INGENIC®
T31 Audio API Reference

Date: 2022-04

Viewer: Jason Xu

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

INGENIC®

T31 Audio API Reference

Release history

Date	Revision	Change
2022-04	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co. Ltd

Add: Junzheng R&D Center, Phase II, Zhongguancun Software Park, Dongbeiwangxi Road, Haidian District, Beijing, China

Tel: 86-10-56345000

Fax: 86-10-56345001

Http: //www.ingenic.com

Content

1 Audio Module overview

1.1 Introduction of the module

Audio is divided into internal codec and external codec, and internal codec is divided into digital mic and analog mic.

1.2 Kernel Configuration

Run the make menuconfig command in the kernel source directory to go to the configuration page and perform the following configuration as required.

1) Amic configuration (Default configured)

```
Device Drivers --->
     <*>Sound card support --->
          <*>Open Sound System --->
             [*]Open Sound System of Xburst --->
               [*]Compile jzsound driver into ko
               [*] Jz On-Chip I2S driver
               [*] xburst internal coedc --->
                      [*] t10 internal codec
```

2) Dmic configuration

```
Device Drivers --->
     <*>Sound card support --->
        <*>Open Sound System --->
           [*]Open Sound System of Xburst --->
              [*]Compile jzsound driver into ko
              [*] Jz On-Chip I2S driver
              [*] xburst internal coedc --->
```

```
                    [*] t10 internal codec
                [*] jz soc t31 dmic enable
```

3) External Codec ES8374 configuration

```
Device Drivers --->
    <*>Sound card support --->
        <*>Open Sound System --->
            [*]Open Sound System of Xburst --->
            [*]Compile jzsound driver into ko
            [*] Jz On-Chip I2S driver
            [*] xburst internal coedc --->
                [*] t10 internal codec
            [*] jz soc t31 dmic enable
            [*] xburst external codec --->
                <*> JZ_v12 i2c controler 2 Interface support
```

1.3 Audio Driver

Driver code is located in the/opensource/drivers/audio/oss2; Verify if the kernel path in the Makefile is correct before compiling. The driver parameters are as follows.

1) The spk_gpio parameter is provided when the Audio driver is loaded to enable the power amplifier. The default driver configuration is PB31; If no driver is required to control this pin, set this parameter to spk_gpio=-1.

```
# insmod audio.ko spk_gpio=-1
```

2) Audio supports both Amic and Dmic functions; This function is disabled by default. If you use it, you need to configure both Amic and Dmic in the kernel. If Audio uses both Amic and Dmic, you can set parameters dmic_amic_sync =1 to ensure the data time difference between amic and dmic is fixed.

```
# insmod audio.ko dmic_amic_sync=1
```

2) Codec supports input ALC function for automatic control of input gain; This function is enabled by default. The alc_mode parameter is provided during driver loading to enable or disable ALC. When this parameter is set to 0, ALC is disabled. Configure the input gain manually.

3) After ALC is enabled, the IMP_AI_SetAlcGain interface is used to set the input relative gain. After the ALC was shut down, manually set the input gain using the IMP_AI_SetGain interface (See section 7.6 for API details).

```
# insmod audio.ko alc_mode=0
```

1.4 Keyword Description

HPF

HPF(High-Pass Filtering)

AGC

AGC(Automatic Gain Control)

AEC

AEC(Acoustic Echo Cancellation)

NS

NS(Nosie Suppression)

AO

AO(Audio output)

AI

AI(Audio Input)

DMIC

DMIC(Digtial mic)

AENC

AENC(Audio Encode)

ADEC

ADEC(Audio Dncode)

2 Audio SDK

2.1 Function module introduction

Audio function includes audio input, audio output, echo cancellation, audio encoding and audio decoding 5 modules.

There are devices and channels for audio input and audio output. One MIC is considered to be a Device, and a MIC can have multiple Channel inputs. In the same way that an SPK is considered a playback Device, an SPK can also have multiple Channel outputs. The current version of the audio API supports only one Channel per Device.

Echo cancellation is located in the audio input interface and is specified in the function description.

Audio encoding The current audio API supports PT_G711A, PT_G711U and PT_G726 audio encoding formats. If you need to add a new encoding mode, you need to register an encoder.

Audio Decoding The current audio API supports audio decoding in PT_G711A, PT_G711U and PT_G726 formats. If you need to add a new decoding mode, you need to register the decoder.

2.2 Describe

The following is a detailed description of each module.

2.2.1 Audio input

Audio input Device ID mapping: 0: corresponds to digital MIC. 1: corresponds to analog MIC

Audio input Channel The CURRENT API supports only one Channel.

Set the volume of audio input. The value ranges from -30 to 120. -30 indicates mute. 120 indicates magnifying the voice by 30dB. The step length is 0.5dB. Where 60 is a critical point of volume setting, the software does not increase or decrease the volume on this value, when the volume value is less than 60, with every drop 1, the volume decreases 0.5dB; When the volume value is greater than 60, the volume increases by 0.5dB for each increment of 1.

Figure 2-1 Audio input gain control flow chart

2.2.2 Audio Output

Audio output Device ID mapping. 0: corresponds to the default SPK. 1: corresponds to other SPKS

Audio output Channel The CURRENT API supports only one Channel.

Set the volume of the audio output. The value ranges from -30 to 120. -30 indicates mute. 120 indicates magnifying the voice by 30dB. The step length is 0.5dB. Where 60 is a critical point of volume setting, the software does not increase or decrease the volume on this value, when the volume value is less than 60, with every drop 1, the volume decreases 0.5dB; When the volume value is greater than 60, the volume increases by 0.5dB for each increment of 1.

Figure 2-2 Audio output gain control flow chart

2.2.3 Echo cancellation

Echo cancellation is a function of the audio input interface, so the audio input device and channel must be enabled before echo cancellation is enabled.

Echo cancellation supports audio sampling rates of 8K and 16K, and the number of data samples per frame is an integer multiple of 10ms audio data (for example, if the sampling rate is 8K, the input data is an integer multiple of 160bytes: 8000 x 2/100).

Echo cancellation for different devices, different packages, echo cancellation will have different effects.

Echo cancellation currently does not support auto-adaptation. Therefore, echo cancellation parameters are configured separately for different devices. The echo cancellation parameter file is stored in the /etc/webrtc_profile.ini configuration file. The format of the configuration file is as follows:

[Set_Far_Frame]

```
Frame_V=0.3
```

[Set_Near_Frame]

```
Frame_V=0.1
delay_ms=150
```

The content in the first label [Set_Far_Frame] represents the remote parameter, that is, the parameter of SPK playback data. Fram_V represents the audio amplitude ratio. Adjust this parameter to adjust the amplitude of playback data (this amplitude is only used for echo cancellation).

The first label [Set_Near_Frame] indicates the near-end parameters, that is, the recording data parameters on the MIC end.

Fram_V represents the audio amplitude ratio. Adjust this parameter to adjust the amplitude of the recorded data (this amplitude is only used for echo cancellation). Delay_ms Because the software and hardware have delay, and SPK and MIC are placed at a certain distance, SPK playback data will be sampled by MIC again, so there will be a certain delay in the presentation of SPK data in MIC data. The time indicates the time difference between playback data and recording data.

Note: For echo cancellation, all configuration parameters are in the /etc/webrtc_profile.ini file in the development board (for the meanings of each parameter, please refer to the specific manual). Echo cancellation takes effect after IMP_AI_EnableAec function is called by the application layer (it doesn't matter whether the speaker is enabled or not, Echo cancellation occurs whenever this function is called. Echo cancellation includes AGC (to control the volume value of recording), NS (to reduce noise), and HPF (high pass filtering), corresponding to [AGC], [HP], and [NS] in the configuration file. True indicates that the function is enabled, false indicates that the function is disabled, and both are enabled by default. The NS level in the configuration file is as follows: KLow, kModerate, kHigh, kVeryHigh four levels. AGC, NS, and HPF have separate functions that can be enabled only in simplex cases. Note that if AEC is enabled, there is no need to call AGC, NS, and HPF functions, because if AEC is enabled in the configuration file, data will be processed directly in AEC.

2.2.4 Audio coding

Audio encoding Currently audio API supports PT_G711A, PT_G711U and PT_G726 audio encoding,If a new encoding method is required, the IMP_AENC_RegisterEncoder interface is called to register the encoder.

2.2.5 Audio decoding

Audio Decoding Currently audio API supports PT_G711A, PT_G711U and PT_G726 audio decoding,If you need to add a new decoding method, you need to register it by calling the IMP_ADEC_RegisterDecoder interface the decoder.

2.2.6 Definition of related variables

2.2.6.1 Decoding way

Table 2-1 Decoding methods

ADEC_MODE_PACK	Pack decoding
ADEC_MODE_STREAM	Sensor driver

2.2.6.2 Audio sampling frequency

Table 2-2 Audio sampling frequency

AUDIO_SAMPLE_RATE_8000	AUDIO_SAMPLE_RATE_8000
8 KHZ sampling rate	8 KHZ sampling rate
AUDIO_SAMPLE_RATE_16000	AUDIO_SAMPLE_RATE_16000
16 KHZ sampling rate	16 KHZ sampling rate
AUDIO_SAMPLE_RATE_24000	AUDIO_SAMPLE_RATE_24000
24 KHZ sampling rate	24 KHZ sampling rate

2.2.6.3 Audio channel mode

Table 2-3 Audio channel modes

AUDIO_SOUND_MODE_MONO	single track
AUDIO_SOUND_MODE_STEREO	Double track

2.2.6.4 Noise suppression level

Table 2-4 Noise suppression levels

NS_LOW	Low level noise suppression
NS_MODERATE	Medium level noise suppression
NS_HIGH	High level noise suppression
NS_VERYHIGH	Highest level of noise suppression

2.3 AI Module API

API	function
IMP_AI_SetPubAttr	Set AI device attributes
IMP_AI_GetPubAttr	Obtain AI device attributes

IMP_AI_Enable	Enable the AI device
IMP_AI_Disable	Disable the AI device
IMP_AI_EnableChn	Enable the AI channel
IMP_AI_DisableChn	The AI channel is disabled
IMP_AI_PollingFrame	Polling AI channel Audio stream cache
IMP_AI_GetFrame	Gets the AI channel audio frame
IMP_AI_ReleaseFrame	Release AI channel audio frame
IMP_AI_SetChnParam	Set AI channel parameters
IMP_AI_GetChnParam	Obtain AI channel parameters
IMP_AI_EnableAec	Enable echo cancellation for AI channel and AO channel
IMP_AI_DisableAec	Disable the AI channel echo cancellation function
IMP_AI_EnableNs	Enable the noise suppression function of the AI channels
IMP_AI_DisableNs	Disables the AI channel noise suppression function
IMP_AI_EnableAgc	Enable automatic gain function for AI channels
IMP_AI_DisableAgc	Disable the AI channel automatic gain feature
IMP_AI_EnableHpf	Enable the high-pass filtering function for the AI channels
IMP_AI_SetHpfCoFrequency	Set the cutoff frequency of the high-pass filter for the AI channel
IMP_AI_DisableHpf	Disable the AI-channel high-pass filtering function
IMP_AI_SetVol	Set the volume of the AI channel
IMP_AI_GetVol	Get the volume of the AI channel
IMP_AI_SetVolMute	Mute the AI channel
IMP_AI_SetGain	Set up the gain of the AI channel
IMP_AI_GetGain	Gain gain for the AI channel
IMP_AI_SetAlcGain	Set the AI channel input gain

IMP_AI_GetAlcGain	Obtain the AIPga gain value
IMP_AI_GetFrameAndRef	Obtain the audio frames and the output reference frames for the AI channel
IMP_AI_EnableAecRefFrame	Open the fetch reference frame
IMP_AI_DisableAecRefFrame	Disables the AI channel to fetch the reference frame function

Table 3-1 AI API functions

2.3.1 IMP_AI_SetPubAttr

【Function】

Set the AI device properties.

【Grammar】

int IMP_AI_SetPubAttr(int audioDevId, IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
attr	Audio Input device property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Need to be called before the IMP_AI_Enable.

- sampling rate

Sampling rate, also called bit sampling frequency, defines the number of signals extracted from continuous signals and forming discrete signals per second, with units expressed by Herz Hz.The sampling rate of the audio represents the number of sampling points per second.The higher the sampling rate, the lower the distortion of the audio data obtained, and the relatively larger the amount of raw data.The T31 built-in codec supports a recording sampling rate of 8K / 16K / 24K / 32K / 44.1K / 48K / 96K.

- Sampling accuracy

  Sampling accuracy refers to the data width for each sampling point in the sampling channel.T31 currently supports only the 16bit sampling precision recordings.

- Audio channel mode

  T31 built-in codec recording support supports mono track.

- The number of cache frames

  The Number of cached frames for internal requests.

- Number of sampling points per frame

  The Number of sampling points per cache frame.

- Channel numbers supported

  The mono recording was set to 1.

【Example】

The following code implements initializing the AI device, acquiring audio frames, and reverse initializing the AI device.

```
1.  /* Step 1: set public attribute of AI device. */
2.  int devID = 1;
3.  IMPAudioIOAttr attr;
4.  attr.samplerate = AUDIO_SAMPLE_RATE_8000;
5.  attr.bitwidth = AUDIO_BIT_WIDTH_16;
6.  attr.soundmode = AUDIO_SOUND_MODE_MONO;
7.  attr.frmNum = 20;
8.  attr.numPerFrm = 400;
9.  attr.chnCnt = 1;
10. ret = IMP_AI_SetPubAttr(devID, &attr);
11. if(ret != 0) {
```

```
12.     IMP_LOG_ERR(TAG, "Set Audio in %d attr err: %d\n", devID, ret);
13.     return set;
14. }
```

2.3.2 IMP_AI_GetPubAttr

【Function】

Gets the AI device properties.

【Grammar】

int IMP_AI_GetPubAttr(int audioDevId, IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
attr	Audio Input device property pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.3 IMP_AI_Enable

【Function】

Enable AI devices.

【Grammar】

int IMP_AI_Enable(int audioDevId);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This function must be called before IMP_AI_SetPubAttr.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.4 IMP_AI_Disable

【Function】

Disable the AI device.

【Grammar】

int IMP_AI_Disable(int audioDevId);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This API works with IMP_AI_Enable. IMP_AI_Disable must be executed before system sleep.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.5 IMP_AI_EnableChn

【Function】

Enable the AI channel.

【Grammar】

int IMP_AI_EnableChn(int audioDevId, int aiChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

IMP_AI_EnableDevice must be called to enable AI channels.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.6 IMP_AI_DisableChn

【Function】

The AI channel is disabled.

【Grammar】

int IMP_AI_DisableChn(int audioDevId, int aiChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Use it together with the IMP_AI_EnableChn.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.7 IMP_AI_PollingFrame

【Function】

Polling AI channel audio flow cache.

【Grammar】

int IMP_AI_PollingFrame(int audioDevId, int aiChn, unsigned int timeout_ms);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
timeout_ms	The Polling timeout time	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Use this interface before using IMP_AI_GetFrame. When the interface is called and the audio data is ready, it can be obtained using IMP_AI_GetFrame.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.8 IMP_AI_GetFrame

【Function】

Get the AI channel audio frames.

【Grammar】

int IMP_AI_GetFrame(int audioDevId, int aiChn, IMPAudioFrame *frm, IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
frm	Audio frame frame pointer	Output
block	Blocking and non-blocking identification	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Release the audio frame and call IMP_AI_ReleaseFrm.

【Example】

```
15. IMPAudioFrame frm;
16.   // Get audio frames
17.   ret = IMP_AI_GetFrame(devID, chnID, &frm, BLOCK);
18.   if(ret != 0) {
19.       IMP_LOG_ERR(TAG, "Audio Get Frame Data error\n");
20.       return ret;
21.   }
```

```
22.   fwrite(frm.virAddr, 1, frm.len, record_file); //Use the audio-frame data
23.  // Release the audio frame
24.   ret = IMP_AI_ReleaseFrame(devID, chnID, &frm);
25.   if(ret != 0) {
26.       IMP_LOG_ERR(TAG, "Audio release frame data error\n");
27.       return ret;
28.   }
```

2.3.9 IMP_AI_ReleaseFrame

【Function】

Release the AI channel audio frames.

【Grammar】

int IMP_AI_ReleaseFrame(int audioDevId, int aiChn, IMPAudioFrame *frm);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
frm	Audio frame frame pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Use it together with the IMP_AI_GetFrame.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.10 IMP_AI_SetChnParam

【Function】

Set up the AI channel parameters's.

【Grammar】

int IMP_AI_SetChnParam(int audioDevId, int aiChn, IMPAudioIChnParam *chnParam);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
chnParam	AI channel parameters	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Call it before the IMP_AI_EnableChn.

【Example】

```
29. int chnID = 0;
30.   IMPAudioIChnParam chnParam;
31.   chnParam.usrFrmDepth = 20;     // the range of valid value is [2,
    MAX_AUDIO_FRAME_NUM].
32.   ret = IMP_AI_SetChnParam(devID, chnID, &chnParam);
33.   if(ret != 0) {
```

```
34.       IMP_LOG_ERR(TAG, "set ai %d channel %d attr err: %d\n", devID, chnID,
    ret);
35.      return ret;
36.  }
```

2.3.11 IMP_AI_GetChnParam

【Function】

Get the AI channel parameters.

【Grammar】

```
int IMP_AI_GetChnParam(int audioDevId, int aiChn,
IMPAudioIChnParam *chnParam);
```

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
chnParam	AI channel parameters	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.12 IMP_AI_EnableAec

【Function】

Enable echo cancellation for AI channels and AO channels Function.

【Grammar】

int IMP_AI_EnableAec(int aiDevId, int aiChn, int aoDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input device number requiring echo offset	Input
aiChn	Audio input channel number requiring echo offset	Input
aoDevId	Audio Output device number requiring echo offset	Input
aoChn	Audio Output channel number requiring echo offset	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Echo elimination for different devices, different packaging, echo elimination will have different effects.

【Example】

Please refer to the example section of sample-Ai-AEC.

2.3.13 IMP_AI_DisableAec

【Function】

Disable the AI Channel Echo offset Function.

【Grammar】

int IMP_AI_DisableAec(int aiDevId, int aiChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Used with the IMP_AI_EnableAec.

【Example】

Please refer to the example section of sample-Ai-AEC.

2.3.14 IMP_AI_EnableNs

【Function】

Enable Noise suppression of AI channels Function.

【Grammar】

int IMP_AI_EnableNs(IMPAudioIOAttr *attr, int mode);

【Formal parameter】

Parameter name	Description	Input / Output
attr	Audio properties that require noise suppression	Input
mode	Level of noise suppression of 0 ~ 3	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

The mode parameter of noise suppression represents the level of noise suppression, ranging from [0 ~ 3], and the higher the level, the cleaner the noise suppression. However, the cleaner the noise suppression, the more sound details are lost, so there is a point of contradiction that is weighed when used.

Echo cancellation includes noise suppression Function, and if echo cancellation is enabled, noise suppression is unnecessary.

【Example】

Please refer to the example section of sample-Audio.

2.3.15 IMP_AI_DisableNs

【Function】

Disable the AI channel noise suppression Function.

【Grammar】

int IMP_AI_DisableNs(void);

【Formal parameter】

None

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Used with the IMP_AI_EnableNs.

【Example】

Please refer to the example section of sample-Audio.

2.3.16 IMP_AI_EnableAgc

【Function】

Enable automatic gain for audio Input Function.

【Grammar】

int IMP_AI_EnableAgc(IMPAudioIOAttr *attr, IMPAudioAgcConfig agcConfig);

【Formal parameter】

Parameter name	Description	Input / Output
attr	Audio properties that require automatic gain control	Input
agcConfig	Parameter configuration of automatic gain,	Input

	configuration magnification	

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

It should be noted that AGC can amplify the gain of sound, but if the parameters of gain are not appropriate, it will lead to broken sound, please adjust yourself when the specific use.

Echo cancellation includes AGC Function, if echo cancellation is enabled.

【Example】

Please refer to the example section of sample-Audio.

2.3.17 IMP_AI_DisableAgc

【Function】

Disable the AI channel automatic gain Function.

【Grammar】

int IMP_AI_DisableAgc(void);

【Formal parameter】

None

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Used with the IMP_AI_EnableAgc.

【Example】

Please refer to the example section of sample-Audio.

2.3.18 IMP_AI_EnableHpf

【Function】

Enable high-pass filtering for AI channel.

【Grammar】

int IMP_AI_EnableHpf(IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
attr	Need to enable high-pass filtering for audio attributes	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Echo cancellation includes HPF Function, and if echo cancellation is enabled, HPF is unnecessary.

【Example】

Please refer to the example section of sample-Audio.

2.3.19 IMP_AI_SetHpfCoFrequency

【Function】

Set the cutoff frequency of the high-pass filter for the AI channel.

【Grammar】

int IMP_AI_SetHpfCoFrequency(int cofrequency);

【Formal parameter】

Parameter name	Description	Input / Output
cofrequency	cut-off frequency	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

This interface needs to be called before the IMP_AI_EnableHpf.

【Example】

Please refer to the example section of sample-Audio.

2.3.20 IMP_AI_DisableHpf

【Function】

Disable the AI-channel high-pass filtering Function.

【Grammar】

int IMP_AI_DisableHpf(void);

【Formal parameter】

None

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

This interface is used with IMP_AI_EnableHpf.

【Example】

Please refer to the example section of sample-Audio.

2.3.21 IMP_AI_SetVol

【Function】

Set the volume of the AI channel.

【Grammar】

int IMP_AI_SetVol(int audioDevId, int aiChn, int aiVol);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiVol	Audio Input Volume Size	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The volume value ranges from -30 to 120. -30 indicates mute, and 120 indicates 30dB amplification. The step length is 0.5dB. Where 60 is a critical point of volume setting, the software does not increase or decrease the volume on this value, when the volume value is less than 60, with every drop 1, the volume decreases 0.5dB; When the volume value is greater than 60, the volume increases by 0.5dB for each increment of 1.

If the aiVol of Input exceeds the range of [-30 , 120], less than-30 will take-30 and more than 120 will take 120.

【Example】

```
37. int volume = 60;
38.  ret = IMP_AI_SetVol(devID, chnID, volume);
39.   if(ret != 0) {
40.       IMP_LOG_ERR(TAG, "Audio Record set volume failed\n");
41.       return ret;
42.   }
```

2.3.22 IMP_AI_GetVol

【Function】

Get the volume of the AI channel.

【Grammar】

int IMP_AI_GetVol(int audioDevId, int aiChn, int *vol);

【Formal parameter】

Parameter	Description	Input /

name		Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiVol	Audio Input Volume Size	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.23 IMP_AI_SetVolMute

【Function】

Mute the AI channel.

【Grammar】

int IMP_AI_SetVolMute(int audioDevId, int aiChn, int mute);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
mute	Audio Input mute flag, mute = 0: off mute, mute = 1: on mute.	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.3.24 IMP_AI_SetGain

【Function】

Set up the gain of the AI channel.

【Grammar】

int IMP_AI_SetGain(int audioDevId, int aiChn, int aiGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiGain	Audio channel Input gain, range [0 ~ 31], corresponding to [-18dB ~ 28.5dB], step 1.5dB.	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

If two channels are used, the gain of both channels is the same. AiGain ranges from 0 to 31. If the

Input value is less than 0, aiGain is set to 0. If the value is greater than 31,aiGain is set to 10.

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.25 IMP_AI_GetGain

【Function】

Gain gain for the AI channel.

【Grammar】

int IMP_AI_GetGain(int audioDevId, int aiChn, int *aiGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiGain	Audio channel Input gain	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AI_SetPubAttr.

2.3.26 IMP_AI_SetAlcGain

【Function】

Set the Input gain for the AI channel.

【Grammar】

int IMP_AI_SetAlcGain(int audioDevId, int aiChn, int aiPgaGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiPgaGain	Audio Input gain, range [0 ~ 7], corresponding to [-13.5dB, +28.5dB], step size 6dB.	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

AiPgaGain ranges from 0 to 7. If the Input value is less than 0, the value of aiPgaGain will be set to 0. If the value is greater than 7, aiGain is set to 7.

【Example】

None.

2.3.27 IMP_AI_GetAlcGain

【Function】

Obtain the AIPga gain value.

【Grammar】

int IMP_AI_getAlcGain(int audioDevId,int aiChn,int *aiPgaGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio Input Device Number	Input
aiChn	Audio input channel number	Input
aiPgaGain	Audio Input gain	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.3.28 IMP_AI_GetFrameAndRef

【Function】

Get the audio frames and the Output reference frames for the AI channel.

【Grammar】

```
int IMP_AI_GetFrameAndRef(int audioDevId, int aiChn, IMPAudioFrame *frm, IMPAudioFrame *ref, IMPBlock block);
```

【Formal parameter】

Parameter name	Description	Input / Output

audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
frm	Audio frame frame pointer	Output
ref	Reference frame frame pointer	Output
block	Blocking / non-blocking ID	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

The following code implements the audio frames and the Output reference frames for obtaining the AI channel.

```
//.....
int ret = -1;
ret = IMP_AI_EnableAecRefFrame(devID, chnID, 0, 0);
if(ret != 0) {
      IMP_LOG_ERR(TAG, "Audio Record enable channel failed\n");
      return NULL;
}

IMPAudioFrame frm;
IMPAudioFrame ref;
// Get the audio frames and the Output reference frames
ret = IMP_AI_GetFrameAndRef(devID, chnID, &frm, &ref, BLOCK);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "Audio Get Frame Data error\n");
    return ret;
}

fwrite(frm.virAddr, 1, frm.len, record_file); // Use the audio-frame data
```

```
19. fwrite(ref.virAddr, 1, ref.len, ref_file); // Use the audio reference frame
20.
21. // Release the audio frame
22. ret = IMP_AI_ReleaseFrame(devID, chnID, &frm);
23. if(ret != 0) {
24.     IMP_LOG_ERR(TAG, "Audio release frame data error\n");
25.     return ret;
26. }
```

2.3.29 IMP_AI_EnableAecRefFrame

【Function】

Open the fetch reference frame.

【Grammar】

int IMP_AI_EnableAecRefFrame(int audioDevId, int aiChn, int audioAoDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
aoDevId	Audio Output Device Number	Input
aoChn	Audio Output Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AI_GetFrameAndRef.

2.3.30 IMP_AI_DisableAecRefFrame

【Function】

Turn off the fetch reference frame.

【Grammar】

int IMP_AI_DisableAecRefFrame(int audioDevId, int aiChn, int audioAoDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio equipment number	Input
aiChn	Audio input channel number	Input
aoDevId	Audio Output Device Number	Input
aoChn	Audio Output Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface is in conjunction with IMP_AI_EnableAecRefFrame.

【Example】

Please refer to the example section of IMP_AI_GetFrameAndRef.

2.4 AO Module API

API name	Function
IMP_AO_SetPubAttr	Set the AO device properties
IMP_AO_GetPubAttr	Get the AO device properties
IMP_AO_Enable	Enable AO devices
IMP_AO_Disable	Disable the AO device
IMP_AO_EnableChn	Enable the AO channel
IMP_AO_DisableChn	Disable the AO channel
IMP_AO_SendFrame	Send the audio frames to the AO channel
IMP_AO_PauseChn	Pause AO channel
IMP_AO_ResumeChn	Restore the AO channel
IMP_AO_ClearChnBuf	Clear the audio data cache in the AO channel
IMP_AO_QueryChnStat	Query the AO channel audio cache status
IMP_AO_SetVol	Set the AO channel volume
IMP_AO_GetVol	Get the AO channel volume
IMP_AO_SetGain	Set up the AO channel gain
IMP_AO_GetGain	Get the AO channel gain
IMP_AO_Soft_Mute	Set the AO channel software to mute
IMP_AO_Soft_UNMute	Unmute the AO channel software
IMP_AO_CacheSwitch	Turn off the AO channel audio caching mechanism
IMP_AO_FlushChnBuf	Wait for the AO channel audio data to play out
IMP_AO_EnableAgc	Enable the AO channel audio automatic gain Function
IMP_AO_DisableAgc	Disable the AO Channel Audio Autogain Function
IMP_AO_EnableHpf	Enable AO channel high-pass filtering Function
IMP_AO_SetHpfCoFrequency	Set the cutoff frequency of the AO-channel

	high-pass filter
IMP_AO_DisableHpf	Disable the AO-channel high-pass filtering Function

Table 3-2 AO API Function

2.4.1 IMP_AO_SetPubAttr

【Function】

Set the AO device properties.

【Grammar】

int IMP_AO_SetPubAttr(int audioDevId, IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
attr	Audio AO Output device property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Audio Output device properties include audio sampling rate, sampling accuracy, audio channel mode, codec encoded channel configuration, number of cache frames, number of sampling points per frame, and Channel numbers supported.

- sampling rate

The T31 built-in codec supports a playback sampling rate of 8K / 16K / 24K / 32K / 44.1K / 48K / 96K.

- Sampling accuracy

  T31 currently only supports 16bit sampling precision sound playback.

- Audio channel mode

  The T31 supports only mono.

- Codec Coding channels

  The T31 supports only mono. Set this parameter to 1.

- The number of cache frames

  Number of cached frames for internal requests.

- Number of sampling points per frame

  Number of sampling points per cache frame.

- Channel numbers supported

  The default is set to 1.

【Example】

The following code implements initializing the AO device, sending audio frames, and reverse initializing the AO device.

```
1.  unsigned char *buf = NULL;
2.  int size = 0;
3.  int ret = -1;
4.
5.  buf = (unsigned char *)malloc(640);
6.  if (buf == NULL) {
7.      IMP_LOG_ERR(TAG, "[ERROR] %s: malloc audio buf error\n", __fu
    nc__);
8.      return NULL;
9.  }
10.
11. FILE *play_file = fopen("play.pcm", "rb");
12. if (play_file == NULL) {
13.     IMP_LOG_ERR(TAG, "[ERROR] %s: fopen %s failed\n", __func__, A
    O_BASIC_TEST_PLAY_FILE);
14.     return NULL;
15. }
16.
17. /* Step 1: set public attribute of AO device. */
```

```
int devID = 0;
IMPAudioIOAttr attr;
attr.samplerate = AUDIO_SAMPLE_RATE_16000;
attr.bitwidth = AUDIO_BIT_WIDTH_16;
attr.soundmode = AUDIO_SOUND_MODE_MONO;
attr.frmNum = 20;
attr.numPerFrm = 640;
attr.chnCnt = 1;
ret = IMP_AO_SetPubAttr(devID, &attr);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "set ao %d attr err: %d\n", devID, ret);
    return NULL;
}

memset(&attr, 0x0, sizeof(attr));
ret = IMP_AO_GetPubAttr(devID, &attr);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "get ao %d attr err: %d\n", devID, ret);
    return NULL;
}

/* Step 2: enable AO device. */
ret = IMP_AO_Enable(devID);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "enable ao %d err\n", devID);
    return NULL;
}

/* Step 3: enable AI channel. */
int chnID = 0;
ret = IMP_AO_EnableChn(devID, chnID);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "Audio play enable channel failed\n");
    return NULL;
}

/* Step 4: Set audio channel volume. */
int chnVol = 80;
ret = IMP_AO_SetVol(devID, chnID, chnVol);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "Audio Play set volume failed\n");
```

```
59.     return NULL;
60. }
61.
62. ret = IMP_AO_GetVol(devID, chnID, &chnVol);
63. if (ret != 0) {
64.     IMP_LOG_ERR(TAG, "Audio Play get volume failed\n");
65.     return NULL;
66. }
67. IMP_LOG_INFO(TAG, "Audio Out GetVol    vol:%d\n", chnVol);
68.
69. int aogain = 28;
70. ret = IMP_AO_SetGain(devID, chnID, aogain);
71. if (ret != 0) {
72.     IMP_LOG_ERR(TAG, "Audio Record Set Gain failed\n");
73.     return NULL;
74. }
75.
76. ret = IMP_AO_GetGain(devID, chnID, &aogain);
77. if (ret != 0) {
78.     IMP_LOG_ERR(TAG, "Audio Record Get Gain failed\n");
79.     return NULL;
80. }
81. IMP_LOG_INFO(TAG, "Audio Out GetGain    gain : %d\n", aogain);
82.
83.   int i = 0;
84.   while (1) {
85.     size = fread(buf, 1, AO_TEST_BUF_SIZE, play_file);
86.     if (size < AO_TEST_BUF_SIZE)
87.         break;
88.
89.     /* Step 5: send frame data. */
90.     IMPAudioFrame frm;
91.     frm.virAddr[0] = (uint32_t *)buf;
92.     frm.virAddr[1] = NULL;
93.     frm.len = size;
94.     ret = IMP_AO_SendFrame(devID, chnID, &frm, BLOCK);
95.     if (ret != 0) {
96.         IMP_LOG_ERR(TAG, "send Frame Data error\n");
97.         return NULL;
98.     }
99.    }
```

```
ret = IMP_AO_FlushChnBuf(devID, chnID);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "IMP_AO_FlushChnBuf error\n");
    return NULL;
}
/* Step 6: disable the audio channel. */
ret = IMP_AO_DisableChn(devID, chnID);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "Audio channel disable error\n");
    return NULL;
}

/* Step 7: disable the audio devices. */
ret = IMP_AO_Disable(devID);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "Audio device disable error\n");
    return NULL;
}
fclose(play_file);
  free(buf);
```

2.4.2 IMP_AO_GetPubAttr

【Function】

Gets the AO device properties.

【Grammar】

int IMP_AO_GetPubAttr(int audioDevId, IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
attr	Audio AO Output device property pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.3 IMP_AO_Enable

【Function】

Enable AO devices.

【Grammar】

int IMP_AO_Enable(int audioDevId);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.4 IMP_AO_Disable

【Function】

Disable the AO device.

【Grammar】

int IMP_AO_Disable(int audioDevId);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
attr	Audio AO Output device property pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface works with the IMP_AO_Enable.

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.5 IMP_AO_EnableChn

【Function】

Enable the AO channel.

【Grammar】

int IMP_AO_EnableChn(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.6 IMP_AO_DisableChn

【Function】

Disable the AO channel.

【Grammar】

int IMP_AO_DisableChn(int audioDevId, int aoChn)

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface works with the IMP_AO_EnableChn.

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.7 IMP_AO_SendFrame

【Function】

Send the audio frames to the AO channel.

【Grammar】

```
int IMP_AO_SendFrame(int audioDevId, int aoChn, IMPAudioFrame
*data, IMPBlock block);
```

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
data	Audio frame frame pointer	Input
block	Blocking / non-blocking ID	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

```
while(1) {
        size = fread(buf, 1, IMP_AUDIO_BUF_SIZE, play_file);
    if(size < IMP_AUDIO_BUF_SIZE)
            break;

        IMPAudioFrame frm;
        frm.virAddr = (uint32_t *)buf;
        frm.len = size;
        ret = IMP_AO_SendFrame(devID, chnID, &frm, BLOCK);
        if(ret != 0) {
            IMP_LOG_ERR(TAG, "send Frame Data error\n");
            return ret;
        }
}
```

2.4.8 IMP_AO_PauseChn

【Function】

Pause AO channel.

【Grammar】

int IMP_AO_PauseChn(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.9 IMP_AO_ResumeChn

【Function】

Restore the AO channel.

【Grammar】

int IMP_AO_ResumeChn(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.10 IMP_AO_ClearChnBuf

【Function】

Clear the audio data cache in the AO channel.

【Grammar】

int IMP_AO_ClearChnBuf(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.11 IMP_AO_QueryChnStat

【Function】

Query the AO channel audio cache status.

【Grammar】

int IMP_AO_QueryChnStat(int audioDevId, int aoChn, IMPAudioOChnState *status);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input

aoChn	Audio AO Device Channel Number	Input
status	Cache the state structure body pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.12 IMP_AO_SetVol

【Function】

Set the AO channel volume.

【Grammar】

int IMP_AO_SetVol(int audioDevId, int aoChn, int aoVol);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
aoVol	Audio Output volume	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The volume value ranges from -30 to 120. -30 indicates mute, and 120 indicates 30dB amplification. The step is 0.5 dB. Where 60 is a critical point of volume setting, the software does not increase or decrease the volume on this value, when the volume value is less than 60, with every drop 1, the volume decreases 0.5dB; When the volume value is greater than 60, the volume increases by 0.5dB for every increment of 1. If the Input aoVol exceeds the range [-30 ~ 120], the value less than -30 will be set to -30, and the value greater than 120 will be set to 120.

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.13 IMP_AO_GetVol

【Function】

Get the AO channel volume.

【Grammar】

int IMP_AO_GetVol(int audioDevId, int aoChn, int *vol);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
vol	Audio Output volume pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.14 IMP_AO_SetGain

【Function】

Set up the AO channel gain.

【Grammar】

int IMP_AO_SetGain(int audioDevId, int aoChn, int aoGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
aoGain	Audio AO Output Analogue gain	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

AoGain ranges from 0 to 31. If the Input value is less than 0, the aoGain value will be set to 0. If the value is greater than 31, aoGain is set to 31.

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.15 IMP_AO_GetGain

【Function】

Get the AO channel gain.

【Grammar】

int IMP_AO_GetGain(int audioDevId, int aoChn, int *aoGain);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
aoGain	Audio AO Output Analogue gain guide	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.16 IMP_AO_Soft_Mute

【Function】

Set the AO channel software to mute.

【Grammar】

int IMP_AO_Soft_Mute(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Call the interface does not silent immediately, but slowly lowers the volume from normal playback until it is really silent.

【Example】

None.

2.4.17 IMP_AO_Soft_UNMute

【Function】

Unmute the AO channel software.

【Grammar】

int IMP_AO_Soft_UNMute(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input

aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.18 IMP_AO_CacheSwitch

【Function】

Turn off the AO channel audio caching mechanism.

【Grammar】

int IMP_AO_CacheSwitch(int audioDevId, int aoChn, int cache_en);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input
cache_en	Cache mechanism switch	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

None.

2.4.19 IMP_AO_FlushChnBuf

【Function】

Wait for the AO channel audio data to play out.

【Grammar】

int IMP_AO_FlushChnBuf(int audioDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
audioDevId	Audio AO device number	Input
aoChn	Audio AO Device Channel Number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_AO_SetPubAttr.

2.4.20 IMP_AO_EnableAgc

【Function】

Enable the AO channel audio automatic gain Function.

【Grammar】

```
int IMP_AO_EnableAgc(IMPAudioIOAttr *attr, IMPAudioAgcConfig agcConfig);
```

【Formal parameter】

Parameter name	Description	Input / Output
attr	Audio attributes that need to automatically gain	Input
agcConfig	Parameter configuration of automatic gain, configuration magnification	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Pay attention to the agcConfig configuration. AGC magnification depends on this parameter. For details about the gain, see IMPAudioAgcConfig. AGC can amplify the gain of sound. However, if the gain parameters are not appropriate, the sound will be broken. You can adjust the gain parameters for specific use. Echo cancellation includes THE AGC function. If echo cancellation is enabled, the automatic gain is not required.

【Example】

None.

2.4.21 IMP_AO_DisableAgc

【Function】

Disable the AO Channel Audio Automatic Gain Control Function.

【Grammar】

int IMP_AO_DisableAgc(void);

【Formal parameter】

None

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

This interface is used with IMP_AO_DisableAgc.

【Example】

None.

2.4.22 IMP_AO_EnableHpf

【Function】

Enable AO channel high-pass filtering Function.

【Grammar】

int IMP_AO_EnableHpf(IMPAudioIOAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
attr	Audio properties requiring high-pass filtering	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Echo cancellation includes HPF Function, and if echo cancellation is enabled, HPF is unnecessary.

【Example】

None.

2.4.23 IMP_AO_SetHpfCoFrequency

【Function】

Set the cutoff frequency of the AO-channel high-pass filter.

【Grammar】

int IMP_AI_SetHpfCoFrequency(int cofrequency);

【Formal parameter】

Parameter name	Description	Input / Output
cofrequency	Cutoff frequency of the high-pass filtering	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

Set the cutoff frequency before opening the high-pass filter.

【Example】

None.

2.4.24 IMP_AO_DisableHpf

【Function】

Disable the AO channel high-pass filtering Function.

【Grammar】

int IMP_AO_DisableHpf(void);

【Formal parameter】

None

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

This interface is used with IMP_AO_EnableHpf.

【Example】

None.

2.5 DMIC Module API

API Name	Function
IMP_DMIC_SetPubAttr	Sets the DMIC channel Input device properties.
IMP_DMIC_GetPubAttr	Gets the DMIC channel Input device properties.

IMP_DMIC_Enable	Enable the DMIC devices.
IMP_DMIC_Disable	Disable the DMIC device.
IMP_DMIC_EnableChn	Enable the DMIC channel.
IMP_DMIC_DisableChn	Disable the DMIC channel.
IMP_DMIC_SetChnParam	Set the DMIC channel parameters.
IMP_DMIC_GetChnParam	Gets the DMIC channel parameters.
IMP_DMIC_PollingFrame	Polling DMIC Channel audio flow cache.
IMP_DMIC_GetFrame	Get the DMIC channel audio frames.
IMP_DMIC_ReleaseFrame	Release the DMIC channel audio frame.
IMP_DMIC_SetUserInfo	Set up the DMIC channel user requirements related information.
IMP_DMIC_EnableAecRefFrame	Enable the DMIC channel to get the reference frames [Function].
IMP_DMIC_GetFrameAndRef	Get the audio frames and reference frames for the DMIC channel.
IMP_DMIC_EnableAec	Enable echo cancellation Function for DMIC channels.
IMP_DMIC_DisableAec	Disable Echo Elimination Function for the DMIC channel.
IMP_DMIC_SetVol	Set the volume of the DMIC channel.
IMP_DMIC_GetVol	Get the volume of the DMIC channel.
IMP_DMIC_SetGain	Set the Input gain for the DMIC channel.
IMP_DMIC_GetGain	Get the Input gain for the DMIC channel.

Table 3-3 DMIC API Function

2.5.1 IMP_DMIC_SetPubAttr

【Function】

Sets the DMIC channel Input device properties.

【Grammar】

```
int IMP_DMIC_SetPubAttr(int dmicDevId, IMPDmicAttr *attr);
```

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital Mac Array Equipment No	Input
attr	Digital mic array audio Input device property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Digital microphone array audio Input device properties include the sampling rate, sampling accuracy, audio channel mode, the number of cache frames, the number of sampling points per frame, and the maximum Channel numbers supported.

- sampling rate

  The T31 Digital Mac Array Audio Input only supports 8K and 16K.

- Sampling accuracy

  T31 Mac array Input, supporting only 16bit sampling accuracy.

- Audio channel mode

  The default is configured as a mono channel.

- The number of cache frames

  Number of cached frames for internal requests.

- Number of sampling points per frame

  The number of sampling points that each cache frame can be cached.

- Maximum Channel numbers

  The maximum value is 4 based on the number of digital MIC arrays.

【Example】

The following code initializes the DMIC device, retrieves the audio frame, and de-initializes the DMIC device.

```
1.  /* Step 1: set dmic user info:if need aec function*/
2.  ret = IMP_DMIC_SetUserInfo(0, 1, 0);    /*No AEC Function
    required*/
3.  if (ret != 0) {
4.      IMP_LOG_ERR(TAG, "dmic set user info error.\n");
5.      return NULL;
6.  }
7.
8.  /* step 2: set dmic audio attr*/
9.  IMPDmicAttr attr;
10. attr.samplerate = DMIC_SAMPLE_RATE_16000;
11. attr.bitwidth = DMIC_BIT_WIDTH_16;
12. attr.soundmode = DMIC_SOUND_MODE_MONO;
13. attr.chnCnt = 4;  //chnCnt=1(1 dmic),2(2 dmic),4(4 dmic)
14. attr.frmNum = 40;
15. attr.numPerFrm = 640;
16.
17. ret = IMP_DMIC_SetPubAttr(0, &attr);
18. if(ret != 0) {
19.     IMP_LOG_ERR(TAG, "DMIC_SetPubAttr failed.\n");
20.     return NULL;
21. }
22.
23. /*step 3: enable DMIC device*/
24. ret = IMP_DMIC_Enable(0);
25. if(ret != 0) {
26.     IMP_LOG_ERR(TAG, "DMIC Enable failed.\n");
27.     return NULL;
```

```
}

/*step 4: set dmic channel attr*/
IMPDmicChnParam chnParam;
chnParam.usrFrmDepth = 40;
ret = IMP_DMIC_SetChnParam(0, 0, &chnParam);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC SetChnParam failed.\n");
    return NULL;
}

/*step 5: enable dmic channel*/
ret = IMP_DMIC_EnableChn(0, 0);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC Enable Channel failed.\n");
    return NULL;
}

/*step 6: set dmic volume*/
ret = IMP_DMIC_SetVol(0, 0, 60);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC Set vol failed.\n");
    return NULL;
}

/*step 7: set dmic gain*/
ret = IMP_DMIC_SetGain(0, 0, 22);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC Set Gain failed.\n");
    return NULL;
}

short *pdata = NULL;
int k = 0;
int record_cnt = 0;

while(1){
    ret = IMP_DMIC_PollingFrame(0, 0, 1000);
    if (ret != 0) {
        IMP_LOG_ERR(TAG, "dmic polling frame data error.\n");
    }
```

```
    ret = IMP_DMIC_GetFrame(0, 0, &g_chnFrm, BLOCK);
    if(ret < 0) {
        printf("IMP_DMIC_GetFrame failed\n");
        break;
    }
    pdata = (short*)(g_chnFrm.rawFrame.virAddr);

   //You can save stream here.

    ret = IMP_DMIC_ReleaseFrame(0, 0, &g_chnFrm) ;
    if (ret < 0) {
        printf("IMP_DMIC_ReleaseFrame failed.\n");
        break;
    }
    if(++record_cnt > DMIC_RECORD_CNT) break;
}

ret = IMP_DMIC_DisableChn(0, 0);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC DisableChn error.\n");
    return NULL;
}

ret = IMP_DMIC_Disable(0);
if (ret != 0){
    IMP_LOG_ERR(TAG, "DMIC Disable error.\n");
    return NULL;
}
```

2.5.2 IMP_DMIC_GetPubAttr

【Function】

Get the DMIC channel Input device properties.

【Grammar】

int IMP_DMIC_GetPubAttr(int dmicDevId, IMPDmicAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output

dmicDevId	Digital mic array device number	Input
attr	Digital mic array Audio Input Device property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.3 IMP_DMIC_Enable

【Function】

Enable the DMIC devices.

【Grammar】

int IMP_DMIC_Enable(int dmicDevId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.4 IMP_DMIC_Disable

【Function】

Disable the DMIC device.

【Grammar】

int IMP_DMIC_Disable(int dmicDevId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface is in conjunction with IMP_DMIC_Enable.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.5 IMP_DMIC_EnableChn

【Function】

Enable the DMIC channel.

【Grammar】

int IMP_DMIC_EnableChn(int dmicDevId, int dmicChnId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The Mac array device must be enabled first.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.6 IMP_DMIC_DisableChn

【Function】

The DMIC channel is disabled.

【Grammar】

int IMP_DMIC_DisableChn(int dmicDevId, int dmicChnId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input

dmicChnId	Digital mic array Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface is used with IMP_DMIC_EnableChn.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.7 IMP_DMIC_SetChnParam

【Function】

Set the DMIC channel parameters.

【Grammar】

int IMP_DMIC_SetChnParam(int dmicDevId, int dmicChnId, IMPDmicChnParam *chnParam);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
chnParam	Digital mic array Audio channel parameters	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Call it before the IMP_DMIC_EnableChn .

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.8 IMP_DMIC_GetChnParam

【Function】

Gets the DMIC channel parameters.

【Grammar】

int IMP_DMIC_GetChnParam(int dmicDevId, int dmicChnId, IMPDmicChnParam *chnParam);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
chnParam	Digital mic array Audio channel parameters	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.9 IMP_DMIC_PollingFrame

【Function】

Polling DMIC Channel audio flow cache.

【Grammar】

int IMP_DMIC_PollingFrame(int dmicDevId, int dmicChnId, unsigned int timeout_ms);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
timeout_ms	Polling Timeout time	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

When the interface call is successful and indicates that the audio data is ready, the audio data can be obtained using IMP_DMIC_GetFrame.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.10 IMP_DMIC_GetFrame

【Function】

Get the DMIC channel audio frames.

【Grammar】

int IMP_DMIC_GetFrame(int dmicDevId, int dmicChnId, IMPDmicChnFrame *chnFrm, IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
chnFrm	Digital mic array Audio channel audio frame frame pointer	Output
block	Blocking / non-blocking ID	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The digital microphone array supports 1, 2, and 4 channels. The original data of the digital microphone array is obtained together with the multi-channel data.

For a 2-channel digital microphone , the data arrangement is as follows:

|mic1|mic2|mic1|mic2|mic1|mic2|.....

For a 4-channel digital MIC, the data arrangement is as follows:

|mic1|mic2|mic3|mic4|mic1|mic2|mic3|mic4|.....

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.11 IMP_DMIC_ReleaseFrame

【Function】

Release DMIC channel audio frames.

【Grammar】

int IMP_DMIC_ReleaseFrame(int dmicDevId, int dmicChnId, IMPDmicChnFrame *chnFrm);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
chnFrm	Mic array audio channel audio frame structure pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface needs to be used with IMP_DMIC_GetFrame, and the acquired audio stream cache must be released in time.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.12 IMP_DMIC_SetUserInfo

【Function】

Set up the DMIC channel user requirements related information.

【Grammar】

int IMP_DMIC_SetUserInfo(int dmicDevId, int aecDmicId, int need_aec);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
aecDmicId	The number of the mic for which the user needs to perform echo cancellation	Input
need_aec	Does the user need to do echo cancellation processing (0: not required ; 1: required)	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.13 IMP_DMIC_EnableAecRefFrame

【Function】

Enable the DMIC channels to get the reference frames Function.

【Grammar】

int IMP_DMIC_EnableAecRefFrame(int dmicDevId, int dmicChnId, int

audioAoDevId, int aoChn);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
aoDevId	Audio output device number that needs echo cancellation	Output
aoChn	Audio output channel number that needs echo cancellation	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_GetFrameAndRef.

2.5.14 IMP_DMIC_GetFrameAndRef

【Function】

Get audio frame and reference frame for DMIC channel.

【Grammar】

int IMP_DMIC_GetFrameAndRef(int dmicDevId, int dmicChnId, IMPDmicChnFrame *chnFrm, IMPDmicFrame *ref, IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
chnFrm	Mike array audio channel audio frame frame pointer	Output
ref	Reference frame frame pointer	Output
block	Blocking / non-blocking ID	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

The following code implements the audio frame and the Output reference frame for obtaining the DMIC channel.

```
IMPDmicChnFrame g_chnFrm;
IMPDmicFrame g_refFrm;
......
ret = IMP_DMIC_EnableAecRefFrame(0, 0, 0, 0);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC EnableAecRef failed.\n");
    return NULL;
}

ret = IMP_DMIC_PollingFrame(0, 0, 1000);
if (ret != 0) {
    IMP_LOG_ERR(TAG, "dmic polling frame data error.\n");
}
ret = IMP_DMIC_GetFrameAndRef(0, 0, &g_chnFrm, &g_refFrm, BLOCK);
```

```
if(ret < 0) {
    printf("IMP_DMIC_GetFrame failed.\n");
    break;
}
ret = IMP_DMIC_ReleaseFrame(0, 0, &g_chnFrm) ;
if (ret < 0) {
    printf("IMP_DMIC_ReleaseFrame failed.\n");
    break;
}
ret = IMP_DMIC_DisableAecRefFrame(0, 0, 0, 0);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "DMIC DisableAec error.\n");
    return NULL;
}
......
```

2.5.15 IMP_DMIC_EnableAec

【Function】

Enable echo cancellation in DMIC channel.

【Grammar】

int IMP_DMIC_EnableAec(int dmicDevId, int dmicChnId, int aoDevId, int aoChId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
aoDevId	Audio output device number that needs echo cancellation.	Input
aoChn	Audio output channel number that needs echo cancellation.	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【Example】

Please refer to the example section of echo cancellation .

2.5.16 IMP_DMIC_DisableAec

【Function】

The echo cancellation function of the DMIC channel is disabled.

【Grammar】

int IMP_DMIC_DisableAec(int dmicDevId, int dmicChnId);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so ； libaudioProcess.so

Configuration file:webrtc_profile.ini

【NB】

This interface needs to be used with IMP_DMIC_EnableAec.

【Example】

Please refer to the example section of echo cancellation .

2.5.17 IMP_DMIC_SetVol

【Function】

Set the volume of the DMIC channel.

【Grammar】

int IMP_DMIC_SetVol(int dmicDevId, int dmicChnId, int dmicVol);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
dmicVol	Digital mic array Audio Input Volume Size	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The volume range of the interface ranges from -30 to 120.-30 indicates mute, and 120 indicates 30dB amplification. The step length is 0.5dB. Where 60 is a critical point of volume setting, the software does not increase or decrease the volume on this value, when the volume value is less than 60, with every drop 1, the volume decreases 0.5dB; When the volume value is greater than 60, the

volume increases by 0.5dB for each increment of 1.

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.18 IMP_DMIC_GetVol

【Function】

Get the volume of the DMIC channel.

【Grammar】

int IMP_DMIC_GetVol(int dmicDevId, int dmicChnId, int *dmicVol);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
dmicVol	Digital mic array Audio Input volume pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.19 IMP_DMIC_SetGain

【Function】

Set the Input gain for the DMIC channel.

【Grammar】

int IMP_DMIC_SetGain(int dmicDevId, int dmicChnId, int dmicGain);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input
dmicChnId	Digital mic array Audio input channel number	Input
dmicGain	Digital mic array Audio Input gain	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.5.20 IMP_DMIC_GetGain

【Function】

Get the Input gain of DMIC channel.

【Grammar】

int IMP_DMIC_GetGain(int dmicDevId, int dmicChnId, int *dmicGain);

【Formal parameter】

Parameter name	Description	Input / Output
dmicDevId	Digital mic array device number	Input

dmicChnId	Digital mic array Audio input channel number	Input
dmicGain	Digital mic array Audio Input Gain pointer	Output

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【Example】

Please refer to the example section of IMP_DMIC_SetPubAttr.

2.6 AENC Module API

API Name	Function
IMP_AENC_CreateChn	Create an audio-encoding channel
IMP_AENC_DestroyChn	Destroy the audio-encoding channels
IMP_AENC_SendFrame	Send the audio-encoded audio frames
IMP_AENC_PollingStream	The Polling-encoded audio stream cache
IMP_AENC_GetStream	Get the postencoded code stream
IMP_AENC_ReleaseStream	Release the code stream obtained from the audio encoding channel
IMP_AENC_RegisterEncoder	Register the encoder
IMP_AENC_UnRegisterEncoder	Log off the encoder

2.6.1 IMP_AENC_CreateChn

【Function】

Create an audio-encoding channel.

【Grammar】

int IMP_AENC_CreateChn(int aeChn, IMPAudioEncChnAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
aeChn	Channel number	Input
attr	Audio-encoded channel property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Currently, the SDK supports PT_G711A, PT_G711U, and PT_G726 encoding, so using the encoding in the SDK, only attr.type = PT_G711A is needed.

If a custom encoder is required, register it, and the example code is described in IMP_AENC_RegisterEncoder.

【Example】

The following code implements the operation of creating a coding channel and destroy the coding channels.

```
1.  int AeChn = 0;
2.  IMPAudioEncChnAttr attr;
3.  attr.type = PT_G711A;
4.  attr.bufSize = 20;
5.  ret = IMP_AENC_CreateChn(AeChn, &attr);
6.  if(ret != 0) {
7.      IMP_LOG_ERR(TAG, "Audio encode create channel failed\n");
8.      return ret;
9.  }
```

```
//........
ret = IMP_AENC_DestroyChn(AeChn);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "Audio encode destroy channel failed\n");
    return ret;
}
```

2.6.2 IMP_AENC_DestroyChn

【Function】

Destroy the audio-encoding channels.

【Grammar】

int IMP_AENC_DestroyChn(int aeChn);

【Formal parameter】

Parameter name	Description	Input / Output
aeChn	Channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface is used with IMP_AENC_CreateChn.

【Example】

Please refer to the example section of IMP_AENC_CreateChn.

2.6.3 IMP_AENC_SendFrame

【Function】

Send the audio-encoded audio frames.

【Grammar】

int IMP_AENC_SendFrame(int aeChn, IMPAudioFrame *frm);

【Formal parameter】

Parameter name	Description	Input / Output
aeChn	Channel number	Input
frm	Audio frame frame pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

The following code realizes the operation of reading audio raw data, sending audio raw data encoding, obtaining encoded data, using coded code stream, and releasing coded code stream.

```
1.  while(1) {
2. // Read a frame of data
3. ret = fread(buf_pcm, 1, IMP_AUDIO_BUF_SIZE, file_pcm);
4. if(ret < IMP_AUDIO_BUF_SIZE)
5.     break;
6.
7. // encode
8. IMPAudioFrame frm;
9. frm.virAddr = (uint32_t *)buf_pcm;
```

```
frm.len = ret;
ret = IMP_AENC_SendFrame(AeChn, &frm);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio encode send frame failed\n");
    return ret;
}
// Gets the encoding code stream
IMPAudioStream stream;
ret = IMP_AENC_GetStream(AeChn, &stream, BLOCK);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio encode get stream failed\n");
    return ret;
}

// Use the encoding code stream
fwrite(stream.stream, 1, stream.len, file_g711);

// Release the encoding code flow
ret = IMP_AENC_ReleaseStream(AeChn, &stream);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio encode release stream failed\n");

    return ret;
}
```

2.6.4 IMP_AENC_PollingStream

【Function】

Polling encodes audio stream cache.

【Grammar】

int IMP_AENC_PollingStream(int AeChn, unsigned int timeout_ms);

【Formal parameter】

Parameter name	Description	Input / Output
AeChn	Audio encoding Input Channel number	Input
timeout_ms	Polling timeout	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

The IMP_AENC_GetStream interface is used before the IMP_AENC_GetStream interface is used. When the interface is called successfully, the audio encoding data is ready and the IMP_AENC_GetStream can be used to retrieve the encoded data.

【Example】

Please refer to the example section of IMP_AENC_SendFrame.

2.6.5 IMP_AENC_GetStream

【Function】

Gets the post-encoding code stream.

【Grammar】

int IMP_AENC_GetStream(int aeChn, IMPAudioStream *stream ,IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
aeChn	Channel number	Input
stream	Get audio code flow	Output
block	Non-blocking identification	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_AENC_SendFrame.

2.6.6 IMP_AENC_ReleaseStream

【Function】

Release the code stream obtained from the audio encoding channel.

【Grammar】

int IMP_AENC_ReleaseStream(int aeChn,IMPAudioStream *stream);

【Formal parameter】

Parameter name	Description	Input / Output
aeChn	Channel number	Input
stream	Get the audio code stream pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_AENC_SendFrame.

2.6.7 IMP_AENC_RegisterEncoder

【Function】

Register the encoder.

【Grammar】

int IMP_AENC_RegisterEncoder(int *handle, IMPAudioEncEncoder *encoder);

【Formal parameter】

Parameter name	Description	Input / Output
handle	Register the handle	Output
encoder	Encoder property structure	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

After registration, use the same method as using the SDK encoder.

【Example】

The following code implements the operation of registering the encoder, using the encoder, and creating the encoding channel.

```
1.  int handle_g711a = 0;
2.  IMPAudioEncEncoder my_encoder;
3.  my_encoder.maxFrmLen = 1024;
4.  sprintf(my_encoder.name, "%s", "MY_G711A");
```

```
5.  my_encoder.openEncoder = NULL; // Encoder callback function
6.  my_encoder.encoderFrm = MY_G711A_Encode_Frm; // Encoder callback
    function
7.  my_encoder.closeEncoder = NULL; // Encoder callback function
8.
9.  ret = IMP_AENC_RegisterEncoder(&handle_g711a, &my_encoder);
10.  if(ret != 0) {
11.      IMP_LOG_ERR(TAG, "IMP_AENC_RegisterEncoder failed\n");
12.      return ret;
13.  }
14.
15.  // Use the encoder
16. int AeChn = 0;
17. IMPAudioEncChnAttr attr;
18. attr.type = handle_g711a;
19. // The encoder type is equal to the value of handle_g711a returned
    successfully registered.
20. attr.bufSize = 20;
21. ret = IMP_AENC_CreateChn(AeChn, &attr);
22. if(ret != 0) {
23.      IMP_LOG_ERR(TAG, "imp audio encode create channel failed\n");

24.      return ret;
25.  }
```

2.6.8 IMP_AENC_UnRegisterEncoder

【Function】

Log off the encoder.

【Grammar】

int IMP_AENC_UnRegisterEncoder(int *handle);

【Formal parameter】

Parameter name	Description	Input / Output
handle	Register the handle (the handle obtained when registering the encoder)	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

Used with the IMP_AENC_RegisterEncoder.

【Example】

Please refer to the example section of IMP_AENC_RegisterEncoder.

2.7 ADEC Module API

API Name	Function
IMP_ADEC_CreateChn	Create an audio decoding channel
IMP_ADEC_DestroyChn	Destroy the audio decoding channel
IMP_ADEC_SendStream	Send the audio code streams to the audio decoding channel
IMP_ADEC_PollingStream	Polling Decodes the audio stream cache
IMP_ADEC_GetStream	Get the post-decoding code stream
IMP_ADEC_ReleaseStream	Release the code stream obtained from the audio decoding channel
IMP_ADEC_ClearChnBuf	Clear the current audio data cache in the audio decoding channel
IMP_ADEC_RegisterDecoder	Register the decoder
IMP_ADEC_UnRegisterDecoder	Log off the decoder

2.7.1 IMP_ADEC_CreateChn

【Function】

Create an audio-encoding channel.

【Grammar】

int IMP_ADEC_CreateChn(int adChn, IMPAudioDecChnAttr *attr);

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input
attr	Channel property pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

The following code implements the operation of creating the decoding channels and destroying the decoding channels.

```
1.  int adChn = 0;
2.  IMPAudioDecChnAttr attr;
3.  attr.type = PT_G711A;
4.  attr.bufSize = 20;
5.  attr.mode = ADEC_MODE_PACK;
6.  ret = IMP_ADEC_CreateChn(adChn, &attr);
7.  if(ret != 0) {
8.          IMP_LOG_ERR(TAG, "imp audio decoder create channel failed
    \n");
```

```
9.          return ret;
10. }
11. ///.........
12. ret = IMP_ADEC_DestroyChn(adChn, &attr);
13. if(ret != 0) {
14.         IMP_LOG_ERR(TAG, "imp audio decoder destroy channel faile
    d\n");
15.         return ret;
16. }
```

2.7.2 IMP_ADEC_DestroyChn

【Function】

Destroy the audio decoding channel.

【Grammar】

int IMP_ADEC_DestroyChn(int adChn)

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_ADEC_CreateChn.

2.7.3 IMP_ADEC_SendFrame

【Function】

Sends an audio code stream to the audio decoder channel.

【Grammar】

int IMP_ADEC_SendStream(int adChn, IMPAudioStream *stream, IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input
stream	Audio stream	Input
block	Blocking / non-blocking ID.	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

The following code realizes the operation of obtaining decoded data, sending it to the data to be decoded, obtaining decoded data, using decoded data, and releasing decoded data.

```
1.  while(1) {
2.  // Get the data to be decoded
3.  ret = fread(buf_g711, 1, IMP_AUDIO_BUF_SIZE/2, file_g711);
4.  if(ret < IMP_AUDIO_BUF_SIZE/2)
5.      break;
6.
7.  // Send decoded data
```

```
IMPAudioStream stream_in;
stream_in.stream = (uint8_t *)buf_g711;
stream_in.len = ret;
ret = IMP_ADEC_SendStream(adChn, &stream_in, BLOCK);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio encode send frame failed\n");
    return ret;
}

// Get decoded data
IMPAudioStream stream_out;
ret = IMP_ADEC_GetStream(adChn, &stream_out, BLOCK);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio decoder get stream failed\n");
    return ret;
}

// Use decoded data
fwrite(stream_out.stream, 1, stream_out.len, file_pcm);

// Release the decoded data
ret = IMP_ADEC_ReleaseStream(adChn, &stream_out);
if(ret != 0) {
    IMP_LOG_ERR(TAG, "imp audio decoder release stream failed\n");

    return ret;
}
```

2.7.4 IMP_ADEC_PollingStream

【Function】

Polling decodes the audio stream cache.

【Grammar】

int IMP_ADEC_PollingStream(int AdChn, unsigned int timeout_ms);

【Formal parameter】

Parameter name	Description	Input / Output

AdChn	Audio decoding Input Channel number	Input
timeout_ms	Polling timeout	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

This interface is used before IMP_ADEC_GetStream is used. When this interface is successfully invoked, the audio decoded data is ready and the IMP_ADEC_GetStream can be used to obtain the decoded data.

【Example】

Please refer to the example section of IMP_ADEC_SendFrame.

2.7.5 IMP_ADEC_GetStream

【Function】

Get the decoded code stream.

【Grammar】

int IMP_ADEC_GetStream(int adChn, IMPAudioStream *stream ,IMPBlock block);

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input
stream	Get the decoded code stream	Input
block	Blocking / non-blocking ID	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_ADEC_SendFrame.

2.7.6 IMP_ADEC_ReleaseStream

【Function】

Releases the stream obtained from the audio decoder channel.

【Grammar】

int IMP_ADEC_ReleaseStream(int adChn,IMPAudioStream *stream);

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input
stream	Audio code stream pointer	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_ADEC_SendFrame.

2.7.7 IMP_ADEC_ClearChnBuf

【Function】

Clears the current audio data cache in the audio decoder channel.

【Grammar】

int IMP_ADEC_ClearChnBuf(int adChn);

【Formal parameter】

Parameter name	Description	Input / Output
adChn	Channel number	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

None.

2.7.8 IMP_ADEC_RegisterEncoder

【Function】

Register decoder.

【Grammar】

int IMP_ADEC_RegisterDecoder(int *handle, IMPAudioDecDecoder *decoder);

【Formal parameter】

Parameter name	Description	Input / Output
handle	Handle to register	Output
decoder	Decoder property structure	Input

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

After registration, use the same method as using the SDK's built-in decoder.

【Example】

The following code implements registering the decoder, using the decoder, and creating a decoder channel.

```
1.  int handle_g711a = 0;
2.  IMPAudioDecDecoder my_decoder;
3.  sprintf(my_decoder.name, "%s", "MY_G711A");
4.  my_decoder.openDecoder = NULL; // Decoder callback function
5.  my_decoder.decodeFrm = MY_G711A_Decode_Frm; // Decoder callback
    function
6.  my_decoder.getFrmInfo = NULL; // Decoder callback function
7.  my_decoder.closeDecoder = NULL; // Decoder callback function
8.
9.  // Register decoder
10. ret = IMP_ADEC_RegisterDecoder(&handle_g711a, &my_decoder);
11. if(ret != 0) {
12.         IMP_LOG_ERR(TAG, "IMP_ADEC_RegisterDecoder failed\n");
```

```
13.         return ret;
14. }
15.
16.  // Use decoder
17. int adChn = 0;
18. IMPAudioDecChnAttr attr;
19. attr.type = handle_g711a; // The decoder type is equal to the handle_g711a returned by the decoder registration.
20. attr.bufSize = 20;
21. attr.mode = ADEC_MODE_PACK;
22. // Create decode channel
23. ret = IMP_ADEC_CreateChn(adChn, &attr);
24.  if(ret != 0) {
25.         IMP_LOG_ERR(TAG, "imp audio decoder create channel failed \n");
26.         return ret;
27. }
```

2.7.9 IMP_ADEC_UnRegisterEncoder

【Function】

Log off the decoder.

【Grammar】

int IMP_ADEC_UnRegisterDecoder(int *handle);

【Formal parameter】

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

【Return value】

Return value 0 success; non-zero failure.

【Dependence】

Header file:imp_audio.h

Library file:libimp.a / libimp.so

【NB】

None

【Example】

Please refer to the example section of IMP_ADEC_RegisterEncoder.

2.8 AI/AO Data Type

AI and AO module related data types definition are as follows:

Name	Definition
IMPAudioSampleRate	Audio sampling rate definition
IMPAudioBitWidth	Audio sampling accuracy definition
IMPAudioSoundMode	Audio channel mode definition
IMPAudioIOAttr	Audio Input Output device property structure
IMPAudioFrame	Audio frame structure
IMPBlock	Audio Stream Block type definition
IMPAudioIChnParam	Audio channel parameters structure
IMPAudioAecRef	Echo elimination reference data source definition
IMPAudioAecConfig	Audio echo cancellation structure
IMPAudioAgcConfig	Automatic gain control structure
IMPAudioOChnState	Data cache state structure for the audio Output channel

2.8.1 IMPAudioSampleRate

【Description】

Audio Sampling Rate definition.

【Definition】

```
typedef enum {
    AUDIO_SAMPLE_RATE_8000   = 8000,
    AUDIO_SAMPLE_RATE_16000 = 16000,
    AUDIO_SAMPLE_RATE_24000 = 24000,
    AUDIO_SAMPLE_RATE_32000 = 32000,
    AUDIO_SAMPLE_RATE_44100 = 44100,
    AUDIO_SAMPLE_RATE_48000 = 48000,
    AUDIO_SAMPLE_RATE_96000 = 96000,
} IMPAudioSampleRate;
```

【Members】

Member name	Description
AUDIO_SAMPLE_RATE_8000	8K sampling rate
AUDIO_SAMPLE_RATE_16000	16K sampling rate
AUDIO_SAMPLE_RATE_24000	24K sampling rate
AUDIO_SAMPLE_RATE_32000	32K sampling rate
AUDIO_SAMPLE_RATE_44100	The 44.1K sampling rate
AUDIO_SAMPLE_RATE_48000	48K sampling rate
AUDIO_SAMPLE_RATE_96000	96K sampling rate

【NB】

None

【Related data types and interfaces】

IMPAudioIOAttr

2.8.2 IMPAudioBitWidth

【Description】

Audio sampling accuracy definition.

【Definition】

```
typedef enum {
    AUDIO_BIT_WIDTH_16 = 16,
} IMPAudioBitWidth;
```

【Members】

Member name	Description
AUDIO_BIT_WIDTH_16	16bit sampling accuracy

【NB】

None

【Related data types and interfaces】

IMPAudioIOAttr

IMPAudioFrame

2.8.3 IMPAudioSoundMode

【Description】

Audio Channel mode Definition.

【Definition】

```
typedef enum {
    AUDIO_SOUND_MODE_MONO   = 1,
    AUDIO_SOUND_MODE_STEREO = 2,
} IMPAudioSoundMode;
```

【Members】

Member name	Description
AUDIO_SOUND_MODE_MONO	single track
AUDIO_SOUND_MODE_STEREO	dual track

【NB】

None

【Related data types and interfaces】

IMPAudioIOAttr

IMPAudioFrame

2.8.4 IMPAudioIOAttr

【Description】

Audio Input Output device property structure.

【Definition】

```
typedef struct {
    IMPAudioSampleRate samplerate;
    IMPAudioBitWidth bitwidth;
    IMPAudioSoundMode soundmode;
    int frmNum;
    int numPerFrm;
    int chnCnt;
} IMPAudioIOAttr;
```

【Members】

Member name	Description
samplerate	Audio sampling rate
bitwidth	Audio sampling accuracy

soundmode	Audio channel mode
frmNum	Number of cached frames, range of values: [2,MAX_AUDIO_FRAME_NUM]
numPerFrm	Number of sampling points per frame
chnCnt	Number of channels supported

【NB】

None

【Related data types and interfaces】

IMP_AI_SetPubAttr

IMP_AI_GetPubAttr

IMP_AO_SetPubAttr

IMP_AO_GetPubAttr

2.8.5 IMPAudioFrame

【Description】

Audio frame structure.

【Definition】

```
typedef struct {
    IMPAudioBitWidth bitwidth;
    IMPAudioSoundMode soundmode;
    uint32_t *virAddr;
    uint32_t phyAddr;
    int64_t timeStamp;
    int seq;
    int len;
} IMPAudioFrame;
```

【Members】

Member name	Description
bitwidth	Audio sampling accuracy
soundmode	Audio channel mode
virAddr	Audio Frame Data Virtual address
phyAddr	Audio Frame data physical address
timeStamp	Audio frame data timestamp in us
seq	Audio frame serial number
len	Audio frame length, measured in byte

【NB】

None

【Related data types and interfaces】

IMP_AI_GetFrame

IMP_AI_GetFrameAndRef

IMP_AI_ReleaseFrame

IMP_AO_SendFrame

2.8.6 IMPAudioIChnParam

【Description】

Audio channel parameters structure.

【Definition】

```
typedef struct {
    int usrFrmDepth;
    int Rev;
} IMPAudioIChnParam;
```

【Members】

Member name	Description
usrFrmDepth	Audio Frame cache depth
Rev	Keep parameters

【NB】

None

【Related data types and interfaces】

IMP_AI_SetChnParam

IMP_AI_GetChnParam

2.8.7 IMPAudioAgcConfig

【Description】

Automatic gain control structure.

【Definition】

```
typedef struct {
    int TargetLevelDbfs;
    int CompressionGaindB;
} IMPAudioAgcConfig;
```

【Members】

Member name	Description
TargetLevelDbfs	Gain level, valued at [0,31], which refers to the target volume level
CompressionGaindB	Set the maximum gain value, [0,90], 0 represents no gain, the larger the value, the higher the gain, the closer the amplification is to the target gain level

【NB】

None

【Related data types and interfaces】

IMP_AI_EnableAgc

IMP_AO_EnableAgc

2.8.8 IMPAudioOChnState

【Description】

Data cache state structure for the audio Output channel.

【Definition】

```
typedef struct {
    int chnTotalNum;
    int chnFreeNum;
    int chnBusyNum;
} IMPAudioOChnState;
```

【Members】

Member name	Description
chnTotalNum	Total number of cache blocks for the Output channel
chnFreeNum	Number of free cache blocks
chnBusyNum	The number of cache blocks being occupied

【NB】

None

【Related data types and interfaces】

IMP_AO_QueryChnStat

2.9 DMIC Data Type

The DMIC module related data types definition are as follows:

Name	Definition
IMPDmicSampleRate	Digital Mac Input device sampling rate definition
IMPDmicBitWidth	Digital Mac Input device sampling depth definition
IMPDmicSoundMode	Digital Mike Input Device Sound Channel mode definition
IMPDmicAttr	Digital Mac Input device property structure
IMPDmicChnParam	Digital Mac Input Device Audio Frame Structure
IMPBlock	Audio stream blocking type definition
IMPDmicFrame	Digital Mac Input Device Audio Frame Structure
IMPDmicChnFrame	Digital Mac channel audio frame structure

2.9.1 IMPDmicSampleRate

【Description】

Digital Mac Input device sampling rate definition.

【Definition】

```
typedef enum {
    DMIC_SAMPLE_RATE_8000 = 8000,
    DMIC_SAMPLE_RATE_16000 = 16000,
} IMPDmicSampleRate;
```

【Members】

Member name	Description
DMIC_SAMPLE_RATE_8000	8K sampling rate
DMIC_SAMPLE_RATE_16000	16K sampling rate

【NB】

None

【Related data types and interfaces】

IMPDmicAttr

2.9.2 IMPDmicBitWidth

【Description】

Digital Mac Input Device Sampling Depth definition.

【Definition】

```
typedef enum {
    DMIC_BIT_WIDTH_16 = 16,
} IMPDmicBitWidth;
```

【Members】

Member name	Description
DMIC_BIT_WIDTH_16 = 16	16 bit sampling accuracy

【NB】

None

【Related data types and interfaces】

IMPDmicAttr

2.9.3 IMPDmicSoundMode

【Description】

Digital Mike Input Device Sound Channel Mode definition.

【Definition】

```
typedef enum {
```

```
    DMIC_SOUND_MODE_MONO = 1,
    DMIC_SOUND_MODE_STEREO = 2,
} IMPDmicSoundMode;
```

【Members】

Member name	Description
DMIC_SOUND_MODE_MONO	single track
DMIC_SOUND_MODE_STEREO	stereophonic

【NB】

None

【Related data types and interfaces】

IMPDmicAttr

2.9.4 IMPDmicAttr

【Description】

Digital Mike Input Device Properties.

【Definition】

```
typedef struct {
    IMPDmicSampleRate samplerate;
    IMPDmicBitWidth bitwidth;
    IMPDmicSoundMode soundmode;
    int frmNum;
    int numPerFrm;
    int chnCnt;
} IMPDmicAttr;
```

【Members】

Member name	Description
samplerate	DMIC sampling rate
bitwidth	DMIC sampling accuracy
soundmode	acoustic pattern
frmNum	Number of DMIC recording cache frames
numPerFrm	Number of sampling points per frame
chnCnt	Maximum number of voice channels supported

【NB】

None

【Related data types and interfaces】

IMP_DMIC_SetPubAttr

IMP_DMIC_GetPubAttr

2.9.5 IMPDmicChnParam

【Description】

Digital Mac Input Device Audio Frame Structure.

【Definition】

```
typedef struct {
    int usrFrmDepth;
    int Rev;
} IMPDmicChnParam;
```

【Members】

Member name	Description

usrFrmDepth	DMIC audio frame cache depth
Rev	retention parameter

【NB】

None

【Related data types and interfaces】

IMP_DMIC_SetChnParam

IMP_DMIC_GetChnParam

2.9.6 IMPDmicFrame

【Description】

Digital Mac Input Device Audio Frame Structure.

【Definition】

```
typedef struct {
    IMPDmicBitWidth bitwidth;
    IMPDmicSoundMode soundmode;
    uint32_t *virAddr;
    uint32_t phyAddr;
    int64_t timeStamp;
    int seq;
    int len;
} IMPDmicFrame;
```

【Members】

Member name	Description
bitwidth	DMIC sampling accuracy
soundmode	DMIC sampling rate
virAddr	DMIC Frame Data virtual address

phyAddr	DMIC Frame Data physical address
timeStamp	DMIC frame data timestamp in us
seq	DMIC Frame data serial number
len	DMIC frame data length, measured in byte

【NB】

None

【Related data types and interfaces】

IMP_DMIC_GetFrame

IMP_DMIC_GetFrameAndRef

IMP_DMIC_ReleaseFrame

2.9.7 IMPDmicChnFrame

【Description】

Digital Mac Input Device Audio Frame Structure.

【Definition】

```
typedef struct {
    IMPDmicFrame rawFrame;
    IMPDmicFrame aecFrame;
} IMPDmicChnFrame;
```

【Members】

Member name	Description
rawFrame	DMIC audio frame raw recording data

aecFrame	The DMIC audio frame selects a frame after the echo is eliminated from the data

【NB】

None

【Related data types and interfaces】

IMP_DMIC_GetFrame

IMP_DMIC_GetFrameAndRef

IMP_DMIC_ReleaseFrame

2.10 AENC Data Type

AENC related data types definition are as follows:

Name	Definition
IMPAudioPalyloadType	Audio net charge data type definition
IMPAudioEncChnAttr	Audio-encoded channel property structure
IMPAudioEncEncoder	Encoder property structure

2.10.1 IMPAudioPalyloadType

【Description】

Audio net charge type enumeration definition.

【Definition】

```
typedef enum {
    PT_PCM          = 0,
    PT_G711A  = 1,
    PT_G711U  = 2,
```

```
        PT_G726     = 3,
        PT_AEC          = 4,
        PT_ADPCM= 5,
        PT_MAX          = 6,
    } IMPAudioPalyloadType;
```

【Members】

Member name	Description
PT_PCM	PCM
PT_G711A	G711A
PT_G711U	G711U
PT_G726	G726
PT_AEC	AEC
PT_ADPCM	ADPCM
PT_PT_MAX	keep

【NB】

None

【Related data types and interfaces】

IMPAudioEncChnAttr

IMPAudioDecChnAttr

2.10.2 IMPAudioEncChnAttr

【Description】

Audio-encoded channel property structure.

【Definition】

```
    typedef struct {
        IMPAudioPalyloadType type;
```

```
    int bufSize;
    uint32_t *value;
} IMPAudioEncChnAttr;
```

【Members】

Member name	Description
type	Audio net charge data type
bufSize	Size of buf, in frames, [2~MAX_AUDIO_FRAME_NUM]
value	Protocol property pointer

【NB】

None

【Related data types and interfaces】

IMP_ADEC_CreateChn

2.10.3 IMPAudioEncEncoder

【Description】

Encoder property structure.

【Definition】

```
typedef struct {
    IMPAudioPalyloadType type;
    int maxFrmLen;
    char name[16];
    int (*openEncoder)(void *encoderAttr, void
            *encoder);
    int (*encoderFrm)(void *encoder, IMPAudioFrame
            *data, unsigned char *outbuf,int *outLen);
```

```
        int (*closeEncoder)(void *encoder);
    } IMPAudioEncEncoder;
```

【Members】

Member name	Description
type	Encoding protocol type
maxFrmLen	Maximum code flow length
name	Encoder name
openEncoder	Open the encoder callback interface
encoderFrm	Encode code stream callback interface
closeEncoder	Turn off the encoder callback interface

【NB】

None

【Related data types and interfaces】

IMP_AENC_RegisterEncoder

2.11 ADEC Data type

ADEC related data types definition are as follows：

Name	Definition
IMPAudioDecMode	Decode mode definition
IMPAudioDecChnAttr	Decoding the channel property structure
IMPAudioDecDecoder	Decoder property structure body

2.11.1 IMPAudioDecMode

【Description】

Decode mode definition.

【Definition】

```
typedef enum {
    ADEC_MODE_PACK   = 0,
    ADEC_MODE_STREAM = 1,
} IMPAudioDecMode;
```

【Members】

Member name	Description
ADEC_MODE_PACK	Pack, by way of decoding the way
ADEC_MODE_STREAM	Stream mode for decoding

【NB】

None

【Related data types and interfaces】

IMPAudioDecChnAttr

2.11.2 IMPAudioDecChnAttr

【Description】

Decoding the channel property structure.

【Definition】

```
typedef struct {
    IMPAudioPalyloadType type;
    int bufSize;
    IMPAudioDecMode mode;
    void *value;
```

```
} IMPAudioDecChnAttr;
```

【Members】

Member name	Description
type	Audio decoding protocol type
bufSize	Audio Decode cache size
mode	Decoding mode
value	Specific protocol property pointer

【NB】

None

【Related data types and interfaces】

IMP_ADEC_CreateChn

2.11.3 IMPAudioDecDecoder

【Description】

Decoder property structure body.

【Definition】

```
typedef struct {
    IMPAudioPalyloadType type;
    char name[16];
    int (*openDecoder)(void *decoderAttr, void *decoder);
    int (*decodeFrm)(void *decoder, unsigned char
            *inbuf,int inLen, unsigned short *outbuf,int
            *outLen,int *chns);
    int (*getFrmInfo)(void *decoder, void *info);
    int (*closeDecoder)(void *decoder);
} IMPAudioDecDecoder;
```

【Members】

Member name	Description
type	Audio decoding protocol type
name	Audio decoder name
openDecoder	Open the decoder callback function
decodeFrm	Decoding the code-stream callback function
getFrmInfo	Get the code stream information callback function
closeDecoder	Turn off the decoder callback function

【NB】

None

【Related data types and interfaces】

IMP_ADEC_RegisterEncoder

2.12 SDK Sample Introduce

The Sample program is located at /samples/libimp-samples,Under the directory are applications that rely on the SDK library, including recording playback, response cancellation, and other applications. Users can refer to the provided sample program to write their own corresponding project code.

After compiling, copy the executable program to the development board for testing. Please refer to the example section of the following table:

Samplee	Function	Cmd	Result
sample-Ai.c	use analog mic to record	./sample-Ai	generate recording file ai_record.pcm

sample-Ao.c	play audio file	./sample-Ao	play audio file ao_paly.pcm with a sampling rate of 16k
sample-Ai-AEC.c	echo cancellation	./sample-Ai-AEC	generate two recording files test_for_play.pcm and test_aec_record.pcm after the program runs
sample-Ai-Ref.c	verify the analog mic audio to obtain the reference frame of the echo cancellation algorithm	./sample-Ai-Ref	generate two recording files test_for_play.pcm and test_aec_record.pcm after the program runs
sample-dmic.c	digital microphone array recording	./sample-dmic	generate dmic0_record.pcm, dmic1_record.pcm after the program runs,corresponding to the sound data recorded by MIC0 and MIC1 in the microphone array