Name	Description
WORD32	sample_rate	Input stream sampling frequency
WORD32	bitrate	Encoder bit rate in bits/sec
WORD32	num_channels_in	Number of input channels
WORD32	num_channels_out	Number of output channels
WORD32	bandwidth	Targeted audio bandwidth in Hz
WORD32	dual_mono	Flag to make 2 SCEs for stereo input files
WORD32	use_tns	Enable/disable TNS
WORD32	noise_filling	Enable/disable noise filling
WORD32	use_adts	Use ADTS header
WORD32	private_bit	Private bit of MPEG Header
WORD32	copyright_bit	Copyright bit of MPEG Header
WORD32	original_copy_bit	Original bit of MPEG Header

WORD32	f_no_stereo_preprocessing	Forbid usage of stereo pre-processing
WORD32	inv_quant	Improve distortion by inverse quantization
WORD32	full_bandwidth	Enable usage of full bandwidth of input
WORD32	bitreservoir_size	Size of bit reservoir
WORD32	length	AAC configuration block length

Table 3-2 ixheaace_aac_enc_config structure description

3.2 Output Configuration Structure

Data Type	Element Name	Description
WORD32	i_out_bytes	Number of encoded output bytes.
WORD32	i_bytes_consumed	Number of bytes used by the encoder in the input buffer.
UWORD32	ui_inp_buf_size	Input buffer size.
UWORD32	malloc_count	Counter holding the value of total memory allocations done.
UWORD32	ui_rem	Memory alignment related parameter.
UWORD32	ui_proc_mem_tabs_size	Codec memory tables size field.
pVOID	pv_ia_process_api_obj	Pointer to encoder API object.
pVOID	arr_alloc_memory[100]	Array containing all the addresses of the dynamically allocated memories requested by the encoder library.
pVOID	malloc_xheaace	Pointer to system memory allocation function.
VOID	free_xheaace	Pointer to dynamically allocated memory freeing function.
ixheaace_version	version	Structure containing information about library name and library version number
ixheaace_mem_info_table	mem_info_table[4]	Structure containing information about the dynamically allocated memories used by the encoder library.
WORD32	input_size	Size of the input file (in samples)

WORD32	samp_freq	AAC core coder sampling frequency
WORD32	header_samp_freq	Sampling frequency to be specified in the header
WORD32	audio_profile	Audio profile
FLOAT32	down_sampling_ratio	Downsampling ratio
pWORD32	pb_inp_buf_32	Pointer to input buffer

Table 3-3 ixheaace_output_config structure description

Data Type	Element Name	Description
UWORD32	ui_size	Size of memory
UWORD32	ui_alignment	Alignment of memory
UWORD32	ui_type	Type of memory
pVOID	mem_ptr	Allocated memory address

Table 3-4 ixheaace_mem_info_table structure description

Data Type	Element Name	Description
WORD8 *	p_lib_name	Pointer to library name string
WORD8 *	p_version_num	Pointer to library version number

Table 3-5 ixheaace_version structure description

4. Reference

[1] *ISO/IEC 14496-3:2001/Amd1, Bandwidth Extension* (MPEG-4)

[2] *ISO/IEC 14496-3:2001/Amd2, Parametric Audio for High Quality Audio* (MPEG-4)