ESWIN

EIC7700 Audio Development Manual

For EIC7700 and EIC7702

Engineering Draft / Rev0.7

Aug.6, 2024

Note

Change History

Version No	Date	Descriptions
Rev 0.1	Jan.22,2024	Initial version
Rev 0.3	Mar.08,2024	Update audio input, audio output.Add audio encoding and audio decoding content.
Rev 0.6	May.14,2024	Update audio input, audio output, audio encoding, and audio decoding parts.
Rev 0.7	Aug.06,2024	Update audio input, audio output, audio encoding, and audio decoding parts.

Table of contents

1. Overview

The audio module includes four sub-modules: audio input, audio output, audio encoding, and audio decoding. The audio input and output module implements audio input and audio output functions by controlling the audio interface of the EIC7700 chip.The audio decoding module support decoding G711a, G711μ, G722, G726, AAC-LC, AAC-HEv1, AAC-HEv2, MP2L2, MP3, and AMR streams. The audio encoding module support encoding PCM data to G711a, G711μ, G722, G726, AAC-LC stream. Additionally, the audio module supports recording and playing raw audio files in LPCM format.

2. API Reference

Audio API is used for audio playback, audio recording, audio data processing, and audio encoding and decoding functionality. Audio API enables developers to integrate audio modules into their applications.

2.1 Audio Input

The Audio Input (AI) module is used to configure and enable AI devices and obtain audio frames. This module provides the following APIs:

- ES_AI_SetPubAttr: Sets attributes of an AI device.
- ES_AI_GetPubAttr: Gets attributes of an AI device.
- ES_AI_Enable: Enables an AI device.
- ES_AI_Disable: Disables an AI device.
- ES_AI_EnableChn: Enables an AI channel.
- ES_AI_DisableChn: Disables an AI channel.
- ES_AI_GetFrame: Gets audio frames.
- ES_AI_ReleaseFrame: Releases the buffer for storing audio frames.
- ES_AI_SetChnParam: Sets parameters of an AI device.
- ES_AI_GetChnParam: Gets parameters of an AI device.
- ES_AI_EnableReSmp: Enables AI resampling.
- ES_AI_DisableReSmp: Disables AI resampling.
- ES_AI_SetTrackMode: Sets the track mode of an AI device.
- ES_AI_GetTrackMode: Gets the track mode of an AI device.
- ES_AI_GetFd: Gets the device file description corresponding to an AI device.
- ES_AI_ClrPubAttr: Clears the attributes of an AI device.
- ES_AI_SaveFile: Enables the audio input file saving functionality.
- ES_AI_QueryFileStatus: Queries whether the audio input device is in the file saving status.
- ES_AI_SetVolume: Sets the AI device volume.
- ES_AI_GetVolume: Gets the AI device volume.
- ES_AI_EnableAecRefFrame: Enables user to obtain the AEC reference frame even when AEC is not enabled.
- ES_AI_DisableAecRefFrame: Disables user obtaining the AEC reference frame when AEC is not enabled.
- ES_AI_SetVQEAttr: Sets the attributes of AI voice quality enhancement.
- ES_AI_GetVQEAttr: Gets the attributes of AI voice quality enhancement.
- ES_AI_EnableVQE: Enables AI voice quality enhancement.
- ES_AI_DisableVQE: Disables AI voice quality enhancement.

2.1.1 ES_AI_SetPubAttr

【Function body】

ES_S32 ES_AI_SetPubAttr (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId,
const AIO_ATTR_S* Attr)

【Description】

Sets attributes of of an AI device.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
Attr	AI device attribute pointer.	Input

【Return】

Return Value	Description
0	Success.
Non 0	Fail, refers to the error code .

【Note】

The attribute of the audio input device determines the format of the input data, including sampling rate, software sampling precision, hardware sampling precision, buffer size, number of sampling points per frame, and channel mode.

- Sampling Rate
  Refers to the hardware sampling rate of AI recording, indicating the number of sampling points collected within 1 second. The hardware sampling rate of AI can be set to 8kHz, 16kHz, 32kHz, or 48kHz.
- Software Sampling Precision
  The software sampling precision set by the user at the application layer refers to the number of bits occupied by each sampling point recorded. The sampling precison of the software can be set to 16bit or 32bit.
- Hardware Sampling Precision
  The sampling precision supported by the hardware refers to the number of bits occupied by each sampling point of the hardware. The hardware sampling precision can be set to 16bit or 32bit.
- Buffer Size
  The buffer size refers to the number of audio frames, and the audio of EIC7700 is buffered in the form of frames. The number of frames can be set to [2, 30]. It is recommended to set 4 or greater, otherwise there is a risk of frame loss.
- Samples
  Samples refers to the number of samples contained in each frame of data. It is generally recommended to set 512 or 1024. When AI is bound to AENC, it is generally set to the pcm data size of one frame of the encoding module.
- channel mode
  AI module supports stereo and mono.

2.1.2 ES_AI_GetPubAttr

【Function body】

ES_S32 ES_AI_GetPubAttr (

AUDIO_CARDAiCardId,
AUDIO_DEV AiDevId,
AIO_ATTR_S* Attr)

【Description】

Gets the attributes of Audio Input device.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
Attr	AI device attribute pointer.	Output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

The attributes of AI needs to be set first before the attributes can be obtained, otherwise the call will fail.

2.1.3 ES_AI_Enable

【Function body】

ES_S32 ES_AI_Enable (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId)

【Description】

Enables AI devices.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description

0	success.
Non 0	Fail, refers to the error code .

【Note】

- The attributes of AI device must be configured before enabled, otherwise an attributes not configured error will be returned.
- If the AI device is already enabled, success will be returned directly.

2.1.4 ES_AI_Disable

【Function body】

ES_S32 ES_AI_Disable (

AUDIO_CARD AiCardId,

AUDIO_DEV AiDevId)

【Description】

Disables AI devices.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- If the AI device is already disabled, success will be returned directly.
- When disable the AI device, if the AI channel under the device is enabled, the channel will be automatically disabled first.
- It is required that before disabling the AI device, first disable the AENC channel and AO device associated with it and using the audio data of the AI. Otherwise, the interface call may fail.

2.1.5 ES_AI_EnableChn

【Function body】

ES_S32 ES_AI_EnableChn (

AUDIO_CARD AiCardId,

AUDIO_DEV AiDevId)

【Description】

Enables AI channel.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Once enabling the AI channel, the corresponding AI device must be enabled first, otherwise an error code indicating the device has not been started will be returned.

2.1.6 ES_AI_DisableChn

【Function body】

```
ES_S32 ES_AI_Disable (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId)
```

【Description】

Disables AI channel.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.7 ES_AI_GetFrame

【Function body】

```
ES_S32 ES_AI_GetFrame (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AUDIO_FRAME_S* Frame,
        AUDIO_AEC_FRAME_S* AecFrame,
        ES_S32 MilliSec)
```

【Description】

Get audio frame.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
Frame	Audio frame structure pointer.	Output
AecFrame	Echo cancellation reference frame structure pointer.	output
MilliSec	Timeout for getting data -1 indicates blocking mode, waiting until there is no data; 0 indicates non-blocking mode, and an error will be returned when there is no data; >0 means blocking s32MilliSec for milliseconds, and an error will be returned when timeout occurs.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- When AI's echo cancellation reference frame acquisition is enabled, AecFrame cannot be a null pointer; When AI's echo cancellation reference frame acquisition is not enabled, AecFrame can be set to null.
- The AI module will cache audio frame data for user acquisition.
- The value of MilliSec must be greater than or equal to -1. When it sets to -1, the blocking mode is used to obtain data. When it sets to 0, the non- blocking mode is used to obtain data. When it is greater than 0, ES_AI_GetFrame will be blocked after MilliSec milliseconds, if there is no data, a timeout will be returned and an error will be reported.
- Before obtaining audio frame data, the corresponding AI channel must be enabled first.

2.1.8 ES_AI_ReleaseFrame

【Function body】

ES_S32 ES_AI_ReleaseFrame (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId,
const AUDIO_FRAME_S* Frame,
const AUDIO_AEC_FRAME_S* AecFrame)

【Description】

Release the audio frame.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
Frame	Audio frame structure pointer.	Input
AecFrame	Echo cancellation reference frame structure pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

If there is no need to release the echo cancellation reference frame, just set AecFrm to NULL.

2.1.9 ES_AI_SetChnParam

【Function body】

ES_S32 ES_AI_SetChnParam (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId,
const AI_CHN_PARAM_S* ChnParam)

【Description】

Set AI channel parameters.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
ChnParam	Audio channel parameters.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The channel Parameters currently has only one variable, which is used to set the buffer depth for users to obtain audio frames. The default depth is the buffer size set by the attribute. The value of this member variable cannot be greater than 30.
- It is recommended to call ES_AI_GetChnParam interface to obtain the default configuration firstly, and then call ES_AI_SetChnParam to modify the configuration to facilitate subsequent expansion.

2.1.10 ES_AI_GetChnParam

【Function body】

```
ES_S32 ES_AI_GetChnParam (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AI_CHN_PARAM_S* ChnParam)
```

【Description】

Get AI channel parameters.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
ChnParam	Audio channel parameters.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.11 ES_AI_EnableReSmp

【Function body】

```
ES_S32 ES_AI_EnableReSmp (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AUDIO_SAMPLE_RATE_E OutSampleRate)
```

【Description】

Enable AI resampling.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
OutSampleRate	The sample rate of the audio output after resampling.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code.

【Note】

- After enabling the AI channel, call this interface to enable the resampling function.
- The resampling function is allowed to be enabled repeatedly, but the properties configured must be same.
- After disabling the AI channel, once enabling the AI channel and using the resampling function, developers need to call this interface to re-enable resampling.

2.1.12 ES_AI_DisableReSmp

【Function body】

```
ES_S32 ES_AI_DisableReSmp (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId)
```

【Description】

Enable AI resampling.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code.

【Note】

- If the AI resampling function is no longer used, this interface should be called to

disable it.

- Developers need call this interface before setting AO device in bonding mode(AI binding AO) or AENC channel using AI binding AENC, otherwise memory errors may occur.

2.1.13 ES_AI_SetTrackMode

【Function body】

```
ES_S32 ES_AI_SetTrackMode (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AUDIO_TRACK_MODE_E TrackMode)
```

【Description】

Set AI channel mode.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
TrackMode	Audio input channel mode.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AI device has been successfully enabled.

2.1.14 ES_AI_GetTrackMode

【Function body】

```
ES_S32 ES_AI_GetTrackMode (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AUDIO_TRACK_MODE_E* TrackMode)
```

【Description】

Get AI channel mode means getting dual channel or mono channel.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer	Input

	to Table 3-1	
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
TrackMode	Audio input channel mode pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AI device has been successfully enabled.

2.1.15 ES_AI_GetFd

【Function body】

```
ES_S32 ES_AI_GetFd (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId)
```

【Description】

Get the device file handle corresponding to the audio input channel number.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.16 ES_AI_ClrPubAttr

【Function body】

```
ES_S32 ES_AI_ClrPubAttr (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId)
```

【Description】

Clear the Pub attribute.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Before clearing device properties, developers need to stop the device.

2.1.17 ES_AI_SaveFile

【Function body】

```
ES_S32 ES_AI_SaveFile (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        const AUDIO_SAVE_FILE_INFO_S* SaveFileInfo)
```

【Description】

Enable the audio input save file function.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
SaveFileInfo	Audio save file attribute structure pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- This interface is used to dump the data before and after VQE processing in AI (when VQE is not enabled, it dumps the data before and after volume and resampling processing). When the echo reference frame is enabled, it can also simultaneously dump the echo-cancelled reference frame before VQE processing.
- After calling this interface, three files of specified size will be written out in the specified directory.

Sin_AiCardx_Devx_xk_fileName.pcm is the input frame before VQE processing,
Rin_AiCardx_Devx_xk_fileName.pcm is the echo cancellation reference frame before VQE processing,
Sou_AiCardx_Devx_xk_fileName.pcm is the output frame after VQE processing. For example, the real names of the three files saved by Card 0 device 0 under the 8k sampling rate are:
Sin_AiCard0_Dev0_8k_fileName.pcm, Rin_AiCard0_Dev0_8k_fileName.pcm,
Sou_AiCard0_Dev0_8k_fileName.pcm.

2.1.18 ES_AI_QueryFileStatus

【Function body】

```
ES_S32 ES_AI_QueryFileStatus (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AUDIO_SAVE_FILE_INFO_S* SaveFileInfo)
```

【Description】

Check whether the audio input channel is in the state of saving files.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
FileStatus	Pointer to the status attribute structure.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

This interface is used to query whether the audio input channel is in the state of saving files. After user calls ES_AI_SaveFile to store file, this interface can be called to query whether the stored file has reached the specified size. If Saving of FileStatus is ES_TRUE, it means the specified size has not been reached, otherwise the specified size has been reached.

2.1.19 ES_AI_SetVolume

【Function body】

```
ES_S32 ES_AI_SetVolume (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        ES_S32 VolumeDb)
```

【Description】

Set the AI device volume.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
VolumeDb	Audio device volume. Value range: [-90, 10]/dB	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.20 ES_AI_GetVolume

【Function body】

```
ES_S32 ES_AI_GetVolume (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        ES_S32* VolumeDb)
```

【Description】

Get the AI device volume size.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
VolumeDb	Audio device volume pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.21 ES_AI_EnableAecRefFrame

【Function body】

```
ES_S32 ES_AI_EnableAecRefFrame (
```

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId,
AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Get the AEC reference frame. When obtaining the AEC reference frame, developers need to turn off the built-in AEC function of essdk.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
AoCardId	sound card number used to obtain the AEC reference frame. Value range: refer to Table 3-1	Input
AoDevId	used to obtain the AEC reference frame. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The interdace enable obtaining the AEC reference frame when the AEC automatic processing function is turned off. The AEC reference frame is returned through the interface ES_AI_GetFrame for the user to perform AEC processing on the upper layer.
- If this interface is called when the AEC function is turned on, the error code ES_ERR_AI_NOT_SUPPORT is returned.
- Call this interface after the AI channel has been successfully enabled.

2.1.22 ES_AI_DisableAecRefFrame

【Function body】

ES_S32 ES_AI_DisableAecRefFrame (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId)

【Description】

It is forbidden to obtain the AEC reference frame when AEC is not turned on.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input

AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Repeated calls to this interface return success.

2.1.23 ES_AI_SetVQEAttr

【Function body】

```
ES_S32 ES_AI_SetVQEAttr (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        const AI_VQE_CONFIG_S* VQEConfig)
```

【Description】

Set properties related to AI's sound quality enhancement function.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
VQEConfig	Audio input sound quality enhancement configuration structure pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.24 ES_AI_GetVQEAttr

【Function body】

```
ES_S32 ES_AI_GetVQEAttr (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId,
        AI_VQE_CONFIG_S* VQEConfig)
```

【Description】

Get the properties related to AI's sound quality enhancement function.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input
VQEConfig	Audio input sound quality enhancement configuration structure pointer .	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.1.25 ES_AI_EnableVQE

【Function body】

```
ES_S32 ES_AI_EnableVQE (
        AUDIO_CARD AiCardId,
        AUDIO_DEV AiDevId)
```

【Description】

Enable AI's sound quality enhancement function.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Corresponding AI channel must be enabled before when call ES_AI_EnableVQE.
- Enabling the sound quality enhancement function of the same AI channel multiple times returns success.
  When the AI channel disabled, developers can call this interface to re-enable the sound quality enhancement function before AI channel enabled.

2.1.26 ES_AI_DisableVQE

【Function body】

ES_S32 ES_AI_DisableVQE (

AUDIO_CARD AiCardId,
AUDIO_DEV AiDevId)

【Description】

Disables AI's sound quality enhancement features.

【Parameters】

Parameter name	Description	Input/Output
AiCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AiDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- When the AI sound quality enhancement function is no longer used, this interface should be called to disable it.
- Disabling the sound quality enhancement function of the same AI channel multiple times returns success.

2.2 Audio Output

The Audio Input (AO) module is used to enable audio output devices and sending audio frames to the output devices.

This module provides the following APIs:

- ES_AO_SetPubAttr: Sets attributes of AO device.
- ES_AO_ GetPubAttr: Gets attributes of AO device.
- ES_AO_ Enable: Enables AO device.
- ES_AO_Disable: Disables AO device.
- ES_AO_EnableChn: Enables AO channel.
- ES_AO_DisableChn: Disables AO channel.
- ES_AO_SendFrame: Sends AO audio frame.
- ES_AO_EnableReSmp: Enables AO resampling.
- ES_AO_DisableReSmp: Disables AO resampling.
- ES_AO_PauseChn: Pauses the AO device.
- ES_AO_ResumeChn: Resumes AO device.
- ES_AO_ClearChnBuf: Clears the current audio data buffer in the AO device.
- ES_AO_QueryChnStat: Queries the current audio data cache status in the AO device.
- ES_AO_SetTrackMode: Sets track mode of the AO device.
- ES_AO_GetTrackMode: Get track mode of the AO device.
- ES_AO_SetVolume: Sets the volume of the AO device.
- ES_AO_GetVolume: Gets the volume of the AO device.
- ES_AO_SetMute: Sets the mute status of the AO device.

- ES_AO_GetMute: Gets the mute status of the AO device.
- ES_AO_GetFd: Gets the device file description corresponding to the AO device.
- ES_AO_SaveFile: Enables the audio output save file function.
- ES_AO_QueryFileStatus: Queries whether the audio output channel is in the status of saving files.
- ES_AO_ClrPubAttr: Clears attributes of AO device.
- ES_AO_SetVQEAttr: Sets the attributes related to the voice quality enhancement function of AO.
- ES_AO_GetVQEAttr: Gets the attributes related to the voice quality enhancement function of AO.
- ES_AO_EnableVQE: Enables AO voice quality enhancement.
- ES_AO_DisableVQE: Disables AO voice quality enhancement.

2.2.1 ES_AO_SetPubAttr

【Function body】

```
ES_S32 ES_AO_SetPubAttr (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        const AIO_ATTR_S* Attr)
```

【Description】

Set attributes of AO device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
Attr	A O device attribute pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

The attribute of the audio output device determines the format of the output data, including sampling rate, software sampling precision, hardware sampling precision, buffer size, number of sampling points per frame, and channel mode.

- Sampling Rate:
  The sampling rate refers to the hardware sampling rate of AO playback, which represents the number of sampling points collected within 1 second. The hardware sampling rate of AO output can be set to 8kHz, 16kHz, 32kHz, 48kHz, 64kHz, 96kHz. If the input sample rate is not within the supported range, you can call ES_AO_EnableReSmp make the conversion. Resampling and VQE prcessing cannot be used when samplerate sets to 64kHz and 96kHz.
- Software Sampling Precision
  Software sampling precision refers to the number of bits occupied by each sampling point recorded by the user. The sampling precision of the software can be

set to 16bit or 32bit.

- Hardware Sampling Precision
  Hardware sampling precision refers to the number of bits occupied by each sampling point of the hardware. The hardware sampling precision can be set to 16bit or 32bit.
- Buffer Size:
  The buffer size refers to the number of audio frames, and the audio of EIC7700 is buffered in the form of frames. The number of frames can be set to [2, 30], and it is recommended to set 4 or greater, otherwise there is a risk of frame loss.
- Samples
  The samples refer to the number of samples contained in each frame of data. It is generally recommended to set it to 512 or 1024. When AI is bound to AO, it is generally set to the pcm data size of one frame of the AI module.
- channel mode
  AO module supports stereo and mono.
- Number of channels
  Represents the number of independent audio channels transmitted simultaneously in a sound signal.

2.2.2 ES_AO_GetPubAttr

【Function body】

```
ES_S32 ES_AO_GetPubAttr (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        AIO_ATTR_S* Attr)
```

【Description】

Gets attributes of the Audio Output.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
Attr	AO device attributes pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

The audio output device determines the format of the input data, including sampling rate, data sampling precision, hardware bit width, buffer size, number of sampling points per frame, and number of channels.

2.2.3 ES_AO_Enable

【Function body】

ES_S32 ES_AO_Enable (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Enables AO device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The attributes of AO device must be configured before enabled, otherwise an attributes not configured error will be returned.
- If the AO device is already enabled, success will be returned directly.

2.2.4 ES_AO_Disable

【Function body】

ES_S32 ES_AO_Disable (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Disables AO device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return Value】

Return Value	Description

0	success.
Non 0	Fail, refers to the error code .

【Note】

- If the AO device is already disabled, success will be returned directly.
- When disable the AO device, if the AO channel under the device is opened, the channel will be automatically disabled first.

2.2.5 ES_AO_EnableChn

【Function body】

ES_S32 ES_AO_EnableChn (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Enables AO channel.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Once enabling the AO channel, the corresponding AO device must be enabled first, otherwise an error code indicating the device has not been started will be returned.

2.2.6 ES_AO_DisableChn

【Function body】

ES_S32 ES_AO_ DisableChn (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Disables AO channel.

【Parameters】

Parameter name	Description	Input/Output

AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return Value】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.2.7 ES_AO_SendFrame

【Function body】

```
ES_S32 ES_AO_SendFrame (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        const AUDIO_FRAME_S* Data,
        ES_S32 MilliSec)
```

【Description】

Sends audio frames.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
Data	Audio frame structure pointer.	Input
MilliSec	Timeout for sending data -1 indicates blocking mode, waiting until the buffer is full; 0 indicates non-blocking mode, and an error will be returned when the buffer is full; >0 means blocking s32MilliSec for milliseconds, and an error will be returned when timeout occurs.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The value of MilliSec must be greater than or equal to -1. When it sets to -1, the blocking mode is used to send data. When it sets to 0, the data is sent in non-blocking mode. When it is greater than 0, ES_AO_SendFrame will be blocked after MilliSec milliseconds, if there is data, the cache that does not send data will return a timeout

and report an error.
- Calling this interface to send audio frames to AO output, the corresponding AO channel must be enabled first.

2.2.8 ES_AO_EnableReSmp

【Function body】

```
ES_S32 ES_AO_EnableReSmp (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        AUDIO_SAMPLE_RATE_E InSampleRate)
```

【Description】

Enables AO resampling.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
InSampleRate	The input sample rate for audio resampling.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- After enabling the AO channel, call this interface to enable the resampling function.
- The resampling function is allowed to be enabled repeatedly, but the properties configured must be same.
- After disabling the AO channel, once enabling the AO channel and using the resampling function, developers need to call this interface to re-enable resampling.
- The same sampling rate cannot be enabled repeatedly.
- ESSDK will internally convert the software sampling rate data into a sampling rate supported by the hardware.

2.2.9 ES_AO_DisableReSmp

【Function body】

```
ES_S32 ES_AO_DisableReSmp (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId)
```

【Description】

Disables AO resampling.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

If the AO resampling function is no longer used, you should call this interface to disable it.

2.2.10 ES_AO_PauseChn

【Function body】

```
ES_S32 ES_AO_PauseChn (
                AUDIO_CARD AoCardId,
                AUDIO_DEV AoDevId)
```

【Description】

Pauses the AO channel.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

When the AO channel is disabled, calling this interface to pause the AO channel is not allowed.

2.2.11 ES_AO_ResumeChn

【Function body】

```
ES_S32 ES_AO_ResumeChn (
```

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Resumes AO channel.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- After the AO channel is suspended, it can be resumed by calling this interface.
- The AO channel is enabled and the playback is paused, calling this interface will return success; otherwise, the call will return an error.

2.2.12 ES_AO_ClearChnBuf

【Function body】

ES_S32 ES_AO_ClearChnBuf (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Clears the current audio data cache in the AO channel.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.13 ES_AO_QueryChnStat

【Function body】

```
ES_S32 ES_AO_QueryChnStat (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        AO_CHN_STATE_S* Status)
```

【Description】

Queries the current audio data cache status in the AO channel.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
Status	Cache status structure pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.14 ES_AO_SetTrackMode

【Function body】

```
ES_S32 ES_AO_ SetTrackMode (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        AUDIO_TRACK_MODE_E TrackMode)
```

【Description】

Sets AO track mode.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

TrackMode	Audio output channel mode.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Call this interface after the AO device has been successfully enabled.
- This interface takes effect when the audio is actually played.

2.2.15 ES_AO_GetTrackMode

【Function body】

ES_S32 ES_AO_ SetTrackMode (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId,
AUDIO_TRACK_MODE_E* TrackMode)

【Description】

Gets the AO track mode.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
TrackMode	Audio output channel mode pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.16 ES_AO_SetVolume

【Function body】

ES_S32 ES_AO_SetVolume (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId,
ES_S32 VolumeDb)

【Description】

Sets the AO device volume.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
VolumeDb	Audio device volume. Value range: [-90, 10]/dB	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.17 ES_AO_GetVolume

【Function body】

```
ES_S32 ES_AO_GetVolume (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        ES_S32* VolumeDb)
```

【Description】

Gets the AO device volume.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
VolumeDb	Audio device volume pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.18 ES_AO_SetMute

【Function body】

```
ES_S32 ES_AO_SetMute (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        ES_BOOL Enable,
        const AUDIO_FADE_S* Fade)
```

【Description】

Sets the mute status of the AO device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
Ao DevId	Audio device number. Value range: refer to Table 3-1	Input
Enable	Whether the audio device is muted. ES_TRUE: Enable mute function; ES_FALSE: Turn off the mute function.	Input
Fade	Fade structure pointer. Fade can be enabled /disabled in the structure pointed to by this pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO channel is successfully enabled.

2.2.19 ES_AO_GetMute

【Function body】

```
ES_S32 ES_AO_GetMute (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        ES_BOOL* Enable,
        const AUDIO_FADE_S* Fade)
```

【Description】

Gets the mute status of the AO device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
Enable	Audio device mute status pointer.	output
Fade	Fade structure pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO device is successfully enabled.

2.2.20 ES_AO_GetFd

【Function body】

```
ES_S32 ES_AO_GetFd (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId)
```

【Description】

Gets the audio device file handle corresponding to the audio output sound card and device.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Call this interface after the AO device is successfully enabled.

2.2.21 ES_AO_SaveFile

【Function body】

```
ES_S32 ES_AO_SaveFile (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        const AUDIO_SAVE_FILE_INFO_S* SaveFileInfo)
```

【Description】

Enables audio output to save files.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
Ao DevId	Audio device number. Value range: refer to Table 3-1	Input
SaveFileInfo	Audio save file attribute structure pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- This interface is used to dump files before and after VQE processing in AO (when VQE is not enabled, it dumps data before and after volume and resampling processing).
- After calling this interface, two files of the specified size will be written out in the specified directory. Sin_AoCardx_Devx_xk_fileName.pcm is the input frame before VQE processing. Sou_AoCardx_Devx_xk_fileName.pcm is the output frame after VQE processing. For example: Card 0 device 0 saved with file names at 8k sampling rate are: Sin_AoCard0_Dev0_8k_fileName.pcm, Sou_AoCard0_Dev0_8k_fileName.pcm.

2.2.22 ES_AO_QueryFileStatus

【Function body】

```
ES_S32 ES_AO_QueryFileStatus (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        AUDIO_FILE_STATUS_S* FileStatus)
```

【Description】

Queries whether the audio input channel is in the state of saving files.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

FileStatus	Pointer to the status attribute structure.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

This interface is used to query whether the audio input channel is in the state of saving files. After the user calls ES_AO_SaveFile to store the file, this interface can be called to query whether the stored file has reached the specified size. If Saving of FileStatus is ES_TRUE, it means the specified size has not been reached, otherwise the specified size has been reached.

2.2.23 ES_AO_ClrPubAttr

【Function body】

```
ES_S32 ES_AO_ClrPubAttr (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId)
```

【Description】

Clears the Pub attribute.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

Before clearing device properties, developors need to stop the device.

2.2.24 ES_AO_SetVQEAttr

【Function body】

```
ES_S32 ES_AO_SetVQEAttr (
        AUDIO_CARD AoCardId,
        AUDIO_DEV AoDevId,
        const AO_VQE_CONFIG_S* VQEConfig)
```

【Description】

Sets the attributes of AO voice quality enhancement.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
VQEConfig	Audio output sound quality enhancement configuration structure pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.2.25 ES_AO_GetVQEAttr

【Function body】

```
ES_S32 ES_AO_GetVQEAttr (
                AUDIO_CARD AoCardId,
                AUDIO_DEV AoDevId,
                AO_VQE_CONFIG_S* VQEConfig)
```

【Description】

Gets the attributes of AO voice quality enhancement.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input
VQEConfig	Audio input sound quality enhancement configuration structure pointer .	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.2.26 ES_AO_EnableVQE

【Function body】

ES_S32 ES_AO_EnableVQE (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Enables AO voice quality enhancement.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Corresponding AO channel must be enabled before enabling the voice quality enhancement function.
- Enabling the sound quality enhancement function of the same AO channel multiple times returns success.
- When the AO channel disabled, developers can call this interface to re-enable the sound quality enhancement function before AO channel enabled.

2.2.27 ES_AO_DisableVQE

【Function body】

ES_S32 ES_AO_DisableVQE (

AUDIO_CARD AoCardId,
AUDIO_DEV AoDevId)

【Description】

Disables AO voice quality enhancement.

【Parameters】

Parameter name	Description	Input/Output
AoCardId	Audio sound card number. Value range: refer to Table 3-1	Input
AoDevId	Audio device number. Value range: refer to Table 3-1	Input

【Return】

Return Value	Description
0	success.

Non 0	Fail, refers to the error code .

【Note】

- When the AO sound quality enhancement function is no longer used, this interface should be called to disable it.
- Disabling the sound quality enhancement function of the same AO channel multiple times returns success.

2.3 Audio Encoding

2.3.1 ES_AENC_CreateChn

【Function body】

```
ES_S32 ES_AENC_CreateChn (
        AENC_CHN AeChn,
        const AENC_CHN_ATTR_S* Attr)
```

【Description】

Creates an audio encoding channel.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Attr	Audio encoding channel attributes pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Audio encoding supports G711a, G711μ, G722, G726, and AAC-LC.
- The encoding and decoding only supports 16-bit linear PCM audio data processing. It is recommended to set the expansion flag to 1 when configuring the public attributes of the AI device so that the AI data automatically expands 8bit to 16bit.
- Properties of audio encoding need to match the properties of the input audio data, such as sampling rate, frame length (number of sample points per frame), etc.
- The buffer size is in frames, and the value range is [2, MAX_AUDIO_FRAME_NUM]. It is recommended to configure it to greater than 10. A buffer configuration that is too small may cause frame loss and other abnormalities.
- When the encode channel is idle, creating a channel will cause an error indicating that the channel has been created.

2.3.2 ES_AENC_DestroyChn

【Function body】

ES_S32 ES_AENC_DestroyChn (AENC_CHN AeChn)

【Description】

Destroys the audio encoding channel.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number: [0, AENC_MAX_CHN_NUM].	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

If the channel is destroyed while acquiring/releasing the code stream or sending a frame, a failure will be returned. Users need to pay attention when synchronizing the process.

2.3.3 ES_AENC_SendFrame

【Function body】

ES_S32 ES_AENC_SendFrame (

AENC_CHN AeChn,
const AUDIO_FRAME_S* Frame,
const AEC_FRAME_S * AecFrm)

【Description】

Sends the encoded audio frames.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Frame	Audio frame structure pointer	Input
AecFrm	Echo cancellation reference frame structure pointer	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- If echo cancellation is not required, pstAecFrm can be set to NULL.
- The audio encoding stream is a non-blocking interface. If the audio code stream

cache is full, failure will be returned directly.

- This interface is used by users sending audio frames for encoding. If the AENC channel has been bound to AI through the system binding (ES_SYS_Bind) interface, it is not necessary or recommended to adjust this interface.
- When calling this interface to send audio encoding audio frames, the corresponding encoding channel must be created first.

2.3.4 ES_AENC_GetStream

【Function body】

```
ES_S32 ES_AENC_GetStream (
        AENC_CHN AeChn,
        AUDIO_STREAM_S* Stream,
        ES_S32 MilliSec)
```

【Description】

Gets the encoded stream data.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Stream	Obtained audio code stream.	output
MilliSec	Timeout for getting data: -1 indicates blocking mode, waiting until there is no data; 0 indicates non-blocking mode, and an error will be returned when there is no data; >0 means blocking s32MilliSec for milliseconds, and an error will be returned when timeout occurs.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The channel must be created before the code stream can be obtained, otherwise failure will be returned directly. If the channel is destroyed during the process of obtaining the code stream, failure will be returned immediately.
- The value of s32MilliSec must be greater than or equal to -1. When it sets to -1, the blocking mode is used to obtain data. When it sets to 0, the non-blocking mode is used to obtain data. When it is greater than 0, ES_AENC_GetStream will be blocked after MilliSec milliseconds, if there is no data, a timeout will be returned and an error will be reported.
- The method of directly obtaining AI original audio data: set the encoding protocol type to PT_LPCM. After binding the AI channel, the audio data obtained from this AENC channel is the AI original data.

2.3.5 ES_AENC_ReleaseStream

【Function body】

```
ES_S32 ES_AENC_ReleaseStream (
                AENC_CHN AeChn,
                const AUDIO_STREAM_S* Stream)
```

【Description】

Release the code stream obtained from the audio encoding channel.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Stream	Obtained code stream pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- It is best to release the code stream immediately. If not released in time, the encoding process will be blocked.
- The code stream released must be the code stream obtained from the corresponding channel, and modifications to the code stream information structure are not allowed. Otherwise, the code stream will not be released, causing the code stream buffer to be lost or even causing a program exception.
- When releasing the code stream, deveploers must ensure that the channel has been created, otherwise failure will be returned directly. If the channel is destroyed during the release of the code stream, failure will be returned immediately.

2.3.6 ES_AENC_GetFd

【Function body】

```
ES_S32 ES_AENC_GetFd (AENC_CHN AeChn)
```

【Description】

Gets the device file handle corresponding to the audio encoding channel number.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.3.7 ES_AENC_RegisterEncoder

【Function body】

ES_S32 ES_AENC_RegisterEncoder (

ES_S32* Handle,
const AENC_ENCODER_S* Encoder)

【Description】

Register an external encoder.

Note: The built-in encoder is the encoder that we have implemented and registered by default. The external encoder refers to the encoder that the user calls this interface to register.

【Parameters】

Parameter name	Description	Input/Output
Handle	Register handle.	Input
Encoder	Encoder attribute structure	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The user registers an encoder with the AENC module by passing in the encoder attribute structure and returns the registration handle. The user can finally unregister the encoder through the registration handle.
- The AENC module can register up to 20 encoders, and it has registered five encoders, including G711a, G711μ, G722 , G726, and AAC-LC.
- The same encoding protocol does not allow repeated registration of encoders. For example, if a G726 encoder has been registered, registration of another G726 encoder is not allowed.

2.3.8 ES_AENC_UnRegisterEncoder

【Function body】

ES_S32 ES_AENC_UnRegisterEncoder (ES_S32 Handle)

【Description】

Unregister the encoder.

【Parameters】

Parameter name	Description	Input/Output
Handle	Registration handle (the handle obtained when registering the encoder).	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Normally there is no need to unregister the encoder.
- Before unregistering the encoder, developers need to destroy all encoding channels created by the encoder. When encoding channels are not destroyed or in destroye proceesing, calling this interface will return an error.

2.3.9 ES_AENC_SetMute

【Function body】

```
ES_S32 ES_AENC_Set Mute (
                AENC_CHN AeChn,
                ES_BOOL Enable)
```

【Description】

Sets the encoding channel mute status.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Audio encoding channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Enable	Whether audio mute is enabled. ES_TRUE: Enable mute function; ES_FALSE: Turn off mute function.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.3.10 ES_AENC_GetMute

【Function body】

```
ES_S32 ES_AENC_GetMute (
                AENC_CHN AeChn,
                ES_BOOL* Enable)
```

【Description】

Gets the encoder mute status.

【Parameters】

Parameter name	Description	Input/Output
AeChn	Audio encoding channel number. Value range: [0, AENC_MAX_CHN_NUM].	Input
Enable	Audio encoding channel mute status pointer.	output

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.4 Audio Decoding

2.4.1 ES_ADEC_CreateChn

【Function body】

```
ES_S32 ES_ADEC_CreateChn (
                ADEC_CHN AdChn,
                const ADEC_CHN_ATTR_S Attr)
```

【Description】

Creates an audio decoding channel.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input
Attr	Channel attribute pointer.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Audio decoding supports G711a, G711μ, G722, G726, AAC-LC, AAC-HEv1/v2, MP2L2, MP3, and AMR.
- Properties of audio decoding need to match the output device properties, such as sampling rate, frame length (number of sample points per frame), etc.
- The buffer size is in frames, and the value range is [2, MAX_AUDIO_FRAME_NUM]. It

is recommended to configure it to greater than 10. A buffer configuration that is too small may cause frame loss and other abnormalities.

- This interface can only be used before the channel is created (or after it is destroyed). If the channel has been created, it returns that the channel has been created.

2.4.2 ES_ADEC_DestroyChn

【Function body】

ES_S32 ES_ADEC_DestroyChn (ADEC_CHN AdChn)

【Description】

Destroys the audio decoding channel.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

If the channel is being acquired/released or frames are being sent, these operations will immediately return failure if the channel is destroyed.

2.4.3 ES_ADEC_SendStream

【Function body】

ES_S32 ES_ADEC_SendStream (

ADEC_CHN AdChn,
const AUDIO_STREAM_S* Stream,
ES_BOOL Block)

【Description】

Sends stream to the audio decoding channel.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input
Stream	Audio stream.	Input
Block	Blocking flag. ES_TRUE: blocking; ES_FALSE: non-blocking.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- When creating a decoding channel, you can specify the decoding mode as pack or stream. The pack mode is used when the packet boundary of the bitstream is known, such as when obtaining the bitstream directly from AENC or reading from a file where the exact frame boundary is known (e.g., in the case of fixed-length speech encoding, it is easy to determine the boundary). Pack mode is efficient in such scenarios. On the other hand, the stream mode is used when the packet boundary of the bitstream is uncertain. This mode is less efficient and may introduce delays. If there are ongoing operations such as obtaining/releasing bitstreams or sending frames, destroying the channel will immediately cause these operations to fail.
- LPCM decoding only supports pack mode.
- When sending data, developers ust ensure that the channel has been created, otherwise failure will be returned directly. If the channel is destroyed during the data sending process, failure will be returned immediately.
- Supports blocking or non-blocking mode for sending code stream.
- When the code stream setted as blocking mode, if the buffer used for decoded audio frame is full, this interface will be blocked until the decoded audio frame data is taken away or the ADEC channel is destroyed.
- Make sure the code stream data sent to the ADEC channel is correct, otherwise it may cause the decoder to exit abnormally.

2.4.4 ES_ADEC_ClearChnBuf

【Function body】

ES_S32 ES_ADEC_ClearChnBuf (ADEC_CHN AdChn)

【Description】

Clears the current audio data buffer in the ADEC channel.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- It is required that the decoding channel has been created. If the channel has not been created, an error code that does not exist in the channel will be returned.
- When using this interface, it is not recommended to use stream mode decoding. When using stream mode decoding to clear the cache, user needs to ensure that after

clearing the cache, the data sent to the decoder must be a complete frame of code stream, otherwise the decoder may not work properly.

- Regardless of whether stream decoding is used, synchronization between the operation of sending data for decoding and the operation of clearing the cache must be ensured.

2.4.5 ES_ADEC_RegisterDecoder

【Function body】

```
ES_S32 ES_ADEC_RegisterDecoder (
        ES_S32* Handle,
        const ADEC_DECODER_S* Decoder)
```

【Description】

Register decoder.

【Parameters】

Parameter name	Description	Input/Output
Handle	Register handle.	Input
Decoder	Decoder attribute structure.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- The user registers a decoder with the ADEC module by passing in the decoder attribute structure and returns the registration handle. The user can finally unregister the decoder through the registration handle.
- The ADEC module can register up to 20 decoders, and it has registered five decoders: LPCM, G711a, G711u, G726, and ADPCM.
- The same decoding protocol does not allow repeated registration of decoders. For example, if a G726 decoder has been registered, registration of another G726 decoder is not allowed.

2.4.6 ES_ADEC_UnRegisterDecoder

【Function body】

```
ES_S32 ES_ADEC_UnRegisterDecoder (ES_S32 Handle)
```

【Description】

Unregister the decoder.

【Parameters】

Parameter name	Description	Input/Output
Handle	Registration handle(the handle obtained when registering the decoder).	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Normally there is no need to unregister the decoder.
- Before unregistering a decoder, developers need to destroy all decoding channels created through the decoder. Calling this interface if it is not destroyed or during the destruction process will return an error.

2.4.7 ES_ADEC_GetFrame

【Function body】

```
ES_S32 ES_ADEC_GetFrame (
        ADEC_CHN AdChn,
        AUDIO_FRAME_INFO_S* FrmInfo,
        ES_BOOL Block)
```

【Description】

Gets the audio decoding frame data.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input
FrmInfo	Audio frame data structure.	output
Block	Whether to obtain in blocking mode.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Must be called after the ADEC channel is created.
- When using this interface to obtain decoded frame data, it is recommended to send the code stream in frames.
- When using this interface to obtain audio frame data, if the transmission code stream is sent by stream, be sure to ensure the timeliness of obtaining the decoded frame data, otherwise an exception will occur.
- When using this interface to obtain audio data, please cancel the binding relationship between ADEC and AO, otherwise the obtained frames will be discontinuous.

2.4.8 ES_ADEC_ReleaseFrame

【Function body】

```
ES_S32 ES_ADEC_ReleaseFrame (
        ADEC_CHN AdChn,
        const AUDIO_FRAME_INFO_S* FrmInfo)
```

【Description】

Release the acquired audio decoding frame data.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input
FrmInfo	Audio frame data structure.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

【Note】

- Must be called after the ADEC channel is created.
- This interface must be used in conjunction with the interface ES_AENC_GetFrame.

2.4.9 ES_ADEC_SendEndOfStream

【Function body】

```
ES_S32 ES_ADEC_SendEndOfStream (
        ADEC_CHN AdChn,
        ES_BOOL Instant)
```

【Description】

Sends the code stream end identifier to the decoder and clear the code stream buffer.

【Parameters】

Parameter name	Description	Input/Output
AdChn	Channel number. Value range: [0, ADEC_MAX_CHN_NUM).	Input
Instant	Whether to clear the cached data inside the decoder immediately. Ranges: ES_FALSE: delayed clearing. The internal buffer of the decoder will not be cleared immediately. The data is stored and decoding will continue until the remaining buffer is less than one frame of data and the clearing operation is performed. ES_TRUE: Clear the decoder’ s internal cache data immediately.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.5 Audio Codec

Audio Codec mainly provides operations on hardware devices through the alsa-lib interface to realize reading and writing of Audio Codec controls.

2.5.1 ES_ACODEC_SetValue

【Function body】

```
ES_S32 ES_ ACODEC_SetValue (
        AUDIO_DEV DevId,
        const char* Control,
        const char* Value)
```

【Description】

Sets the value of the specified audio codec control.

【Parameters】

Parameter name	Description	Input/Output
DevId	Audio sound card device number. Value range: refer to Table 3-1	Input
Control	The name of the audio codec control.	Input
Value	The value to set for the audio codec control.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

2.5.2 ES_ACODEC_GetValue

【Function body】

```
ES_S32 ES_ ACODEC_GetValue(
        AUDIO_DEV DevId,
        const char* Control,
        const char* Value,
        ES_U32 Size)
```

【Description】

Get the current value of specified audio codec control on the audio device.

【Parameters】

Parameter name	Description	Input/Output
DevId	Audio sound card device number. value range: refer to Table 3-1	Input
Control	Audio The name of the Codec control.	Input
Value	Buffer used to store retrieved values.	output
Size	The size of the buffer in bytes.	Input

【Return】

Return Value	Description
0	success.
Non 0	Fail, refers to the error code .

3. Data Types And Data Structures

3.1 Audio Input And Output

The data structures of AI and AO are defined as follows:

- AUDIO_CARD: Defines the audio sound card device number.
- AUDIO_DEV: Defines the audio device number.
- AI_VQE_MASK_HPF: MASK of VQE HPF.
- AI_VQE_MASK_RNR: MASK of VQE RNR.
- AI_VQE_MASK_DRC: MASK of VQE DRC.
- AI_VQE_MASK_EQ: MASK of VQE EQ.
- AI_VQE_MASK_AGC: MASK of VQE AGC.
- AO_VQE_MASK_HPF: MASK of AO VQE HPF.
- AO_VQE_MASK_ANR: MASK of AO VQE ANR.
- AO_VQE_MASK_AGC: MASK of AO VQE AGC.
- AO_VQE_MASK_EQ: MASK of AO VQE EQ.
- MAX_AUDIO_FILE_PATH_LEN: The maximum length limit of the path of the audio saving file.
- AUDIO_SAMPLE_RATE_E: Defines the audio sampling rate.
- AUDIO_BIT_WIDTH_E: Defines the audio sampling precision.
- AUDIO_SOUND_MODE_E: Defines the audio channel mode.
- AIO_ATTR_S: Defines the audio input and output device attribute structure.
- AI_CHN_PARAM_S: Defines the channel Parameters structure.
- AUDIO_FRAME_S: Defines the audio frame data structure.
- AUDIO_AEC_FRAME_S: Defines the echo cancellation reference frame information structure.
- AUDIO_AGC_CONFIG_S: Defines the audio automatic gain control configuration information structure.
- AI_AEC_CONFIG_S: Defines the audio echo cancellation configuration information structure.
- AUDIO_ANR_CONFIG_S: Defines the audio voice noise reduction function configuration information structure.
- AUDIO_HPF_CONFIG_S: Defines the audio high-pass filtering function configuration information structure.
- AI_RNR_CONFIG_S: Defines the audio recording noise elimination function configuration information structure.
- VQE_EQ_BAND_NUM: Defines the number of frequency bands that the EQ function can adjust.
- AUDIO_EQ_CONFIG_S: Defines the audio equalizer function configuration information structure.
- AI_DRC_CONFIG_S: Defines the audio dynamic compression control function configuration information structure.
- AI_VQE_CONFIG_S: Defines the AI voice quality enhancement configuration information

structure.

- AO_VQE_CONFIG_S: Defines the AO voice quality enhancement configuration information structure.
- AUDIO_STREAM_S: Defines the audio stream structure.
- AO_CHN_STATE_S: Data buffer status structure of the audio output channel .
- AUDIO_TRACK_MODE_E: Audio device channel mode type.
- AUDIO_FADE_RATE_E: Audio device fade rate type.
- AUDIO_FADE_S: Audio device fade setting structure.
- AUDIO_SAVE_FILE_INFO_S: Defines the audio save file function configuration information structure.
- AUDIO_FILE_STATUS_S: Defines the audio file saving status structure.

3.1.1 AUDIO_CARD

【Description】

Define the audio sound card device number.

【definition】

typedef ES_S32 AUDIO_CARD

【Note】

AUDIO_CARD should be an integer >=0 according to alsa rules.

【See Also】

Card No	Description
0	External Codec 1
1	External Codec 2
2	External Codec 0 or HDMI

3.1.2 AUDIO_DEV

【Description】

Define the audio device number.

【Definition】

typedef ES_S32 AUDIO_DEV

【Note】

AUDIO_DEV should be an integer > =0 according to alsa rules.

Usually use USB HUB expands USB devices.

Device No	Description
0	Card0 corresponds to external codec 1 Card 2 supports HDMI
1	Card1 corresponds to external codec 2 Card2 corresponds to external codec 0

【AI And AO Device Number】

Table 3-1

Model	Card ID	Device ID
AI	[0, 2]	[0,1], 0: hdmi device; 1: Codec device
AO	[0, 2]	[0,1], 0: hdmi device; 1: Codec device
AI and AO USB expansion	USB expansion sound card ID, supports up to 12 USB sound cards according to user requirement, range [3, x]	USB extension device ID

3.1.3 AI_VQE_MASK_HPF

【Description】

Defines the VQE HPF function.

【Definition】

#define AI_VQE_MASK_HPF 0x1

【See Also】

Assigning a value to the AI_VQE_CONFIG_S structure member OpenMask indicates turning on the HPF function. For example, AI_VQE_MASK_HPF, means turning on HPF functions.

3.1.4 AI_VQE_MASK_RNR

【Description】

Defines the VQE RNR function.

【Definition】

#define AI_VQE_MASK_RNR 0x20

3.1.5 AI_VQE_MASK_DRC

【Description】

Defines the VQE DRC function.

【Definition】

#define AI_RECORDVQE_MASK_DRC 0x4

3.1.6 AI_VQE_MASK_EQ

【Description】

Defines the VQE EQ function.

【Definition】

#define AI_VQE_MASK_EQ 0x2

3.1.7 AI_VQE_MASK_AGC

【Description】

Defines the VQE AGC function.

【Definition】

#define AI_RECORDVQE_MASK_AGC 0x10

3.1.8 AO_VQE_MASK_HPF

【Description】

Defines the AO VQE HPF function.

【Definition】

#define AO_VQE_MASK_HPF 0x1

【See Also】

Assigning a value to the AO_VQE_CONFIG_S structure member OpenMask indicates turning on the HPF function. For example, AO_VQE_MASK_AGC | AO_VQE_MASK_HPF means turning on the AGC and HPF functions.

3.1.9 AO_VQE_MASK_ANR

【Description】

Defines the AO VQE ANR function.

【Definition】

#define AO_TALKVQE_MASK_ANR 0x2

3.1.10 AO_VQE_MASK_AGC

【Description】

Defines the AO VQE AGC function.

【Definition】

#define AO_VQE_MASK_AGC 0x4

3.1.11 AO_VQE_MASK_EQ

【Description】

Defines the AO VQE EQ function.

【Definition】

#define AO_VQE_MASK_EQ 0x8

3.1.12 MAX_AUDIO_FILE_PATH_LEN

【Description】

Defines the maximum length limit for the path to the audio save file.

【Definition】

```
#define MAX_AUDIO_FILE_PATH_LEN 256
```

【See Also】

AUDIO_SAVE_FILE_INFO_S

3.1.13 AUDIO_SAMPLE_RATE_E

【Description】

Defines the audio sample rate.

【Definition】

```
typedef enum {
    AUDIO_SAMPLE_RATE_8000 =8000, /* 8K samplerate */
    AUDIO_SAMPLE_RATE_12000 =12000, /* 12K samplerate */
    AUDIO_SAMPLE_RATE_11025 =11025, /* 11.025K samplerate */
    AUDIO_SAMPLE_RATE_16000 =16000, /* 16K samplerate */
    AUDIO_SAMPLE_RATE_22050 =22050, /* 22.050K samplerate */
    AUDIO_SAMPLE_RATE_24000 =24000, /* 24K samplerate */
    AUDIO_SAMPLE_RATE_32000 =32000, /* 32K samplerate */
    AUDIO_SAMPLE_RATE_44100 =44100, /* 44.1K samplerate */
    AUDIO_SAMPLE_RATE_48000 =48000, /* 48K samplerate */
    AUDIO_SAMPLE_RATE_64000 =64000, /* 64K samplerate*/
    AUDIO_SAMPLE_RATE_96000 =96000, /* 96K samplerate*/
    AUDIO_SAMPLE_RATE_BUTT,
} AUDIO_SAMPLE_RATE_E
```

【Members】

Member name	Description
AUDIO_SAMPLE_RATE_8000	8kHz sampling rate.
AUDIO_SAMPLE_RATE_12000	12kHz sampling rate.
AUDIO_SAMPLE_RATE_11025	11.025kHz sampling rate.
AUDIO_SAMPLE_RATE_16000	16kHz sampling rate.
AUDIO_SAMPLE_RATE_22050	22.05kHz sampling rate.
AUDIO_SAMPLE_RATE_24000	24kHz sampling rate.
AUDIO_SAMPLE_RATE_32000	32kHz sampling rate.
AUDIO_SAMPLE_RATE_44100	44.1kHz sampling rate.
AUDIO_SAMPLE_RATE_48000	48kHz sampling rate.
AUDIO_SAMPLE_RATE_64000	64kHz sampling rate.
AUDIO_SAMPLE_RATE_96000	96kHz sampling rate.

【Note】

- The enumeration values do not start from 0, but instead correspond to the actual sampling rate values.
- The 96kHz sampling rate is only applicable to AO playback. It does not support data recording, resampling, VQE, and other processing..

【See Also】

AIO_ATTR_S

3.1.14 AUDIO_BIT_WIDTH_E

【Description】

Defines the audio sampling precision.

【Definition】

```
typedef enum {
    AUDIO_BIT_WIDTH_8 = 0, /* 8bit width */
    AUDIO_BIT_WIDTH_16 = 1, /* 16bit width*/
    AUDIO_BIT_WIDTH_24 = 2, /* 24bit width*/
    AUDIO_BIT_WIDTH_32 = 3, /* 32bit width*/
    AUDIO_BIT_WIDTH_BUTT,
} AUDIO_BIT_WIDTH_E
```

【Member】

Member name	Description
AUDIO_BIT_WIDTH_8	The sampling accuracy is 8 bits wide.
AUDIO_BIT_WIDTH_16	The sampling accuracy is 16 bits wide.
AUDIO_BIT_WIDTH_24	The sampling accuracy is 24bit bit width.
AUDIO_BIT_WIDTH_32	The sampling precision is 32bit wide.

【See Also】

AIO_ATTR_S

3.1.15 AUDIO_SOUND_MODE_E

【Description】

Defines the audio channel mode.

【Definition】

```
typedef enum {
    AUDIO_SOUND_MODE_MONO = 0, /*mono*/
    AUDIO_SOUND_MODE_STEREO = 1, /*stereo*/
    AUDIO_SOUND_MODE_BUTT
} AUDIO_SOUND_MODE_E
```

【Members】

Member name	Description
AUDIO_SOUND_MODE_MONO	Mono.
AUDIO_SOUND_MODE_STEREO	Stereo.

【See Also】

AIO_ATTR_S

3.1.16 AIO_ATTR_S

【Description】

Define the audio input and output device property structure.

【Definition】

```
typedef struct {
    AUDIO_SAMPLE_RATE_E Samplerate;
    AUDIO_BIT_WIDTH_E Bitwidth;
    AUDIO_SOUND_MODE_E Soundmode;
    ES_U32 FrmNum;
    ES_U32 PtNumPerFrm;
    ES_U32 ChnCnt;
} AIO_ATTR_S
```

【Members】

Member name	Description
Samplerate	Audio sample rate. Static properties.
Bitwidth	Audio sampling accuracy. Static properties.
Soundmode	Audio channel mode. Static properties.
FrmNum	Number of cached frames. Static properties.
PtNumPerFrm	The number of sampling points per frame. Static properties.
ChnCnt	Number of channels supported. Static properties. AI lacks this parameter.

【See Also】

ES_AI_SetPubAttr

ES_AO_SetPubAttr

3.1.17 AI_CHN_PARAM_S

【Description】

Define the channel Parameters structure.

【Definition】

```
typedef struct {
    ES_U32 UsrFrmDepth;
} AI_CHN_PARAM_S
```

【Members】

Member name	Description
UsrFrmDepth	Audio frame buffer depth.

3.1.18 AUDIO_FRAME_S

【Description】

Define the audio frame structure.

【Definition】

```
typedef struct {
    AUDIO_BIT_WIDTH_E Bitwidth;
    AUDIO_SOUND_MODE_E Soundmode;
    ES_U8 *VirAddr;
    ES_U64 TimeStamp;
    ES_U32Seq;
    ES_U32 Len;
} AUDIO_FRAME_S
```

【Members】

Member name	Description
Bitwidth	Audio sampling accuracy.
Soundmode	Audio channel mode.
VirAddr	Audio frame data virtual address.
TimeStamp	Audio frame timestamp. (temporarily invalid) Measured in μs.
Seq	Audio frame number.
Len	Audio frame length. In bytes.

【Note】

Len refers to the data length of a single channel.

3.1.19 AUDIO_AEC_FRAME_S

【Description】

Defines the AEC reference frame information structure.

【Definition】

```
typedef struct {
    AUDIO_FRAME_S *RefFrame; /* AEC reference audio frame */
    ES_BOOL Valid; /* whether frame is valid */
    ES_BOOL SysBind; /* whether is sysbind */
} AUDIO_AEC_FRAME_S
```

【Members】

Member name	Description
RefFrame	AEC reference frame structure.
Valid	Reference frame valid flag. Ranges: ES_TRUE: The reference frame is valid. ES_FALSE: The reference frame is invalid. When invalid, this reference frame cannot be used for echo cancellation.
SysBind	Whether AI and AENC are system bound.

3.1.20 AUDIO_AGC_CONFIG_S

【Description】

Defines the audio AGC configuration information structure.

【Definition】

```
typedef struct {
    ES_BOOL UsrMode;
```

ES_U8 TargetLevel;

ES_U8 CompressionGain;

ES_U8 LimiterEnable;

} AUDIO_AGC_CONFIG_S

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off
TargetLevel	Target level, this value is the maximum level threshold after AGC processing; range: [-30 ~ 0]dB; adjustment step 1dB
CompressionGain	Gain amplitude, controls the degree of closeness to the target gain. When the signal is small, a larger gain amplitude is needed to make the signal reach the target gain level; range: [0, 90]dB; adjustment step 1dB
LimiterEnable	Amplitude limiting is enabled. When limiting is enabled, smoothing will be performed when the signal amplitude is close to the target gain; value [0, 1]

【Note】

- The above advanced parameters only take effect when the user mode is turned on, otherwise they are configured according to the corresponding default values.
- When configuring parameters, the correctness of advanced parameters will only be checked when the user mode is turned on. Only correct advanced parameters can be configured successfully.

【See Also】

AI_VQE_CONFIG_S

AO_VQE_CONFIG_S

3.1.21 AI_AEC_CONFIG_S

【Description】

Defines the AEC configuration information structure.

【Definition】

typedef struct {

ES_BOOL UsrMode;

ES_U8 EchoMode;

} AUDIO_AEC_CONFIG_S

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off
EchoMode	Echo cancellation suppression level, the higher the level, the better the effect. Range: [0, 1].

3.1.22 AUDIO_ANR_CONFIG_S

【Description】

Define the ANR configuration information structure.

【Definition】

```
typedef struct {
    ES_BOOL UsrMode;
    ES_U8 Mode;
} AUDIO_ANR_CONFIG_S
```

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off
Mode	Noise reduction level, the higher the level, the more obvious the effect; range: [0, 3]..

【Note】

- The above advanced parameters only take effect when the user mode is turned on, otherwise they are configured according to the corresponding default values.
- When configuring parameters, the correctness of advanced parameters will only be checked when the user mode is turned on. Only correct advanced parameters can be configured successfully.

【See Also】

AI_VQE_CONFIG_S

AO_VQE_CONFIG_S

3.1.23 AUDIO_HPF_CONFIG_S

【Description】

Defines the HPF configuration information structure.

【definition】

```
typedef struct {
```

```
    ES_BOOL UsrMode;
    ES_U16 HpfFreq;
} AUDIO_HPF_CONFIG_S
```

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off.
hpF	HPF cutoff frequency; range: [80, 200]Hz.

【Note】

- The above advanced parameters only take effect when the user mode is turned on, otherwise they are configured according to the corresponding default values.
- When configuring parameters, the correctness of advanced parameters will only be checked when the user mode is turned on. Only correct advanced parameters can be configured successfully.

【See Also】

AI_VQE_CONFIG_S

AO_VQE_CONFIG_S

3.1.24 VQE_EQ_BAND_NUM

【Description】

Defines the number of frequency bands that the EQ function can adjust.

【Definition】

```
#define VQE_EQ_BAND_NUM 10
```

【See Also】

AUDIO_EQ_CONFIG_S

3.1.25 AUDIO_EQ_CONFIG_S

【Description】

Define the EQ configuration information structure.

【definition】

```
typedef struct {
    ES_BOOL UsrMode;
    ES_S8 GaindB[VQE_EQ_BAND_NUM];
} AUDIO_EQ_CONFIG_S
```

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off.
GaindB	EQ band gain adjustment, the frequency bands are 32Hz, 62Hz, 125Hz, 250Hz, 500Hz, 1kHz, 2kHz, 4kHz, 8kHz, 16kHz. 16kHz is invalid when the working sampling rate is 16 kHz, and 8k and 16kHz are invalid when the working sampling rate is 8 kHz. The value range of each frequency band is: [-15, 15]dB, with an adjustment step of 1dB.

3.1.26 AI_DRC_CONFIG_S

【Description】

Defines the audio automatic volume control function configuration information structure.

【Definition】

```
typedef struct {
    ES_BOOL UsrMode;
    ES_S8 Threshold;
    ES_U8 Knee;
    ES_U8 Ratio;
    ES_U16 Attack;
    ES_U16 Release;
    ES_S8 PostGain;
} AI_DRC_CONFIG_S
```

【Members】

Member name	Description
UsrMode	Whether to use user mode: 0: automatic mode 1: User mode Default is 0 off
Threshold	Compression threshold, indicating the level at which the signal begins to be compressed, value range: [-90, 0]dB.
Knee	The soft knee point range means setting a soft knee point above the compression threshold. After exceeding the soft knee point range, it is a linear compression section. The value range is: [0, 90]dB .
Ratio	Compression ratio, indicating the compression ratio of the linear compression segment, value range: [1, 10].
Attack	The time (ms) when the signal changes from large to small, value range : [10, 500].
Release	The time for the signal to change from small to large (ms), value range : [10 , 500].
PostGain	Post-gain refers to the overall gain adjustment of the signal after compression, [-10, 10]dB.

【Note】

- The advanced parameters only take effect when the user mode is enabled, otherwise they are configured according to the corresponding default values.
- When configuring parameters, the correctness of advanced parameters will only be checked when the user mode is turned on. Only correct advanced parameters can be configured successfully.

3.1.27 AI_VQE_CONFIG_S

【Description】

Defines the AI voice quality enhancement configuration information structure.

【Definition】

```
typedef struct {
    ES_U32 OpenMask;
    ES_S32 WorkSampleRate; /* Sample Rate: 8kHz/16kHz/32kHz/48kHz */
    ES_S32 FrameSamples;
    AUDIO_HPF_CONFIG_S HpfConfig;
    AUDIO_EQ_CONFIG_S EqConfig;
    AI_DRC_CONFIG_S DrcConfig;
    AUDIO_ANR_CONFIG_S AnrConfig;
    AUDIO_AGC_CONFIG_S AgcConfig;
} AI_VQE_CONFIG_S
```

【Members】

Member name	Description
OpenMask	MASK value for enabling each function of VQE.
WorkSampleRate	Working sampling frequency. This Parameters is the working sampling rate of the internal functional algorithm.
FrameSamples	The frame length of VQE, that is, the number of sampling points.
HpfConfig	Configuration information related to the high-pass filter function.
EqConfig	Weighing instrument related configuration information.
DrcConfig	Dynamic compression control function configuration information.
AnrConfig	Configuration information related to the voice noise reduction function.
AgcConfig	Configuration information related to automatic gain control.

3.1.28 AO_VQE_CONFIG_S

【Description】

Defines the AO voice quality enhancement configuration information structure.

【Definition】

```
typedef struct {
    ES_U32 OpenMask;
    ES_S32 WorkSampleRate; /* Sample Rate: 8 k Hz/16 k Hz /32kHz /48KHz. */
    ES_S32 FrameSample;
    AUDIO_HPF_CONFIG_S HpfCfg;
    AUDIO_ANR_CONFIG_S AnrCfg;
    AUDIO_AGC_CONFIG_S AgcCfg;
    AUDIO_EQ_CONFIG_S EqCfg;
} AO_VQE_CONFIG_S
```

【Members】

Member name	Description
OpenMask	MASK value for enabling each function of VQE.
WorkSampleRate	Working sampling frequency. This Parameters is the working sampling rate of the internal functional algorithm.
FrameSamples	The frame length of VQE, that is, the number of sampling points.
HpfConfig	Configuration information related to the high-pass filter function.
AnrConfig	Configuration information related to the voice noise reduction function.
AgcConfig	Configuration information related to automatic gain control.
EqConfig	Weighing instrument related configuration information.

3.1.29 AUDIO_STREAM_S

【Description】

Defines the audio stream structure.

【Definition】

```
typedef struct {
    ES_U8 ATTRIBUTE *Stream;
    ES_U32 Len;
    ES_U64 TimeStamp;
    ES_U32 Seq;
} AUDIO_STREAM_S
```

【Members】

Member name	Description
Stream	Audio stream data pointer.
Len	Audio stream length. In bytes.
TimeStamp	Audio stream timestamp.
Seq	Audio stream serial number.

【See Also】

ES_AENC_GetStream

3.1.30 AO_CHN_STATE_S

【Description】

The data cache status structure of the audio output channel.

【Definition】

```
typedef struct {
    ES_U32 ChnTotalNum;
    ES_U32 ChnFreeNum;
    ES_U32 ChnBusyNum;
} AO_CHN_STATE_S
```

【Members】

Member name	Description
ChnTotalNum	The total number of buffer blocks for the output channel.
ChnFreeNum	The number of free cache blocks available.
ChnBusyNum	The number of occupied cache blocks.

【See Also】

ES_AO_QueryChnStat

3.1.31 AUDIO_TRACK_MODE_E

【Description】

Defines the audio device channel mode type.

【Definition】

```
typedef enum {
    AUDIO_TRACK_NORMAL = 0,
    AUDIO_TRACK_BOTH_LEFT = 1,
    AUDIO_TRACK_BOTH_RIGHT = 2,
    AUDIO_TRACK_EXCHANGE = 3,
    AUDIO_TRACK_MIX = 4,
    AUDIO_TRACK_LEFT_MUTE = 5,
    AUDIO_TRACK_RIGHT_MUTE = 6,
    AUDIO_TRACK_BOTH_MUTE = 7,
    AUDIO_TRACK_BUTT
} AUDIO_TRACK_MODE_E
```

【Members】

Member name	Description
AUDIO_TRACK_NORMAL	Normal mode, no processing is done.
AUDIO_TRACK_BOTH_LEFT	Both channels are all left channel sound.
AUDIO_TRACK_BOTH_RIGHT	Both channels are all right channel sounds.
AUDIO_TRACK_EXCHANGE	The left and right channel data are interchanged, the left channel is the right channel sound, and the right channel is the left channel sound.
AUDIO_TRACK_MIX	The left and right channel output is the sum of the left and right channels (mixing).
AUDIO_TRACK_LEFT_MUTE	The left channel is muted and the right channel plays the original right channel sound.
AUDIO_TRACK_RIGHT_MUTE	The right channel is muted and the left channel plays the original left channel sound.
AUDIO_TRACK_BOTH_MUTE	The left and right channels are muted.

【See Also】

ES_AI_SetTrackMode

ES_AO_SetTrackMode

3.1.32 AUDIO_FADE_S

【Description】

Audio output device fade configuration structure.

【Definition】

typedef struct {

ES_BOOL Fade;

ES_U32 FadeInRate;

ES_U32 FadeOutRate;

} AUDIO_FADE_S

【Members】

Member name	Description
Fade	Whether to enable the fade-in and fade-out function. ES_TRUE: Turn on the fade function. ES_FALSE: Turn off the fade function.
FadeInRate	Audio output device volume fade-in time. Range: 500~5000/ms.
FadeOutRate	Audio output device volume fade-out time. Range: 500~5000/ms.

【See Also】

ES_AO_SetMute

3.1.33 AUDIO_SAVE_FILE_INFO_S

【Description】

Defines the audio save file function configuration information structure.

【Definition】

typedef struct {

ES_BOOL Cfg;

ES_CHAR FilePath[MAX_AUDIO_FILE_PATH_LEN];

ES_CHAR FileName[MAX_AUDIO_FILE_NAME_LEN];

ES_U32 FileSize; /*in KB*/

} AUDIO_SAVE_FILE_INFO_S

【Members】

Member name	Description
cfg	Configure the enable switch.
FilePath	Audio file saving path.
FileName	Audio file save name.
FileSize	File size, unit KB.

【Note】

FilePath uses the current location by default, FileName uses "default" by default, and FileSize uses 1M by default. The FileName is only part of the real file name of the audio file

saved. For example, the real names of the three files saved by Card 0 device 0 under the 8k sampling rate are:

Sin_AiCard0_Dev0_8k_default.pcm, Rin_AiCard0_Dev0_8k_default.pcm,

Sou_AiCard0_Dev0_8k_default.pcm.

3.1.34 AUDIO_FILE_STATUS_S

【Description】

Defines the audio file saving status structure.

【Definition】

```
typedef struct {
    ES_BOOL Saving;
} AUDIO_FILE_STATUS_S
```

【Members】

Member name	Description
Saving	Whether it is in the file saving state. ES_TRUE: in file saving state; ES_FALSE: Not in file saving state.

3.2 Audio Encoding

3.2.1 AENC_MAX_CHN_NUM

【Description】

Defines the maximum number of audio encoding channels.

【Definition】

```
#define AENC_MAX_CHN_NUM 8
```

3.2.2 AENC_ATTR_G711_S

【Description】

Defines the G.711 encoding protocol attribute structure.

【Definition】

```
typedef struct {
    ES_U32 IsULaw;
} AENC_ATTR_G711_S;
```

【Members】

Member name	Description
IsULaw	1: μlaw 0: a law.

3.2.3 AENC_ATTR_G722_S

【Description】

Defines the G.712 encoding protocol attribute structure.

【Definition】

```
typedef struct {
    int32_t BitRate;
} AENC_ATTR_G711_S;
```

【Members】

Member name	Description
BitRate	G.722 protocol code rate.

3.2.4 AENC_ATTR_G726_S

【Description】

Defines the audio sampling precision.

【Definition】

```
typedef struct {
    int32_t BitRate;
} AENC_ATTR_G726_S
```

【Members】

Member name	Description
BitRate	G.726 protocol code rate.

【See Also】

G726_BPS_E

3.2.5 AENC_ATTR_AMR_S

【Description】

Defines the audio sampling precision.

【Definition】

```
typedef struct {
    int32_t BitRate;
    int32_t Dtx;
    int32_t IsWb;
} AENC_ATTR_AMR_S
```

【Members】

Member name	Description
BitRate	AMR protocol ratio bit rate.
dbx	To enable or disable the Discontinuous Transmission (DTX) feature.
wxya	Used to configure whether the encoder should operate in wideband mode.

3.2.6 AENC_CHN_ATTR_S

【Description】

Defines the LPCM encoding protocol attribute structure.

【Definition】

```
typedef struct {
    PAYLOAD_TYPE_E Type; /*payload type ()*/
    ES_U32 PtNumPerFrm;
    ES_U32 BufSize; /*buf size [2~MAX_AUDIO_FRAME_NUM]*/
    ES_VOID ATTRIBUTE *Value; /*point to attribute of definite audio encoder*/
} AENC_CHN_ATTR_S
```

【Members】

Member name	Description
Type	Audio encoding protocol type. Static properties.
PtNumPerFrm	The frame length corresponding to the audio encoding protocol (the length of the audio frame received during encoding is less than or equal to this frame can be encoded).
BufSize	Audio encoding buffer size. Value range: [2, MAX_AUDIO_FRAME_NUM], in frames. Static properties.
Value	Specific protocol attribute pointer. Static properties.

3.2.7 AENC_ENCODER_S

【Description】

Defines the encoder attribute structure.

【Definition】

```
typedef struct {
    PAYLOAD_TYPE_E Type;
```

```
    ES_U32 MaxFrmLen;
    ES_CHAR Name[17];
    ES_S32(*FunOpenEncoder)
    (ES_VOID *EncoderAttr, ES_VOID **Encoder);
    ES_S32 (*FunEncodeFrm)(ES_VOID *Encoder, const ES_U8 *Input, ES_U32 InputSize,
                           ES_U8 *output, ES_U32 *outSize);
    ES_S32 (*FunCloseEncoder)(ES_VOID *Encoder);
} AENC_ENCODER_S
```

【Members】

Member name	Description
Type	Encoding protocol type.
MaxFrmLen	Maximum code stream length.
Name	Encoder name.
FunOpenEncoder	Open the function pointer of the encoder
FunEncodeFrm	encoded function pointer
FunCloseEncoder	Close encoder function pointer

3.3 Audio Decoding

3.3.1 MAX_AUDIO_FRAME_NUM

【Description】

Defines the maximum number of audio decoding buffer frames.

【Definition】

```
#define MAX_AUDIO_FRAME_NUM 300
```

3.3.2 ADEC_MAX_CHN_NUM

【Description】

Defines the maximum number of audio decoding channels.

【definition】

```
#define ADEC_MAX_CHN_NUM 16
```

3.3.3 ADEC_ATTR_G711_S

【Description】

Define the G.711 decoding protocol attribute structure.

【Definition】

```
typedef struct {
    ES_U32 IsULaw;
```

} ADEC_ATTR_G711_S

【Members】

Member name	Description
IsULaw	1: µlaw 0: a law.

3.3.4 ADEC_ATTR_G722_S

【Description】

Defines the G.722 decoding protocol attribute structure.

【Definition】

typedef struct {

int32_t BitRate;

} ADEC_ATTR_G726_S

【Members】

Member name	Description
BitRate	G.722 protocol code rate.

3.3.5 ADEC_ATTR_G726_S

【Description】

Defines the G.726 decoding protocol attribute structure.

【Definition】

typedef struct {

int32_t BitRate;

} ADEC_ATTR_G726_S

【Members】

Member name	Description
BitRate	G.726 protocol code rate.

3.3.6 ADEC_ATTR_AMR_S

【Description】

Defines the G.726 decoding protocol attribute structure.

【Definition】

typedef struct {

ES_S32 IsWb;

} ADEC_ATTR_AMR_S

【Members】

Member name	Description
IsWb	Used to configure whether the encoder should operate in wideband mode.

3.3.7 ADEC_ATTR_AAC_S

【Description】

Defines the G.726 decoding protocol attribute structure.

【Definition】

```
typedef struct {
    ADEC_AAC_PROFILE_E Profile;
    ADEC_AAC_TRANS_TYPE_E TransType;
    ES_S32 OutputFormat;
} ADEC_ATTR_AAC_S
```

【Members】

Member name	Description
Profile	Indicates AAC decoding profile.
TransType	AAC transfer type.
OutputFormat	Indicates that the output bit width is 1-16bit, 3-32bit, 1-24bit.

【See Also】

ADEC_AAC_PROFILE_E, ADEC_AAC_TRANS_TYPE_E

3.3.8 ADEC_MODE_E

【Description】

Defines the decoding method.

【Definition】

```
typedef enum {
    ADEC_MODE_PACK = 0,
    ADEC_MODE_STREAM,
    ADEC_MODE_BUTT
} ADEC_MODE_E
```

【Members】

Member name	Description
ADEC_MODE_PACK	Pack mode decoding.
ADEC_MODE_STREAM	Decoding in stream mode.

【Note】

- The pack method is used when the user confirms that the current stream packet is the encoding result of one frame of data. The decoder will directly decode it. If it is not one frame, the decoder will make an error. The efficiency of this mode is relatively high. If the stream packets encoded by the AENC module are not damaged, they can be decoded in this way. Library file: libesaudiosdk.so
- The stream mode is used when the user cannot confirm whether the current code stream packet is a frame of data. The decoder needs to judge the code stream and cache it. This working method is inefficient and is generally used for reading file code streams for decoding or is uncertain. The situation of code stream packet boundary. Of course, since the length of the speech encoding code stream is fixed, it is easy to determine the frame boundaries in the code stream. It is recommended to use the pack method for decoding.

3.3.9 ADEC_AAC_TRANS_TYPE_E

【Description】

Defines the decoding method.

【Definition】

```
typedef enum {
    ADEC_AAC_TRANS_TYPE_ADTS = 0,
    ADEC_AAC_TRANS_TYPE_LOAS = 1,
    ADEC_AAC_TRANS_TYPE_LATM_MCP1 = 2,
    ADEC_AAC_TRANS_TYPE_BUTT
} ADEC_AAC_TRANS_TYPE_E
```

【Members】

Member name	Description
ADEC_AAC_TRANS_TYPE_ADTS	Use ADTS (Audio Data Transport Stream) transmission format.
ADEC_AAC_TRANS_TYPE_LOAS	Use LOAS (Low Overhead Audio Stream) transmission format.
ADEC_AAC_TRANS_TYPE_LATM_MCP1	Use LATM (Low-Overhead Audio Transport Multiplex) transmission format.
ADEC_AAC_TRANS_TYPE_BUTT	Reserved enumeration values.

3.3.10 ADEC_AAC_PROFILE_E

【Description】

Defines the decoding method.

【Definition】

```
typedef enum {
    ADEC_AAC_TYPE_AACMAIN = 0,
    ADEC_AAC_TYPE_AACLC = 1,
```

```
    ADEC_AAC_TYPE_AACHE = 2,
    ADEC_AAC_TYPE_AACLD = 3,
    ADEC_AAC_TYPE_BUTT,
} ADEC_MODE_E
```

【Members】

Member name	Description
ADEC_AAC_TYPE_AACMAIN	Configure as main profile.
ADEC_AAC_TYPE_AACLC	AAC low complexity configuration is a low complexity configuration of AAC encoding.
ADEC_AAC_TYPE_AACHE	AAC high-efficiency configuration is a high-efficiency configuration of AAC encoding.
ADEC_AAC_TYPE_AACLD	AAC low latency configuration is a low latency configuration for AAC encoding.

3.3.11 ADEC_CHN_ATTR_S

【Description】

Defines the decoding channel attribute structure.

【Definition】

```
typedef struct {
    PAYLOAD_TYPE_E Type;
    ES_U32BufSize;
    ADEC_MODE_E Mode;
    ES_VOID ATTRIBUTE *Value;
} ADEC_CHN_ATTR_S
```

【Members】

Member name	Description
Type	Audio decoding protocol type. Static properties.
BufSize	Audio decoding buffer size. Value range: [2, MAX_AUDIO_FRAME_NUM], in frames. Static properties.
Mode	decoding method. Static properties.
Value	Specific protocol attribute pointer.

3.3.12 ADEC_DECODER_S

【Description】

Defines the decoder attribute structure.

【Definition】

```
typedef struct {
    PAYLOAD_TYPE_E Type;
    ES_CHAR Name[17];
    ES_S32 (*FunOpenDecoder)(ES_VOID *DecoderAttr, ES_VOID **Decoder);
    ES_S32 (*FunDecodeFrm)(ES_VOID *Decoder, const ES_U8 *InputBuffer, ES_S32 InputSize, ES_U8 *OutBuffer, ES_U32 *OutSize);
    ES_S32 (*FunGetFrmInfo)(ES_VOID *Decoder, ES_VOID *Info);
    ES_S32 (*FunCloseDecoder)(ES_VOID *Decoder);
    ES_S32 (*FunResetDecoder)(ES_VOID *Decoder);
} ADEC_DECODER_S
```

【Members】

Member name	Description
Type	Audio decoding protocol type.
Name	Decoder name.
FunOpenDecoder	Open the function pointer of the decoder.
FunDecodeFrm	Function pointer for decoding.
FunGetFrmInfo	Function pointer to obtain audio frame information.
FunCloseDecoder	Function pointer to close the decoder.
FunResetDecoder	Clear the cache buffer and reset the decoder.

3.3.13 AUDIO_FRAME_INFO_S

【Description】

Defines the decoded audio frame information structure.

【Definition】

```
typedef struct {
    AUDIO_FRAME_S *Frame;
    ES_U32Id;
} AUDIO_FRAME_INFO_S
```

【Members】

Member name	Description
Frame	Audio frame pointer.
ID	Index of the audio frame, range [0, 49].

3.3.14 ADEC_CHN_STATE_S

【Description】

Defines the decoded audio frame information structure.

【Definition】

```
typedef struct {
    ES_BOOL EndOfStream;
    ES_U32 BufferFrmNum;
    ES_U32 BufferFreeNum;
    ES_U32 BufferBusyNum;
} ADEC_CHN_STATE_S
```

【Members】

Member name	Description
EndOfStream	Decoding stream end status.
BufferFrmNum	The total number of buffer blocks in the decoding channel.
BufferFreeNum	The number of free cache blocks available.
BufferBusyNum	The number of occupied cache blocks.

4. Error Code

Error code	Value	Description
AI_ERR_INVALID_DEVID	0xA0096001	AI device invalid device ID
AI_ERR_ILLEGAL_PARAM	0xA0096003	AI device illegal parameters
AI_ERR_NULL_PTR	0xA0096006	AI device null pointer
AI_ERR_NOT_CONFIG	0xA0096007	AI device is not configured
AI_ERR_NOT_SUPPORT	0xA0096008	Operations not supported by AI devices
AI_ERR_NOT_PERM	0xA0096009	AI device does not have permission
AI_ERR_NOT_ENABLED	0xA0096005	AI device is not enabled
AI_ERR_NOMEM	0xA009600C	AI device out of memory
AI_ERR_NOBUF	0xA009600D	AI device buffer insufficient
AI_ERR_BUF_EMPTY	0xA009600E	AI device buffer is empty
AI_ERR_BUF_FULL	0xA009600F	AI device buffer is full
AI_ERR_SYS_NOTREADY	0xA0096010	AI device system is not ready
AI_ERR_BUSY	0xA0096012	AI equipment operations are busy
AI_ERR_VQE_ERR	0xA0096041	AI device VQE error
AO_ERR_INVALID_DEVID	0xA00A6001	AO device invalid device ID
AO_ERR_ILLEGAL_PARAM	0xA00A6003	AO device illegal parameters
AO_ERR_NULL_PTR	0xA00A6006	AO device null pointer
AO_ERR_NOT_CONFIG	0xA00A6007	AO device is not configured
AO_ERR_NOT_SUPPORT	0xA00A6008	Operations not supported by AO devices
AO_ERR_NOT_PERM	0xA00A6009	AO device does not have permission

AO_ERR_NOT_ENABLED	0xA00A6005	AO device is not enabled
AO_ERR_NOMEM	0xA00A600C	AO device has insufficient memory
AO_ERR_NOBUF	0xA00A600D	AO device buffer insufficient
AO_ERR_BUF_EMPTY	0xA00A600E	AO device buffer is empty
AO_ERR_BUF_FULL	0xA00A600F	AO device buffer is full
AO_ERR_SYS_NOTREADY	0xA00A6010	AO equipment system is not ready
AO_ERR_BUSY	0xA00A6012	AO equipment operation is busy
AO_ERR_VQE_ERR	0xA00A6041	AO device VQE error
AENC_ERR_INVALID_DEVID	0xA00B6001	AENC Invalid Device ID
AENC_ERR_INVALID_CHNID	0xA00B6002	AENC invalid channel ID
AENC_ERR_ILLEGAL_PARAM	0xA00B6003	AENC Parameters is illegal
AENC_ERR_EXIST	0xA00B6004	AENC channel already exists
AENC_ERR_UNEXIST	0xA00B6005	AENC channel does not exist
AENC_ERR_NULL_PTR	0xA00B6006	AENC null pointer usage
AENC_ERR_NOT_CONFIG	0xA00B6007	AENC is not configured
AENC_ERR_NOT_SUPPORT	0xA00B6008	Operations not supported by AENC
AENC_ERR_NOT_PERM	0xA00B6009	AENC permission is not allowed
AENC_ERR_NOMEM	0xA00B600C	AENC memory allocation failed
AENC_ERR_NOBUF	0xA00B600D	AENC buffer allocation failed
AENC_ERR_BUF_EMPTY	0xA00B600E	AENC buffer is empty
AENC_ERR_BUF_FULL	0xA00B600F	AENC buffer full
AENC_ERR_SYS_NOTREADY	0xA00B6010	AENC system is not ready
AENC_ERR_ENCODER_ERR	0xA00B6040	AENC encoder internal error
AENC_ERR_VQE_ERR	0xA00B6041	AENCVQE internal error
ADEC_ERR_INVALID_DEVID	0xA00C6001	Invalid ADEC device ID
ADEC_ERR_INVALID_CHNID	0xA00C6002	Invalid ADEC channel ID
ADEC_ERR_ILLEGAL_PARAM	0xA00C6003	ADEC Parameters is illegal
ADEC_ERR_EXIST	0xA00C6004	ADEC channel already exists
ADEC_ERR_UNEXIST	0xA00C6005	ADEC channel does not exist
ADEC_ERR_NULL_PTR	0xA00C6006	ADEC uses a null pointer
ADEC_ERR_NOT_CONFIG	0xA00C6007	ADEC is not configured
ADEC_ERR_NOT_SUPPORT	0xA00C6008	Operations not supported by ADEC
ADEC_ERR_NOT_PERM	0xA00C6009	ADEC operation permission is not allowed
ADEC_ERR_NOMEM	0xA00C600C	ADEC memory allocation failed
ADEC_ERR_NOBUF	0xA00C600D	ADEC buffer allocation failed
ADEC_ERR_BUF_EMPTY	0xA00C600E	ADEC buffer is empty
ADEC_ERR_BUF_FULL	0xA00C600F	ADEC buffer full
ADEC_ERR_SYS_NOTREADY	0xA00C6010	ADEC system is not ready
ADEC_ERR_DECODER_ERR	0xA00C6040	ADEC decoder internal error

ADEC_ERR_BUF_LACK	0xA00C6041	ADEC input is less than one frame