SWIPPA SDK User Manual

History:

Time	Content	Author	Version
2005-06-14	1st version	Jianming. Feng Senior R&D Manager.	1.0
2010-02-22	2nd version	Jianming. Feng Cindy. Shen	1.1
2010-03-05	3rd version	Jianming. Feng	2.0

Contents

1 SWIPPA Introduction

SWIPPA™ is a VoIP recording component, with high customizability and compatibility in any special situation. SWIPPA™ module is designed around client/server architecture, in which the client is the application requesting the recording service, and the server providing the service is the recording module. The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve flexibility and interoperability as compared to centralized and time sharing processing. A single machine can play both client and server roles, depending on the software configuration, which means both application and the recording modules can be loaded into one machine. Moreover, the client can be served by multiple recording servers.

SWIPPA consists of SWIPPA runtime library and SWIPPA SDK.

- SWIPPA runtime library is to direct VoIP protocol implementation and provide recording control interface.
- SWIPPA SDK provides recording application interface and related documents.

1.1 SWIPPA SDK

Get SWIPPA SDK from SWIPPA release package, for details check “SWIPPA Quick Start Manual”.

SWIPPA SDK includes following documents:

Serial No.	File	Description
1	SWIPPA32c.h	SWIPPA Client Lib C language head file
2	SWIPPA32c.lib	SWIPPA Client Lib lib file
3	SWIPPA SDK User Manual	The manual

1.2 Reference

SWIPPA Quick Start Manual.

2 Constants Definition

2.1 Record Event Type

```
enum VLM_Event
{
        EventReportInfo          = 200,
        EventBeginRecord         = 201,
        EventEndRecord           = 202,
        EventCaptchDTMF          = 203,
        EventUpdateStatus        = 204,
```

};

2.2 Audio File Encoding Format

Constants Name	Value	Description
CODEC_ALAUW	0x06	G.711.a
CODEC_ULAUW	0x07	G.711.u
CODEC_ADPCM	0x11	ADPCM
CODEC_GSM610	0x31	GSM
CODEC_MP3	0x55	MP3

2.3 Extension Identification Type

Constants Name	Value	Description
EXTID_TYPE_IPV4	1	IP Address
EXTID_TYPE_MAC	2	MAC Address
EXTID_TYPE_EXTNO	3	Station Number

2.4 Others

Constants Name	Value	Description
MAX_PHONENO	32	Telephone number maximum length
MAX_MSGPARA	512	Information structure data length
MAX_INFO	256	Information maximum length
DEFAULT_IPPA_PORT	16000	Default IPPA Server connection port

3 SWIPPA Client Lib API

SWIPPA Client Lib includes recording control interface and real-time monitoring interface.

3.1 Initialization and Unload

3.1.1 Description

Before the recording application calls recording interface or real-time monitoring interface, you must call SWIPinitialize to allocate client resource. Before the recording application stops, you had better call SWIPUninitialize to release off client resource.

3.1.2 Function Interface

1. SWIPinitialize

【Function Prototype】

void __Stdcall SWIPinitialize ();

【Parameter Declaration】

Parameter	Description
-	No Parameter Available

【Returned Value】

Returned Value	Description
-	No Returned Value

【Function Description】

Dynamic library initialization function allocates client resource.

【Notes】

Client resource can not be allocated repeatedly, so you should call SWIPUninitialize to release off client resource before calling SWIPinitialize again.

Only after allocating client resources can the recording application call recording interface or real-time monitoring interface.

【Correlation Function】

SWIPUninitialize.

2. SWIPUninitialize

【Function Prototype】

void __Stdcall SWIPUninitialize ();

【Parameter Declaration】

Parameter	Description
-	No Parameter

【Returned Value】

Returned Value	Description

Returned Value	Description
-	No Returned Value

【Function Description】

Dynamic library unload to releases off client resource.

【Notes】

After releasing off client resources, any other interface function(except SWIPinitialize), will fail.

【Correlation Function】

SWIPinitialize.

3. SWIPGetLastErrCode

【Function Prototype】

long __Stdcall SWIPGetLastErrCode ();

【Parameter Declaration】

Parameter	Description
-	No parameter

【Returned Value】

Returned Value	Description
= 0	No error code
< 0	Error code caused by last call

【Function Description】

Get the error code caused by last call.

【Notes】

This function can be called only after SWIPinitialize is called to allocate client resource.

【Correlation Function】

SWIPGetLastErrMsg.

4. SWIPGetLastErrMsg

【Function Prototype】

void __Stdcall SWIPGetLastErrMsg (char* *AErrBuf,* **long** *ABufLength***);**

【Parameter Declaration】

Parameter	Description
char* AErrBuf	Receive an error message buffer pointer
long ABufLength	Buffer length, error message up to 256 characters (include C string zero), and so the buffer length is recommended to be equal to or more than 256 characters

【Returned Value】

Returned Value	Description
-	No returned value

【Function Description】

Receive the error message caused by call failure.

【Notes】

The buffer size of AErrBuf is recommended to use 256 characters. If the error message is bigger than the buffer size, partial error message equal to the buffer size will be received.

This function can be called only after SWIPinitialize is called to allocate client resource.

【correlation Function】

SWIPGetLastErrCode.

3.2 Recording Control Interface

3.2.1 Description

When SWIPPA detects calling start or off-hook, SWIPPA sends EventBeginRecord event to record application (conversation), then the application calls SWIPStartRecord function to start recording.

When SWIPPA detects call ending or hang-up, SWIPPA sends EventEndRecord event to record application (conversation), then the application calls SWIPStopRecord function to end recording.

3.2.2 Function Interface

1. SWIPOpenConn

【Function Prototype】

Long __stdcall SWIPOpenConn(const char* *AHost*, **long** *APort***)**

【Parameter Declaration】

Parameter	Description
const char* AHost	IPPA Serve IP Address
long APort	Connection port，default is DEFAULT_IPPA_PORT, see“2 Constants Definition”

【Returned Value】

Returned Value	Description
> 0	Return to connect handle
<= 0	Call function fails, error codes see “5 Error Table”. Call “SWIPGetLastErrMsg” to return error message

【Function Description】

Open connection between client and SWIPPA server.

【Correlation Function】

SWIPCloseConn.

2. SWIPCloseConn

【Function Prototype】

void __stdcall SWIPCloseConn (long *AConnHandle***)**

【Parameter Declaration】

Parameter	Description
long AConnHandle	Handle returned by calling SWIPOpenConn function

【Returned Value】

Returned Value	Description
-	No returned value

【Function Description】

Disconnection between client and SWIPPA server.

【Declaration Function】

SWIPOpenConn.

3. SWIPIsConnOpened

【Function Prototype】

long __stdcall SWIPIsConnOpened(long *AConnHandle*)

【Parameter Declaration】

Parameter	Description
long AConnHandle	Handle returned by calling SWIPOpenConn function

【Returned Value】

Returned Value	Description
SWIPRet_success	Connection has been on
Others	Call function fails, error code see “5 Error Table”. Call SWIPGetLastErrMsg to return error message

【Function Description】

Check the connection with IPPA Server is disconnected or not. If yes, you must call SWIPCloseConn first, and then call SWIPOpenConn to connect.

【Correlation Function】

SWIPOpenConn.

4. SWIPSetChanInfo

【Function Prototype】

long __stdcall SWIPSetChanInfo(long *AConnHandle*, **ChannelInfo_Struct*** *AChanInfo*)

【Parameter Declaration】

Parameter	Description
long AConnHandle	Handle returned by calling SWIPOpenConn function
ChannelInfo_Struct* AChanInfo	Pointer to Channel information struct

【Returned Value】

Returned Value	Description
SWIPRet_success	Successfully set
Others	Call function fails, error code see “5 Error Table”. Call SWIPGetLastErrMsg to return error message

【Function Description】

Set property for recording channel.

5. SWIPGetEvent

【Function Prototype】

long __stdcall SWIPGetEvent(long *&AEventType*, **long** *&ADatasize*, **void*** *AData***)**

【Parameter Declaration】

Parameter	Description
long &AEventType	Event type: definition of event type, please see “2.1 Record Event Type”
long &ADatasize	Input is the buffer size of AData, while output is the length of returned data
void* AData	Input is the data buffer, output is the structure of returned related event. See “4.2 Record Event Structure”

【Returned Value】

Returned Value	Description
SWIPRet_success	Some data is returned successfully
Others	Call function fails, see “5 Error Table” for error code. Call SWIPGetLastErrMsg to return error message

【Function Description】

Obtain recording event, show channel status, and get incoming and outgoing phone number.

【Notes】
Getting the event of EventBeginRecord means to start phone call or off-hook.

Getting the event of EventEndRecord means to end phone call or hang up.

Getting the event of EventCaptchDTMF means to have received DTMF in conversation. To improve performance, VoIP phone shall not use In-band DTMF.

Getting the event of EventUpdateStatus means to update the conversation message, such as station number, caller number and call direction.

6. SWIPStartRecord

【Function Prototype】

long __stdcall SWIPStartRecord(long *AConnHandle,* **long** *AChID,* **long** *ACodecFormat,* **const char*** *AFilename,* **CBStreamWrite *** *AStreamWrite,* **void*** *ACBParam*);

【Parameter Declaration】

Parameter	Description
long AConnHandle	Handle returned by calling SWIPOpenConn function
long AChID	Channel number
long ACodecFormat	Output audio format
const char* AFilePath	Output the server audio file path, if passing NULL to this parameter, it means no audio file will be generated
CBStreamWrite *AStreamWrite	Output the callback function point of the audio stream, if passing NULL to this parameter, it means no audio file will be generated
void* ACBParam	Callback parameter, when you call the AStreamWrite callback function, the original value of ACBParam will be delivered to AStreamWrite. (Optional)

【Returned Value】

Returned Value	Description
SWIPRet_success	Success
Others	Fail

【Function Description】

Arrange one channel to start recording. The output record can be audio file or stream, and can be both of them.

【Notes】

If you call SWIPStartRecord function, AFilePath and AStreamWrite parameter can not be set NULL simultaneously. After calling SWIPOpenConn function, you can call SWIPStartRecord function anytime, like before conversation, during conversation or finish conversation.

If you call SWIPStartRecord before starting conversation, the recorded audio is silence.

7. SWIPStopRecord

【Function Prototype】

Long __stdcall SWIPStopRecord(long *AConnHandle,* **long** *AChID*)

【Parameter Declaration】

Parameter	Description

Parameter	Description
long AConnHandle	Handle returned by calling SWIPOpenConn function
long AChID	Channel Number

【Returned Value】

Returned Value	Description
SWIPRet_success	Success
Others	Failure

【Function Description】

Set one channel to stop recording.

【Notes】

After calling SWIPStartRecord function, you can call SWIPStopRecord function anytime, like during conversation or finish conversation

If the function has not been called after the end of conversation, multiple conversations can be recorded in one audio file.

3.2.3 Voice Stream Output Callback Interface

【Function Prototype】

typedef void (__stdcall *CBStreamWrite)(long *AChID,* **const void*** *AData,***long** *ASize,* **void*** *ACBParam***);**

【Parameter Declaration】

Parameter	Description
long AChID	Channel ID
const void* AData	Voice stream data
long ASize	Length of voice stream data
void* ACBParam	Callback parameter, made when the application calls SWIPStartRecord function

【Returned Value】

Returned Value	Description
-	No returned value

【Function Description】

Callback function of recording voice stream output

【Correlation Function】

SWIPStartRecord.

4 Structure Definition

4.1 Channel Information Structure

```
typedef struct ChannelInfo_Struct
{
  DWORD      dwChID;                  // Logic channel number
  DWORD      dwApplicationID;         //Application ID, retained by system
  long       dwExtIDType;             //Extension Identification Type
  char       szExtID[MAX_PHONENO];    //It can be IP address, MAC address or
                                      //extension number
}ChannelInfo_Struct;
```

Notice:

dwChID is the channel identification number in the entire system, and can not used twice in the same system. dwChID number is between 0 and 2048.

4.2 Record Event Structure

There are five types of events, each of which belongs to an event structure. Please see the following table:

Serial No.	Event	Event Structure	Trigger Time
1	EventReportInfo	EventReportInfo_Struct	System retained System information, like warning message
2	EventBeginRecord	EventBeginRecord_Struct	Start recording or off-hook
3	EventEndRecord	EventEndRecord_Struct	Recording ends or headset hang-up
4	EventCaptureDTMF	EventCaptureDTMF_Struct	Retrieve DTMF
5	EventUpdateStatus	EventUpdateStatus_Struct	Retrieve incoming number or outgoing number

1. EventReportInfo_Struct

typedef struct EventReportInfo_Struct

```
{
DWORD     dwSize;                    // Structure dimension
DWORD     dwInfoType;                //Information type
DWORD     dwChID;                    // Logic channel number
char      szInfo[MAX_INFO];          // Information
} EventReportInfo_Struct;
```

 Notice：

Please remain the above structure in the system.

2. EventBeginRecord_Struct

```
typedef struct EventBeginRecord_Struct
{
DWORD     dwSize;                              // Structure dimension
  DWORD    dwChID;                             // Logic channel number
  DWORD    dwCallDirection;                    // 1=Calling, 0=Called
 long         dwStoreCodec;                    // Compression format
  char         szCallerNo[MAX_PHONENO];        // Calling number
  char         szCalledNo[MAX_PHONENO];        // Called number
 char         szCallRefID[MAX_CALLRefID];     // CallRefID
}EventBeginRecord_Struct;
```

3. EventEndRecord_Struct

```
typedef struct EventEndRecord_Struct
{
  DWORD     dwSize;                                  // Structure dimension
  DWORD     dwChID;                                  // Logic channel number
  DWORD     dwTalkDurationSeconds;                   // Talking time
  DWORD     dwStatus;          // bit1=pick up, bit2=setup, bit3=ring, bit4=DTMF,
                               // bit5=talking, bit6=receive the other part number, bit7=hang
up, bit8=function key
}EventEndRecord_Struct;s
```

Notice：

dwTalkDurationSeconds is the conversation time, but not the recording time. The recording time (time difference after calling SWIPStopRecord and SWIPStartRecord function) is counted by application itself.

4. EventCaptureDTMF_Struct

```
typedef struct EventCaptureDTMF_Struct
{
  DWORD     dwSize;          // Structure dimension
  DWORD     dwChID;          // Logic channel number
  char      chDTMF;          // '1' '2' '3' '4' '5' '6' '7'8' '9' '0' '*' '#'
}EventCaptureDTMF_Struct;
```

5. EventUpdateStatus_Struct

```
typedef struct EventUpdateStatus_Struct
{
  DWORD dwSize;                 // Structure dimension
  DWORD dwChID;                 // Logic channel number
  DWORD dwCallDirection;        // 1=Calling（Station calls to other party）,0=Called（Other party calls to station）
  char szOtherPartyPhoneNo[MAX_PHONENO];    // Other party's phone number
}EventUpdateStatus_Struct;
```

5 Error Table

Constant Name	Error Code	Description
SWIPSWIPRet_Success	0	Success
SWIPRet_Error	-2001	Unknown error
SWIPRet_Params_Error	-2002	Parameter error
SWIPRet_Func_Except	-2003	Function execution extension
SWIPRet_Func_TimeOut	-2004	Function execution timeout
SWIPRet_Conn_Failed	-2005	Connection failure
SWIPRet_Inited	-2006	Initialized
SWIPRet_Buf_Full	-2007	Buffer full
SWIPRet_Login_Failed	-2008	Login failure
SWIPRet_Connect_NotOpened	-2010	Connection not opened

Constant Name	Error Code	Description
SWIPRet_UnInited	-2011	Uninitialized
SWIPRet_MallocFailed	-2012	Asynchronous return
SWIPRet_NOData	-2013	No data
SWIPRet_CHFull	-2014	Authorized channel is full
SWIPRet_RecordStarted	-2015	Channel has started recording
SWIPRet_CHUnExist	-2016	Channel does not exist
SWIPRet_INVALIDVCOC	-2017	Unavailable VCOC or VCOC resource is full