
Juphoon Protocol Framework

Media

Published: Sep 2007

For more information on Juphoon Protocol Framework, see <http://www.juphoon.com>

Juphoon Media Integration Specification

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87287820

Fax: +86-574-87304379

Text Part Number:

Copyright © 2007, Juphoon System Software Corporation.

All rights reserved.

Contents

JUPHOON MEDIA INTEGRATION SPECIFICATION.....	1
1. INTRODUCTION.....	4
1.1 JUPHOON MEDIA	4
1.2 AUDIENCE	4
1.3 SCOPE.....	4
1.4 ABBREVIATIONS	4
2. SYSTEM ENVIRONMENT.....	4
2.1 BASIC DATA TYPES	4
3. INTERFACES REQUIREMENT.....	5
3.1 INTRODUCE.....	5
3.2 STRUCTURES	6
3.2.1 <i>ST_AUDIO_CODEC</i>	6
3.2.2 <i>ST_VIDEO_CODEC</i>	6
3.2.3 <i>ST_ZOS_INET_ADDR</i>	6
3.2.4 <i>PFN_AUDIORTCPAPPIND</i>	7
3.2.5 <i>PFN_AUDIOEVNTPROC</i>	7
3.2.6 <i>PFN_AUDIODTMFPROC</i>	7
3.2.7 <i>ST_MEDIA_FUNC_TBL</i>	7
3.2.8 <i>ST_AUDIO_FUNC_TBL</i>	8
3.2.9 <i>ST_VIDEO_FUNC_TBL</i>	10
3.3 MEDIA.....	10
3.3.1 <i>PFN_MEDIAOPEN</i>	10
3.3.2 <i>PFN_MEDIACLOSE</i>	11
3.4 AUDIO	11
3.4.1 <i>Basic</i>	12
3.4.2 <i>RTP / RTCP</i>	19
3.4.3 <i>DTMF</i>	21
3.4.4 <i>Conference</i>	23
3.4.5 <i>Sound Play</i>	24
3.4.6 <i>Sound Send</i>	28
3.4.7 <i>Sound Record</i>	30
3.4.8 <i>Qos and etc</i>	33
3.5 VIDEO.....	36
3.5.1 <i>Basic</i>	37
4. USAGE EXPLANATION.....	43
4.1 AUDIO	44

List of Tables

TABLE 2-1: BASIC DATA TYPES	5
TABLE 5-1: CALLING SEQUENCE FOR AUDIO CONVERSATION	46

1. Introduction

1.1 Juphoon Media

Juphoon Media Module provides the operating interface on media device and RTP channel. The module was designed to integrate with different media engine easily. To use the function of specific media engine, the interfaces of the media engine must meet the requirement of Juphoon Media Module.

1.2 Audience

The readers of this document are assumed to have a working knowledge of audio and video.

1.3 Scope

This document provides the definitions of Juphoon Media interfaces for users not the details on how to realize them.

1.4 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
AEC	Acoustic Echo Cancellation
AGC	Automatic Gain Control
RTP	Real-Time Transport Protocol
VAD	Voice Activity Detection
DTMF	Dual Tone Multi Frequency

2. System environment

2.1 Basic Data Types

There are some basic data types provided by ZOS platform. Table 2-1 lists these types used by the media module.

Name	Type
ZDOUBLE	double
ZFLOAT	float
ZLONG	long
ZINT	int
ZSHORT	short
ZCHAR	char
ZULONG	unsigned long
ZUINT	unsigned int
ZSIZE_T	unsigned int
ZUSHORT	unsigned short

ZUCHAR	unsigned char
ZBOOL	int
ZVOID	void

Table 2-1: Basic Data Types

3. Interfaces Requirement

3.1 Introduce

Juphoon Media Module has 3 parts of interfaces: media, audio and video. The interfaces of media were reserved for those media engine which can only operating audio and video simultaneously in some situations. And not all the interfaces of audio and video must be implemented if you only want the basic functions of conversation. The minimum set of interfaces for audio conversation includes:

- [PFN_AUDIOINIT](#)
- [PFN_AUDIODESTROY](#)
- [PFN_AUDIOOPEN](#)
- [PFN_AUDIOCLOSE](#)
- [PFN_AUDIOSETRMTADDR](#)
- [PFN_AUDIOSETSENDPAYLOAD](#)
- [PFN_AUDIOSETSEND](#)
- [PFN_AUDIOSETRECV](#)
- [PFN_AUDIOSETPLAY](#)
- [PFN_AUDIOSETREC](#)
- [PFN_AUDIOSETCDC](#)
- [PFN_AUDIOGETCDC](#)

And following interfaces should be implemented if you want a conversation with video.

- [PFN_VIDEOINIT](#)
- [PFN_VIDEODESTROY](#)
- [PFN_VIDEOOPEN](#)
- [PFN_VIDEOCLOSE](#)
- [PFN_VIDEOSETRMTADDR](#)
- [PFN_VIDEOSETSENDPAYLOAD](#)
- [PFN_VIDEOSETDISPLAYRECT](#)
- [PFN_VIDEOSETPREVIEWRECT](#)
- [PFN_VIDEORUN](#)
- [PFN_VIDEOSTOP](#)
- [PFN_VIDEOSETCDC](#)
- [PFN_VIDEOGETCDC](#)

3.2 Structures

3.2.1 ST_AUDIO_CODEC

This structure is used for Audio_SedCdc and Audio_GetCdc, which indicates the audio codec configuration including RTP payload, name, sampling rate, channel number, RTP packet time and bit rate.

```
typedef struct tagAUDIO_CODEC
{
    ZULONG dwPayload;           /* rtp payload type EN_RTP_PAYLOAD_TYPE */
    ZCHAR *pcName;             /* audio codec name */
    ZULONG dwSplRate;          /* audio sample rate (Hz) */
    ZULONG dwBitsPerSpl;       /* audio bits per sample */
    ZULONG dwChnlNum;          /* audio channel number */
    ZULONG dwPktTime;          /* audio packet time */
    ZULONG dwBitRate;          /* audio codec bit rate */
} ST_AUDIO_CODEC;
```

3.2.2 ST_VIDEO_CODEC

This structure is used for Video_SedCdc and Video_GetCdc, which indicates the video codec configuration.

```
typedef struct tagVIDEO_CODEC
{
    ZULONG dwPayload;           /* rtp payload type */
    ZUCHAR *pcName;            /* video codec name */
    ZULONG dwBitRate;          /* video codec bit rate */
    ZULONG dwFrmRate;          /* video codec frame rate */
    ZULONG dwXRes;              /* video x resolution */
    ZULONG dwYRes;              /* video y resolution */
} ST_VIDEO_CODEC;
```

3.2.3 ST_ZOS_INET_ADDR

It is the structure of ZOS IP address.

```

typedef struct tagZOS_INET_ADDR
{
    ZUCHAR ucType; /* ZINET_IPV4... */
    ZUCHAR aucSpare[1]; /* for 32 bit alignment */
    ZUSHORT wPort; /* not order[host or n/w] dependent */
    union
    {
        ZULONG dwIp; /* not order[host or n/w] dependent */
        ZUCHAR aucIp[ZINET_IPV4_ADDR_SIZE]; /* ipv4 address */
        ZUCHAR aucIpv6[ZINET_IPV6_ADDR_SIZE]; /* ipv6 address */
    } u;
} ST_ZOS_INET_ADDR;

```

3.2.4 PFN_AUDIORTCPAPPIND

This is the type definition of function which will be called after audio module received the RTCP application package.

```

typedef ZINT (* PFN_AUDIORTCPAPPIND)(ZULONG dwStrmId, ZUCHAR ucSubType,
    ZUINT iSsrc, ZUCHAR *pucAppData, ZUINT iAppLen)

```

3.2.5 PFN_AUDIOEVNTPROC

The action of this type definition of function is defined by audio engine. It can be used for inform user when event happened, like recording timeout, play sound over, etc.

```

typedef ZINT (*PFN_AUDIOEVNTPROC)(ZUCHAR ucEvtType, ZVOID *pEvent)

```

3.2.6 PFN_AUDIODTMFPROC

This is the type definition of function which will be called after audio module received the DTMF package according to RFC2833.

```

typedef ZVOID (* PFN_AUDIODTMFPROC)(ZULONG dwStrmId, ZUCHAR ucEventType)

```

3.2.7 ST_MEDIA_FUNC_TBL

This structure contains 2 interfaces which could be called for media stream opening and closing. It means opening and closing audio and video channel simultaneously. By default, they will be setted to the fake functions which were made for protocol singal test.

```
typedef struct tagMEDIA_FUNC_TBL
{
    PFN_MEDIAOPEN pfnOpen;          /* media open */
    PFN_MEDIACLOSE pfnClose;       /* media close */
} ST_MEDIA_FUNC_TBL;
```

3.2.8 ST_AUDIO_FUNC_TBL

This structure contains all interfaces used by audio module. These interfaces should be initialized to specific set of functions. By default, they will be setted to the fake functions which were made for protocol singal test.

```

/* audio function table */
typedef struct tagAUDIO_FUNC_TBL
{
    PFN_AUDIOINIT pfnInit;          /* audio init */
    PFN_AUDIODESTROY pfnDestroy;    /* audio destroy */
    PFN_AUDIOOPEN pfnOpen;         /* open a stream */
    PFN_AUDIOCLOSE pfnClose;       /* close a stream */
    PFN_AUDIORTPGETID pfnRtpGetId; /* get rtp id */
    PFN_AUDIORTCPENABLE pfnRtcpEnable; /* enable rtcp */
    PFN_AUDIORTCPAPPSEND pfnRtcpAppSend; /* send rtcp app package */
    PFN_AUDIORTCPAPPSETIND pfnRtcpAppSetInd; /* set rtcp app indicate */
    PFN_AUDIOSETDTMFCALLBACK pfnSetDtmfCallback; /* set dtmf callback */
    PFN_AUDIOSENDDTMF pfnSendDtmf; /* send dtmf */
    PFN_AUDIOTONEPLAY pfnTonePlay; /* play a tone */
    PFN_AUDIOTONESTOP pfnToneStop; /* stop playing tone */
    PFN_AUDIOSNDPLAYSTART pfnSndPlayStart; /* play sound data */
    PFN_AUDIOSNDPLAYSTARTX pfnSndPlayStartX; /* play sound data from file */
    PFN_AUDIOSNDPLAYSTOP pfnSndPlayStop; /* stop play sound */
    PFN_AUDIOSNDSSENDSTART pfnSndSendStart; /* send sound data */
    PFN_AUDIOSNDSSENDSTARTX pfnSndSendStartX; /* send sound from file */
    PFN_AUDIOSNDSSENDSTOP pfnSndSendStop; /* stop send sound */
    PFN_AUDIOSETRMTADDR pfnSetRmtAddr; /* set remote address */
    PFN_AUDIOSETSENDPAYLOAD pfnSetSendPayload; /* set sending payload */
    PFN_AUDIOGETSENDPAYLOAD pfnGetSendPayload; /* get sending payload */
    PFN_AUDIOSETSEND pfnSetSend; /* set sending state */
    PFN_AUDIOSETRECV pfnSetRecv; /* set receiving state */
    PFN_AUDIOSETPLAY pfnSetPlay; /* set playback state */
    PFN_AUDIOSETREC pfnSetRec; /* set recording state */
    PFN_AUDIOSETAEC pfnSetAec; /* set aec type */
    PFN_AUDIOGETAEC pfnGetAec; /* get aec state */
    PFN_AUDIOSETANR pfnSetAnr; /* set anr state */
    PFN_AUDIOGETANR pfnGetAnr; /* get anr state */
    PFN_AUDIOSETAGC pfnSetAgc; /* set agc state */
    PFN_AUDIOGETAGC pfnGetAgc; /* get agc state */
    PFN_AUDIOSETVAD pfnSetVad; /* set vad state */
    PFN_AUDIOGETVAD pfnGetVad; /* get vad state */
    PFN_AUDIOSETCONF pfnSetConf; /* set conference state of a stream */
    PFN_AUDIOGETCONF pfnGetConf; /* get conference state */
    PFN_AUDIOSETCDC pfnSetCdc; /* set codec setting */
    PFN_AUDIOGETCDC pfnGetCdc; /* get codec setting */
    PFN_AUDIORECPPLAYSTART pfnRecPlayStart; /* set codec setting */
    PFN_AUDIORECPPLAYSTOP pfnRecPlayStop; /* get codec setting */
    PFN_AUDIORECMICSTART pfnRecMicStart; /* set codec setting */
    PFN_AUDIORECMICSTOP pfnRecMicStop; /* get codec setting */
    PFN_AUDIORECCALLSTART pfnRecCallStart; /* set codec setting */

```

```

    PFN_AUDIORECCALLSTOP pfnRecCallStop; /* get codec setting */
} ST_AUDIO_FUNC_TBL;

```

3.2.9 ST_VIDEO_FUNC_TBL

This structure contains all interfaces used by video module. These interfaces should be initialized to specific set of functions. By default, they will be setted to the fake functions which were made for protocol singal test.

```

typedef struct tagVIDEO_FUNC_TBL
{
    PFN_VIDEOINIT pfnInit;          /* video init */
    PFN_VIDEODESTROY pfnDestroy;   /* video destroy */
    PFN_VIDEOOPEN pfnOpen;         /* video open stream */
    PFN_VIDEOCLOSE pfnClose;       /* video close stream */
    PFN_VIDEOSETRMTADDR pfnSetRmtAddr; /* video set remote address */
    PFN_VIDEOSETSENDPAYLOAD pfnSetSendPayload; /* video set sending payload */
    PFN_VIDEOSETDISPLAYRECT pfnSetDisplayRect; /* video set display rectangle */
    PFN_VIDEOSETPREVIEWRECT pfnSetPreviewRect; /* video set preview rectangle */
    PFN_VIDEORUN pfnRun;           /* video start transmitting */
    PFN_VIDEOSTOP pfnStop;         /* video stop transmitting */
    PFN_VIDEOSETCDC pfnSetCdc;     /* video set codec config */
    PFN_VIDEOGETCDC pfnGetCdc;     /* video get codec config */
} ST_VIDEO_FUNC_TBL;

```

3.3 Media

These interfaces used to operating audio and video simultaneously which depends on the implement of media engine.

3.3.1 PFN_MEDIAOPEN

This function is responsible for opening audio and video stream simultaneously.

```

typedef ZINT (* PFN_MEDIAOPEN)(ZULONG dwAudioIp, ZUSHORT wAudioPort,
                               ZULONG dwVideoIp, ZUSHORT wVideoPort, ZULONG *pdwStrmId)

```

[Parameters]

Input parameter:

ZULONG dwAudioIp

Local ip setting for audio stream.

ZUSHORT wAudioPort

RTP port number for audio stream. It will be assigned to the new RTP channel of audio stream.

ZULONG dwVideoIp

Local ip setting for video stream.

ZUSHORT wVideoPort

RTP port number for video stream. It will be assigned to the new RTP channel of video stream.

Output parameter:

ZULONG *pdwStrmId

A pointer to the stream id created.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.3.2 PFN_MEDIACLOSE

This function is responsible for closing audio and video stream simultaneously.

```
typedef ZINT (* PFN_MEDIACLOSE)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG dwStrmId

Indicates the stream which to close.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4 Audio

These interfaces used to operating audio alone. If the audio stream was created by *PFN_MEDIAOPEN*, the interface of *PFN_AUDIOOPEN* should not be called. Otherwise, it will create another audio stream which may not be what you expect.

3.4.1 Basic

3.4.1.1 *PFN_AUDIOINIT*

This function is responsible for initiating audio module to work properly and will be called before any other interfaces of audio.

```
typedef ZINT (* PFN_AUDIOINIT)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.2 *PFN_AUDIODESTROY*

This function is responsible for destroying audio module and releasing all related resources. It should stop any running stream and release resources related with them.

```
typedef ZVOID (* PFN_AUDIODESTROY)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.3 *PFN_AUDIOOPEN*

PFN_AUDIOOPEN is used to open an audio stream. It will allocate and initialize all resources with the audio stream. And it will also open a RTP channel with specific ip and port.

```
typedef ZINT (* PFN_AUDIOOPEN)(ZULONG dwIp, ZUSHORT wRtpPort,  
                                ZULONG *pdwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwIp*
Local ip setting.

ZUSHORT *wRtpPort*
RTP port number, which will be assigned to the new RTP channel which is to be opened later.

Output parameter:

ZULONG **pdwStrmId*
A pointer to the newly created stream ID.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.1.4 PFN_AUDIOCLOSE

PFN_AUDIOCLOSE closes an audio stream that was opened by *PFN_AUDIOOPEN* or *PFN_MEDIAOPEN*. It will first stop the stream while it in running, and release all resources of the audio stream, including RTP channel, codecs and etc.

```
typedef ZINT (* PFN_AUDIOCLOSE)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*
Indicates the stream be close.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.1.5 PFN_AUDIOSETEVNTCALLBACK

PFN_AUDIOSETEVNTCALLBACK sets the audio event callback process function.

```
typedef ZINT (* PFN_AUDIOSETEVNTCALLBACK)(PFN_AUDIOEVNTPROC pfnEvtProc)
```

[Parameters]

Input parameter:

PFN_AUDIOEVENTPROC pfnEvtProc

Indicates the audio event callback process function.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.6 PFN_AUDIOSETRMTADDR

This function is used to set the address of a remote host with which the local host will communicate.

```
typedef ZINT (* PFN_AUDIOSETRMTADDR)(ZULONG dwStrmId,  
                                     ST\_ZOS\_INET\_ADDR *pstRmtAddr)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

[ST_ZOS_INET_ADDR](#) **pstRmtAddr*

The address of the remote host with which the local host will communicate with.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.7 PFN_AUDIOSETSENDPAYLOAD

This function is used to set sending audio codec data by RTP payload. Usually, before this function was called, the codec's parameters were configed by *PFN_AUDIOGETCDC* and *PFN_AUDIOSETCDC*.

```
typedef ZINT (* PFN_AUDIOSETSENDPAYLOAD)(ZULONG dwStrmId, ZUCHAR ucPayload)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZUCHAR *ucPayload*

The RTP payload of audio codec.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.8 PFN_AUDIOGETSENDPAYLOAD

This function is used to get sending audio codec data by RTP payload.

```
typedef ZINT (* PFN_AUDIOGETSENDPAYLOAD)(ZULONG dwStrmId, ZUCHAR *pucPayload)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

Output parameter:

ZUCHAR **pucPayload*

Pointer to sending audio codec's payload.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.9 PFN_AUDIOSETSEND

PFN_AUDIOSETSEND is to sets the sending state of a specific stream. This interface should only take effect on RTP channel processing.

```
typedef ZINT (* PFN_AUDIOSETSEND)(ZULONG dwStrmId, ZBOOL bStart)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZBOOL *bStart*

If ZTRUE, indicates the audio stream to send data through the RTP channel. Or, stop sending.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.10 PFN_AUDIOSETRECV

PFN_AUDIOSETRECV is to sets the receiving state of a specific stream. This interface should only take effect on RTP channel processing.

```
typedef ZINT (* PFN_AUDIOSETRECV)(ZULONG dwStrmId, ZBOOL bStart)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZBOOL *bStart*

If ZTRUE, indicates the audio stream to process data received from the RTP channel. Or, stop processing.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.11 PFN_AUDIOSETPLAY

PFN_AUDIOSETPLAY is to sets the playing state of a specific stream. This interface will affect on play data filling to the device.

```
typedef ZINT (* PFN_AUDIOSETPLAY)(ZULONG dwStrmId, ZBOOL bStart)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZBOOL *bStart*

If ZTRUE, indicates the audio stream to play out data received from the RTP channel.

Or, stop playing.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.12 PFN_AUDIOSETREC

PFN_AUDIOSETREC is to sets the recording state of a specific stream. This interface will affect on recording data read from device.

```
typedef ZINT (* PFN_AUDIOSETREC)(ZULONG dwStrmId, ZBOOL bStart)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZBOOL *bStart*

If ZTRUE, indicates the audio stream to record data from sound device. Or, stop recording.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.13 PFN_AUDIOSETCDC

PFN_AUDIOSETCDC is to set an audio codec configuration. There are 5 parameters:

- **dwPayload** is the RTP payload of codec,
- **pcName** is the audio codec standard name, which used in SDP,

- **dwSplRate** is the codec sampling rate in Hz,
- **dwBitsPerSpl** is the bit width of a sample,
- **dwChnlNum** is the channel number of sampling data,
- **dwPktTime** is the RTP packet time in microseconds,
- **dwBitRate** is the audio codec bit rate.

In which, **pcName**, **dwSplRate**, **dwBitsPerSpl** and **dwChnlNum** would not be changed.

```
typedef ZINT (* PFN_AUDIOSETCDC)(ZULONG dwStrmId, ST\_AUDIO\_CODEC *pstCfg)
```

[Parameters]

Input parameter:

```
ZULONG dwStrmId
```

The audio stream ID.

```
ST\_AUDIO\_CODEC *pstCfg
```

Indicates the audio codec config.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.1.14 PFN_AUDIOGETCDC

PFN_AUDIOGETCDC gets the audio codec of specific name.

```
typedef ZINT (* PFN_AUDIOGETCDC)(ZULONG dwStrmId, ZCHAR *pcName,  
ST\_AUDIO\_CODEC *pstCdcCfg)
```

[Parameters]

Input parameter:

```
ZULONG dwStrmId
```

The audio stream ID.

```
ZCHAR *pcName
```

Audio codec name.

Output parameter:

```
ST\_AUDIO\_CODEC *pstCdcCfg
```

Point to the codec if found, else it will be setted to ZNULL.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.2 RTP / RTCP

3.4.2.1 *PFN_AUDIORTPGETID*

PFN_AUDIORTPGETID is to get the RTP channel id, which only implement by Juphoon audio engine.

```
typedef ZULONG (* PFN_AUDIORTPGETID)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

```
ZULONG dwStrmId
```

Indicates the stream to get RTP id of which.

Output parameter:

None.

[Return value]

RTP channel id associate with specific stream if created, otherwise ZMAXULONG.

3.4.2.2 *PFN_AUDIORTCPENABLE*

PFN_AUDIORTCPENABLE is to enable or disable the RTCP functions in RTP channel.

```
typedef ZINT (* PFN_AUDIORTCPENABLE)(ZULONG dwStrmId, ZBOOL bEnable)
```

[Parameters]

Input parameter:

```
ZULONG dwStrmId
```

Audio stream ID, created by *Audio_Open*.

```
ZBOOL bEnable
```

Set ZTRUE to enable RTCP function, ZFALSE to disable RTCP function.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.2.3 PFN_AUDIORTCPAPPSSEND

PFN_AUDIORTCPAPPSSEND is to send application RTCP package through the RTP channel of audio stream.

```
typedef ZINT (* PFN_AUDIORTCPAPPSSEND)(ZULONG dwStrmId, ZUCHAR ucSubType,  
                                         ZUCHAR *pucData, ZULONG dwLen)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*
Audio stream ID, created by *Audio_Open*.

ZUCHAR *ucSubType*
Subtype value of application RTCP package.

ZUCHAR **pucData*
Pointer to application RTCP data.

ZULONG *dwLen*
Length of application RTCP data.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.2.4 PFN_AUDIORTCPAPPSETIND

PFN_AUDIORTCPAPPSETIND is to set RTCP callback for application package processing after received.

```
typedef ZINT (* PFN_AUDIORTCPAPPSETIND)(ZULONG dwStrmId,  
                                         PFN\_AUDIORTCPAPPIND pfnRtcpAppInd)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*
Audio stream ID, created by *Audio_Open*.

[PFN_AUDIORTCPAPPIND](#) *pfnRtcpAppInd*
RTCP callback for application package processing.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.3 DTMF

3.4.3.1 PFN_AUDIOSETDTMFCALLBACK

PFN_AUDIOSETDTMFCALLBACK is to set audio engine callback for DTMF processing after received (RFC2833).

```
typedef ZINT (* PFN_AUDIOSETDTMFCALLBACK)(ZULONG dwStrmId,
                                           PFN\_AUDIODTMFPROC pfnDtmfProc)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Audio stream ID, created by *Audio_Open*.

[PFN_AUDIODTMFPROC](#) *pfnDtmfProc*

Audio engine callback for DTMF processing.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.3.2 PFN_AUDIOSENDDTMF

PFN_AUDIOSENDDTMF is to send DTMF data through the audio stream. This interface support 2 types of DTMF data, inband and outband (RFC2833).

```
typedef ZINT (* PFN_AUDIOSENDDTMF)(ZULONG dwStrmId, ZUCHAR ucEvtType,
                                     ZULONG dwLen, ZBOOL bInBand);
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Audio stream ID.

ZUCHAR *ucEvtType*

Dtmf type defined in RFC2833. The allowable event types are: (EN_AUDIO_DTMF_0 to EN_AUDIO_DTMF_D are supported in inband mode)

```
/* DTMF named events */
EN_AUDIO_DTMF_0 = 0,
EN_AUDIO_DTMF_1 = 1,
EN_AUDIO_DTMF_2 = 2,
EN_AUDIO_DTMF_3 = 3,
EN_AUDIO_DTMF_4 = 4,
EN_AUDIO_DTMF_5 = 5,
EN_AUDIO_DTMF_6 = 6,
EN_AUDIO_DTMF_7 = 7,
EN_AUDIO_DTMF_8 = 8,
EN_AUDIO_DTMF_9 = 9,
EN_AUDIO_DTMF_STAR = 10,
EN_AUDIO_DTMF_POUND = 11,
EN_AUDIO_DTMF_A = 12,
EN_AUDIO_DTMF_B = 13,
EN_AUDIO_DTMF_C = 14,
EN_AUDIO_DTMF_D = 15,
EN_AUDIO_DTMF_FLASH = 16,

/* E.182 line events */
EN_AUDIO_DTMF_OFF_HOOK = 64, /* off hook */
EN_AUDIO_DTMF_ON_HOOK = 65, /* on hook */
EN_AUDIO_DTMF_DIAL = 66, /* Dial tone */
EN_AUDIO_DTMF_PABX_DT = 67, /* PABX internal dial tone */
EN_AUDIO_DTMF_SPECIAL_DT = 68, /* Special dial tone */
EN_AUDIO_DTMF_SECOND_DT = 69, /* Second dial tone */
EN_AUDIO_DTMF_RING_BACK = 70, /* Ringing tone */
EN_AUDIO_DTMF_SPEC_RT = 71, /* Special ringing tone */
EN_AUDIO_DTMF_BUSY = 72, /* Busy tone */
EN_AUDIO_DTMF_CONG = 73, /* Congestion tone */
EN_AUDIO_DTMF_SPEC_INFO = 74, /* Special information tone */
EN_AUDIO_DTMF_COMF = 75, /* Comfort tone */
EN_AUDIO_DTMF_HOLD = 76, /* Hold tone */
EN_AUDIO_DTMF_RECORD = 77, /* Record tone */
EN_AUDIO_DTMF_CALLER_WAIT = 78, /* Caller waiting tone */
EN_AUDIO_DTMF_CALL_WAIT = 79, /* Call waiting tone */
EN_AUDIO_DTMF_PAY = 80, /* Pay tone */
EN_AUDIO_DTMF_POS_IND = 81, /* Positive indication tone */
EN_AUDIO_DTMF_NEG_IND = 82, /* Negative indication tone */
```

```

EN_AUDIO_DTMF_WARN = 83,          /* Warning tone */
EN_AUDIO_DTMF_INTRUSION = 84,     /* Intrusion tone */
EN_AUDIO_DTMF_CALL_CARD = 85,     /* Calling card service tone */
EN_AUDIO_DTMF_PAYPHONE = 86,     /* Payphone recognition tone */
EN_AUDIO_DTMF_CAS = 87,          /* CPE alerting signal (CAS) */
EN_AUDIO_DTMF_OFF_HOOK_WARN = 88, /* Off-hook warning tone */
EN_AUDIO_DTMF_RING = 89          /* Ring */

```

ZULONG *dwLen*

Indicates how many milliseconds the DTMF data should last.

ZBOOL *bInBand*

If ZTRUE, the DTMF data will be sent by inband mode. Or it will be sent by outband mode (RFC2833).

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.4 Conference

3.4.4.1 PFN_AUDIOSETCONF

PFN_AUDIOSETCONF is to add / remove a specific stream to / from a conference. There would be only one conference at most.

```
typedef ZINT (* PFN_AUDIOSETCONF)(ZULONG dwStrmId, ZBOOL bEnable)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

ZBOOL *bEnable*

If it sets to ZTRUE, indicate audio engine to mix the sound data of the stream to the conference.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.4.2 PFN_AUDIOGETCONF

PFN_AUDIOGETCONF checks whether the specific stream is set to conference.

```
typedef ZINT (* PFN_AUDIOGETCONF)(ZULONG dwStrmId, ZBOOL *pbEnable)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

Output parameter:

ZBOOL **pbEnable*

A pointer to a Boolean variable. If the stream is added to conference, the value will set ZTRUE, or it will be ZFALSE.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.5 Sound Play

3.4.5.1 PFN_AUDIOTONEPLAY

PFN_AUDIOTONEPLAY is used to indicate audio module to generate and play a certain DTMF tone. So the tone type has the same value with the DTMF event type. It can assign a time length to play. If the time length is 0, means expecting never stop playing tone until *PFN_AUDIOTONESTOP* was called.

```
typedef ZINT (* PFN_AUDIOTONEPLAY)(ZUCHAR ucToneType, ZULONG dwLen)
```

[Parameters]

Input parameter:

ZUCHAR *ucToneType*

Tone type, indicating which kind of sound is to be played. The allowable event types are:

```

EN_AUDIO_TONE_DIAL = EN_AUDIO_DTMF_DIAL,
EN_AUDIO_TONE_BUSY = EN_AUDIO_DTMF_BUSY,
EN_AUDIO_TONE_RING = EN_AUDIO_DTMF_RING,
EN_AUDIO_TONE_RING_BACK = EN_AUDIO_DTMF_RING_BACK,
EN_AUDIO_TONE_CONGESTION = EN_AUDIO_DTMF_CONG,
EN_AUDIO_TONE_SPEC_INFO = EN_AUDIO_DTMF_SPEC_INFO,
EN_AUDIO_TONE_WARN = EN_AUDIO_DTMF_WARN,
EN_AUDIO_TONE_CALL_WAIT = EN_AUDIO_DTMF_CALL_WAIT,
EN_AUDIO_TONE_CALLER_WAIT = EN_AUDIO_DTMF_CALLER_WAIT,
EN_AUDIO_TONE_ON_HOOK = EN_AUDIO_DTMF_ON_HOOK,
EN_AUDIO_TONE_0 = EN_AUDIO_DTMF_0,
EN_AUDIO_TONE_1 = EN_AUDIO_DTMF_1,
EN_AUDIO_TONE_2 = EN_AUDIO_DTMF_2,
EN_AUDIO_TONE_3 = EN_AUDIO_DTMF_3,
EN_AUDIO_TONE_4 = EN_AUDIO_DTMF_4,
EN_AUDIO_TONE_5 = EN_AUDIO_DTMF_5,
EN_AUDIO_TONE_6 = EN_AUDIO_DTMF_6,
EN_AUDIO_TONE_7 = EN_AUDIO_DTMF_7,
EN_AUDIO_TONE_8 = EN_AUDIO_DTMF_8,
EN_AUDIO_TONE_9 = EN_AUDIO_DTMF_9,
EN_AUDIO_TONE_STAR = EN_AUDIO_DTMF_STAR,
EN_AUDIO_TONE_POUND = EN_AUDIO_DTMF_POUND,
EN_AUDIO_TONE_A = EN_AUDIO_DTMF_A,
EN_AUDIO_TONE_B = EN_AUDIO_DTMF_B,
EN_AUDIO_TONE_C = EN_AUDIO_DTMF_C,
EN_AUDIO_TONE_D = EN_AUDIO_DTMF_D

```

ZULONG *dwLen*

Indicates how many milliseconds the sound should be played.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.5.2 *PFN_AUDIOTONESTOP*

PFN_AUDIOTONESTOP is used to indicate audio module to stop playing tone.

```
typedef ZINT (* PFN_AUDIOTONESTOP)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.5.3 PFN_AUDIOSNDPLAYSTART

PFN_AUDIOSNDPLAYSTART loads sound buffer data from memory to playback at the local side. The file data must be pcm raw data, 8000Hz sampling, 16bits quantized, mono. The time length has the same meaning with *PFN_AUDIOTONEPLAY*'s. Additional, it can also assign a play cycle count. If both the time length and play cycle are 0, means expecting never stop playing until *PFN_AUDIOSNDSTOP* was called.

```
typedef ZINT (* PFN_AUDIOSNDPLAYSTART)(ZUCHAR *pucData, ZULONG dwSize,
                                       ZULONG dwLen, ZULONG dwCycle)
```

[Parameters]

Input parameter:

ZUCHAR *pucData

Pcm data which to be loaded to audio engine for playing.

ZULONG dwSize

Length in byte of pcm data.

ZULONG dwLen

Indicates the time length in second which the sound played.

ZULONG dwCycle

Indicates the cycle count which the sound played. The time length and the cycle count can be both setted with a non-zero number. The sound stoped if either of them reached. If both of them setted with 0, the sound will never stop until *Audio_SndPlayStop* has been called.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.5.4 *PFN_AUDIOSNDPLAYSTARTX*

PFN_AUDIOSNDPLAYSTARTX loads sound buffer data from file to prepare to playback at the local side. The time length and play cycle have the same meaning with *PFN_AUDIOSNDPLAYSTART*'s.

```
typedef ZINT (* PFN_AUDIOSNDPLAYSTARTX)(ZCHAR *pcFileName, ZCHAR *pcFileType,
                                         ZULONG dwLen, ZULONG dwCycle)
```

[Parameters]

Input parameter:

ZCHAR *pcFileName

A pointer to the name of sound file which is to be loaded.

ZCHAR *pcFileType

The sound file type.

ZULONG *dwLen*

Indicates the time length in second which the sound played.

ZULONG *dwCycle*

Indicates the cycle count which the sound played. The time length and the cycle count can be both setted with a non-zero number. The sound stoped if either of them reached. If both of them setted with 0, the sound will never stop until `Audio_SndPlayStop` has been called.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.5.5 *PFN_AUDIOSNDPLAYSTOP*

PFN_AUDIOSNDPLAYSTOP stops playing sound at the local side.

```
typedef ZINT (* PFN_AUDIOSNDPLAYSTOP)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.6 Sound Send

3.4.6.1 PFN_AUDIOSNSENDSTART

PFN_AUDIOSNSENDSTART loads sound buffer data from memory to be sent to the remote side of the stream, and the sound will not play at local host. The file data must be pcm raw data, 8000Hz sampling, 16bits quantized, mono. The time length and play cycle have the same meaning with *PFN_AUDIOSNDPLAY*'s.

```
typedef ZINT (* PFN_AUDIOSNSENDSTART)(ZULONG dwStrmId, ZUCHAR *pucData,  
                                       ZULONG dwSize, ZULONG dwLen, ZULONG dwCycle)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream which the sound should be sent through.

ZUCHAR **pucData*

Pcm data which to be loaded to audio engine for sending.

ZULONG *dwSize*

Length in byte of pcm data.

ZULONG *dwLen*

Indicates the time length in second which the sound sent.

ZULONG *dwCycle*

Indicates the cycle count which the sound sent. The time length and the cycle count can be both setted with a non-zero number. The sound sending stoped if either of them reached. If both of them setted with 0, the sound sending will never stop until *Audio_SndSendStop* has been called.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.6.2 *PFN_AUDIOSNDSSENDSTARTX*

PFN_AUDIOSNDSSENDSTARTX loads sound buffer data from file to be sent to the remote side of the stream. The time length and play cycle have the same meaning with *PFN_AUDIOSNDSSENDSTART*'s.

```
typedef ZINT (* PFN_AUDIOSNDSSENDSTARTX)(ZULONG dwStrmId, ZCHAR *pcFileName,
                                          ZCHAR *pcFileType, ZULONG dwLen, ZULONG dwCycle)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream which the sound should be sent through.

ZCHAR **pcFileName*

A pointer to the name of sound file which is to be loaded.

ZCHAR **pcFileType*

The sound file type.

ZULONG *dwLen*

Indicates the time length in second which the sound sent.

ZULONG *dwCycle*

Indicates the cycle count which the sound sent. The time length and the cycle count can be both setted with a non-zero number. The sound sending stoped if either of them reached. If both of them setted with 0, the sound sending will never stop until *Audio_SndPlayStop* has been called.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.6.3 *PFN_AUDIOSNDSSENDSTOP*

PFN_AUDIOSNDSSENDSTOP indicate the audio module to stop sending.

```
typedef ZINT (* PFN_AUDIOSNDSSENDSTOP)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream which's sound sending should be stopped.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7 Sound Record

3.4.7.1 *PFN_AUDIORECPLAYSTART*

PFN_AUDIORECPLAYSTART indicates the audio engine to start recording the playout sound of a specific stream.

```
typedef ZINT (* PFN_AUDIORECPLAYSTART)(ZULONG dwStrmId, ZCHAR *pcFileName,  
ZCHAR *pcCdcName)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream which to be recorded.

ZCHAR **pcFileName*

Indicates the file name to which the recording data saved.

ZCHAR **pcCdcName*

Indicates the recording data format.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7.2 *PFN_AUDIORECPLAYSTOP*

PFN_AUDIORECPLAYSTOP indicates the audio engine to stop recording the playout sound of a specific stream.

```
typedef ZINT (* PFN_AUDIORECPLAYSTOP)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream which to be recorded.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7.3 PFN_AUDIORECMICSTART

PFN_AUDIORECMICSTART indicates the audio engine to start recording the microphone at the local side. Recording starts whenever a stream exists.

```
typedef ZINT (* PFN_AUDIORECMICSTART)(ZCHAR *pcFileName, ZCHAR *pcCdcName)
```

[Parameters]

Input parameter:

ZCHAR **pcFileName*

Indicates the file name to which the recording data saved.

ZCHAR **pcCdcName*

Indicates the recording data format.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7.4 PFN_AUDIORECMICSTOP

PFN_AUDIORECMICSTOP indicates the audio engine to stop recording the microphone.

```
typedef ZINT (* PFN_AUDIORECMICSTOP)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7.5 *PFN_AUDIORECCALLSTART*

PFN_AUDIORECCALLSTART indicates the audio engine to start recording the sound of the call. Both records playing and recording data. Recording should never start until there is an audio stream operated at least.

```
typedef ZINT (* PFN_AUDIORECCALLSTART)(ZCHAR *pcFileName, ZCHAR *pcCdcName)
```

[Parameters]

Input parameter:

ZCHAR **pcFileName*

Indicates the file name to which the recording data saved.

ZCHAR **pcCdcName*

Indicates the recording data format.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.7.6 *PFN_AUDIORECCALLSTOP*

PFN_AUDIORECCALLSTOP indicates the audio engine to stop recording.

```
typedef ZINT (* PFN_AUDIORECCALLSTOP)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.8 Qos and etc

3.4.8.1 PFN_AUDIOSETAEC

PFN_AUDIOSETAEC is to sets [Acoustic Echo Cancellation \(AEC\)](#) operation type of the audio module.

```
typedef ZINT (* PFN_AUDIOSETAEC)(ZUCHAR ucType)
```

[Parameters]

Input parameter:

ZUCHAR *ucType*

Indicates the type of AEC operation, which should be one of:

EN_AUDIO_EC_OFF = 0,

EN_AUDIO_EC_AEC = 1,

EN_AUDIO_EC_AES = 2

EN_AUDIO_EC_AEC and EN_AUDIO_EC_AES are both indicate the audio engine enable AEC.

EN_AUDIO_EC_AEC is for full-duplex, EN_AUDIO_EC_AES is for half-duplex.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.8.2 PFN_AUDIOGETAEC

PFN_AUDIOGETAEC checks whether the [AEC](#) is enabled.

```
typedef ZINT (* PFN_AUDIOGETAEC)(ZBOOL *pbEnable)
```

[Parameters]

Input parameter:

None.

Output parameter:

ZBOOL **pbEnable*

A pointer to a Boolean variable. If AEC is enabled, the variable will be ZTRUE. Else it will be ZFALSE.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.8.3 *PFN_AUDIOSETANR*

PFN_AUDIOSETANR is to enable or disable decrease noise operation in the audio module.

```
typedef ZINT (* PFN_AUDIOSETANR)(ZBOOL bEnable)
```

[Parameters]

Input parameter:

ZBOOL *bEnable*

If it sets to ZTRUE, then the audio module will be enabled to perform decrease noise on the recording data. Otherwise, the audio module will not to perform decrease noise.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.8.4 *PFN_AUDIOGETANR*

PFN_AUDIOGETANR checks whether the decrease noise is enabled.

```
typedef ZINT (* PFN_AUDIOGETANR)(ZBOOL *pbEnable)
```

[Parameters]

Input parameter:

None.

Output parameter:

ZBOOL **pbEnable*

A pointer to a Boolean variable. If decrease noise is enabled, the Boolean variable is ZTRUE, else is ZFALSE.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.8.5 PFN_AUDIOSETAGC

PFN_AUDIOSETAGC enable or disable [Automatic gain control \(AGC\)](#) operation in the audio module.

```
typedef ZINT (* PFN_AUDIOSETAGC)(ZBOOL bEnable)
```

[Parameters]

Input parameter:

ZBOOL *bEnable*

If it sets to ZTRUE, then the audio module will be enabled to perform [AGC](#). Otherwise, the audio module will not to perform [AGC](#).

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.8.6 PFN_AUDIOGETAGC

PFN_AUDIOGETAGC checks whether the [AGC](#) is enabled.

```
typedef ZINT (* PFN_AUDIOGETAGC)(ZBOOL *pbEnable)
```

[Parameters]

Input parameter:

None.

Output parameter:

ZBOOL **pbEnable*

A pointer to a Boolean variable. If [AGC](#) is enabled, the Boolean variable is ZTRUE, else is ZFALSE.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

3.4.8.7 PFN_AUDIOSETVAD

PFN_AUDIOSETVAD enable or disable [Voice Activity Detection \(VAD\)](#) operation in the audio module.

```
typedef ZINT (* PFN_AUDIOSETVAD)(ZBOOL bEnable)
```

[Parameters]

Input parameter:

ZBOOL *bEnable*

If it sets to ZTRUE, then the audio module will be enabled to perform [VAD](#). Otherwise, the audio module will not to perform [VAD](#).

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.4.8.8 PFN_AUDIOGETVAD

PFN_AUDIOGETVAD checks whether the [VAD](#) is enabled .

```
typedef ZINT (* PFN_AUDIOGETVAD)(ZBOOL *pbEnable)
```

[Parameters]

Input parameter:

None.

Output parameter:

ZBOOL **pbEnable*

A pointer to a Boolean variable. If [VAD](#) is enabled, the Boolean variable is ZTRUE, else is ZFALSE.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5 Video

These interfaces used to operating video alone. If the video stream was created by *PFN_MEDIAOPEN*, the interface of *PFN_VIDEOOPEN* should not be called. Otherwise, it will create another video stream which may not be what you expect.

3.5.1 Basic

3.5.1.1 *PFN_VIDEONIT*

This function is responsible for initiating video module to work properly. This function will be called before any other interface of video.

```
typedef ZINT (*PFN_VIDEONIT)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.2 *PFN_VIDEODESTROY*

This function is responsible for destroying video module and releasing all related resources. This function may be called there has a video stream running.

```
typedef ZVOID (*PFN_VIDEODESTROY)()
```

[Parameters]

Input parameter:

None.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.3 *PFN_VIDEOPEN*

PFN_VIDEOPEN is used to open a video stream. It will also open a RTP channel with specific IP and port along with the video stream. This function would not be called if the audio and video stream has been created by *PFN_MEDIAOPEN* before.

```
typedef ZINT (*PFN_VIDEOPEN)(ZULONG dwIp, ZUSHORT wRtpPort, ZULONG *pdwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwIp*

Local ip setting.

ZUSHORT *wRtpPort*

RTP port number, which will be assigned to the new RTP channel which is to be opened later.

Output parameter:

ZULONG **pdwStrmId*

A pointer to the newly created stream ID.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

3.5.1.4 *PFN_VIDEOCLOSE*

PFN_VIDEOCLOSE closes a video stream that was opened by *PFN_VIDEOOPEN* or *PFN_MEDIAOPEN*.

```
typedef ZINT (* PFN_VIDEOCLOSE)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

Indicates the stream be close.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.5 *PFN_VIDEOSETRMTADDR*

This function is used to set the address of a remote host with which the local host will communicate with.

```
typedef ZINT (* PFN_VIDEOSETRMTADDR)(ZULONG dwStrmId,
                                     ST\_ZOS\_INET\_ADDR *pstRmtAddr)
```

[Parameters]

Input parameter:

`ZULONG dwStrmId`

The audio stream ID.

`ST_ZOS_INET_ADDR *pstRmtAddr`

The address of the remote host with which the local host will communicate with.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.6 PFN_VIDEOSETSENDPAYLOAD

This function is used to set sending video codec data by RTP payload. Like corresponding interface in audio, before this function was called, the codec's parameters were configed by *PFN_VIDEOGETCDC* and *PFN_VIDEOSETCDC*.

```
typedef ZINT (* PFN_VIDEOSETSENDPAYLOAD)(ZULONG dwStrmId, ZUCHAR ucPayload)
```

[Parameters]

Input parameter:

`ZULONG dwStrmId`

The video stream ID.

`ZUCHAR ucPayload`

The RTP payload of video codec.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.7 PFN_VIDEOSETDISPLAYRECT

This function is used to set display rectangle in which the video from remote will display.

```
typedef ZINT (* PFN_VIDEOSETDISPLAYRECT)(ZULONG dwStrmId, ZULONG dwX, ZULONG dwY,  
ZULONG dwWidth, ZULONG dwHeight)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The video stream ID.

ZULONG *dwX*

Indicates the x coordinate of top-left corner of the display rectangle.

ZULONG *dwY*

Indicates the y coordinate of top-left corner of the display rectangle.

ZULONG *dwWidth*

Indicates the width of display rectangle in pixel.

ZULONG *dwHeight*

Indicates the height of display rectangle in pixel.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.8 *PFN_VIDEOSETPREVIEWRECT*

This function is used to set preview rectangle in which the video from local camera will display. It would support PinP mode.

```
typedef ZINT (* PFN_VIDEOSETPREVIEWRECT)(ZULONG dwStrmId, ZULONG dwX, ZULONG dwY,  
ZULONG dwWidth, ZULONG dwHeight)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The video stream ID.

ZULONG *dwX*

Indicates the x coordinate of top-left corner of the preview rectangle.

ZULONG *dwY*

Indicates the y coordinate of top-left corner of the preview rectangle.

ZULONG *dwWidth*

Indicates the width of preview rectangle in pixel.

ZULONG *dwHeight*

Indicates the height of preview rectangle in pixel.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.9 PFN_VIDEORUN

This function is used to start the processing of video data. It will get the video engine to handle the video data from local camera and remote host, and then display them on the screen.

```
typedef ZINT (* PFN_VIDEORUN)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The video stream ID.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.10 PFN_VIDEOSTOP

This function is used to stop the processing of video data.

```
typedef ZINT (* PFN_VIDEOSTOP)(ZULONG dwStrmId)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The video stream ID.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.11 PFN_VIDEOSETCDC

PFN_VIDEOSETCDC is to set the configuration of one video codec. There are 6 parameters:

- **dwPayload** is the RTP payload of codec,
- **pcName** is the standard video codec name in SDP which can't be changed,
- **dwBitRate** is the codec bit rate in kbps,
- **dwFrmRate** is the codec frame rate in fps,
- **dwXRes** is the resolution of x coordinate,
- **dwYRes** is the resolution of y coordinate.

```
typedef ZINT (* PFN_VIDEOSETCDC)(ZULONG dwStrmId, ST\_VIDEO\_CODEC *pstCfg)
```

[Parameters]

Input parameter:

ZULONG *dwStrmId*

The audio stream ID.

[ST_VIDEO_CODEC](#) **pstCfg*

Indicates the video codec config.

Output parameter:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

3.5.1.12 PFN_VIDEOGETCDC

PFN_VIDEOGETCDC gets the video codec of specific name.

```
typedef ZINT (* PFN_VIDEOGETCDC)(ZULONG dwStrmId, ZCHAR *pcName,  
                                ST_VIDEO_CODEC *pstCdcCfg)
```

[Parameters]

Input parameter:

```
ZULONG dwStrmId  
The video stream ID.
```

```
ZCHAR *pcName  
Video codec name.
```

Output parameter:

```
ST_VIDEO_CODEC *pstCdcCfg  
Point to the codec if found, else it will be setted to ZNULL.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

4. Usage Explanation

We here only take the audio module for example, and assume following functions have been implemented.

```
Xxx_AudioInit()  
Xxx_AudioDestory()  
Xxx_AudioOpen()  
Xxx_AudioClose()  
.....
```

To use the customized audio module, you should add statements to re-initialise the audio functions table. The program may look like:

```

Int main()
{
    ST_AUDIO_FUNC_TBL stFuncTbl;

    .....
    /* call module config init */
    Audio_CfgInit();

    .....
    /* re-initialise the audio function table */
    stFuncTbl.pfnInit = Xxx_AudioInit;
    stFuncTbl.pfnInit = Xxx_AudioInit;
    stFuncTbl.pfnOpen = Xxx_AudioOpen;
    stFuncTbl.pfnClose = Xxx_AudioClose;
    Audio_CfgSetFuncTbl(&stFuncTbl);

    .....
    /* call module start */
    Audio_Start();

    .....
    /* start the program process */
    .....
}

```

Video integration may take the similar operation before use.

4.1 Audio

Example in this section assume only audio can be used. So during the stage of initialization of program, *Media_Start* and *Video_Start* were not needed. Following table shows the calling sequence during a common conversation.

Audio Interface called	Function Called	Explains
Initialization		
Audio_CfgInit()		Set the default config of audio. After returned from this function, you can customize the config of audio. If you do not want to change anything, calling this interface is not necessary.
Audio_Start()	PFN_AUDIOINIT	To do the initialization work in the module.
Prepare for Call		
ZULONG dwStrmId, dwIp; Zos_InetAddr("192.168.0.45", &dwIp); Audio_Open(dwIp, 37000, &dwStrmId);	PFN_AUDIOOPEN	Open an audio stream with specific ip and port.
Set the Stream Configuration		

<p>ST_AUDIO_CODEC stCodec;</p> <p>Audio_GetCdc(dwStrmId, "PCMU", &stCodec);</p> <p>[SET THE CODEC WITH PARAMETERS NEGOCIATED]</p> <p>Audio_SetCdc(dwStrmId, &stCodec);</p>	<p>PFN_AUDIOGETCDC</p> <p>PFN_AUDIOSETCDC</p>	<p>After exchange of signal protocol, you may set the codec with the parameters negociated.</p>
<p>Audio_SetSendPayload(dwStrmId, 0);</p>	<p>PFN_AUDIOSETSENDPAYLOAD</p>	<p>Set the sending RTP payload.</p>
<p>ST_ZOS_INET_ADDR stPAddr;</p> <p>ZOS_IPV4_ADDR_SET_STR(&stPAddr, "192.168.0.250");</p> <p>stPAddr.wPort = 37000;</p> <p>Audio_SetRmtAddr(dwStrmId, &stPAddr);</p>	<p>PFN_AUDIOSETRMTADDR</p>	<p>Set the remote host address negociated by signal protocol.</p>
Start a Normal Conversation		
<p>Audio_SetRecv(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETRECV</p>	<p>Start processing data receiving from RTP channel.</p>
<p>Audio_SetSend(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETSEND</p>	<p>Start sending data at local side.</p>
<p>Audio_SetPlay(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETPLAY</p>	<p>Start playing out the receiving data from remote host.</p>
<p>Audio_SetRec(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETREC</p>	<p>Start recording data from microphone.</p>
Mute the Microphone		
<p>Audio_SetRec(dwStrmId, ZFALSE);</p>	<p>PFN_AUDIOSETREC</p>	<p>Stop recording data from microphone.</p>
Un-Mute the Microphone		
<p>Audio_SetRec(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETREC</p>	<p>Start recording data from microphone.</p>
Hold the Conversation		
<p>Audio_SetRecv(dwStrmId, ZFALSE);</p>	<p>PFN_AUDIOSETRECV</p>	<p>Stop processing data receiving from RTP channel.</p>
<p>Audio_SetRec(dwStrmId, ZFALSE);</p>	<p>PFN_AUDIOSETREC</p>	<p>Stop recording data from microphone.</p>
Un-Hold the Conversation		
<p>Audio_SetRecv(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETRECV</p>	<p>Start processing data receiving from RTP channel.</p>
<p>Audio_SetRec(dwStrmId, ZTRUE);</p>	<p>PFN_AUDIOSETREC</p>	<p>Start recording data from</p>

		microphone.
Call Terminated		
Audio_Close(dwStrmId);	PFN_AUDIOCLOSE	Stop and close an audio stream. Then release the resources of the stream.

Table 4-1: Calling Sequence for Audio Conversation