聯陽半導體
ITE Tech. Inc.

iTE 可視對講系統

Leaf mediastreamer 架構說明文件

V0.5

ITE TECH. INC.

修訂日期	修訂說明	頁次
2016/06/23	初建版本 V0.1	
2016/07/15	V0.5	

目錄

1. 前言

1.1 編寫目的

本文件的主要目的是讓不是使用Linphone系統的客戶可以很容易整合到iTE對講系統，提供影像和語音相關API之操作及使用說明，另外也包括通話拍照和留影留言等相關應用的使用說明。
如果使用者是架構在Linphone系統或是使用SIP/RTP通訊, 可以參考 “iTE 可視對講系統-Streaming架構說明文件”。

1.2 適用範圍

ITE 相關平台。

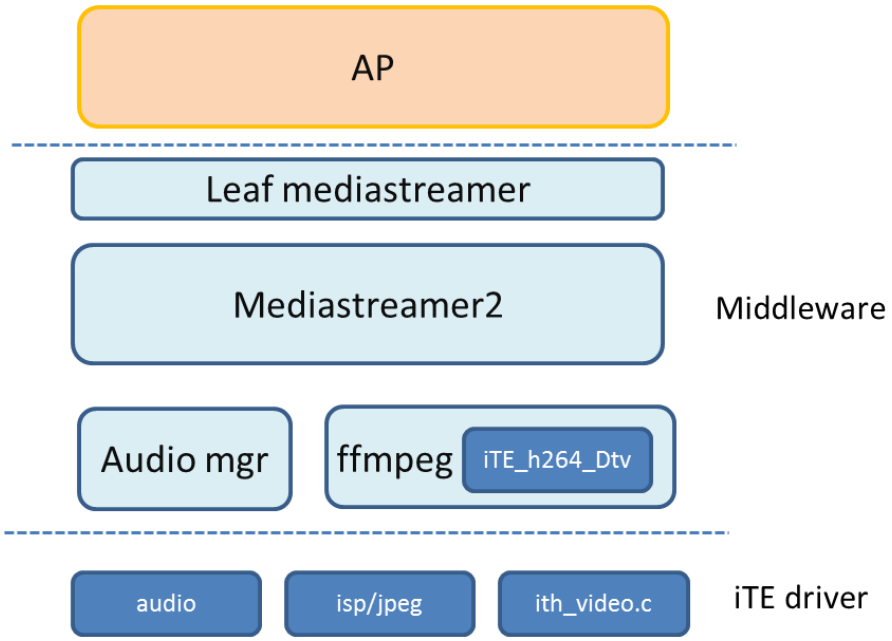
1.3 適用人員

軟體應用程式開發者，建議先了解mediastreamer2基本觀念。

2. Leaf mediastreamer

Leaf mediastreamer 是架構於mediastreamer2之上，提供可視對講系統語音及影像相關應用之API，使用者只要調用相關API及修改相關代碼，就可以很容易的實現對講系統音視頻的應用。

下圖為Leaf mediastreamer架構



Leaf mediastreamer 是介於AP層和mediastrem2之間輕量級的介面，提供音視頻相關API，
Leaf mediastreamer主要處理相關模組的初始化、A/V filter的建立、執行緒的建立及底層驅動程式的控制等，使用者並不需要了解細節，只需要調用相關API使用，API 詳細使用方式請參考第三章所述。除了調用相關API外，使用者需要根據使用者平台的網路協議及A/V封裝格式修改msudp.c，詳細作法請參考第四章所述。

mediastrem2相關資料可參考 http://www.linphone.org/technical-corner/mediastreamer2/overview，另外也可參考“iTE 可視對講系統-Streaming架構說明文件”。

3. Leaf mediastreamer API

Leaf mediastreamer 相關API實作代碼請參考sdk/share/leaf/leaf_mediastream.c

3.1 Leaf mesiastreamer Initialization

```
LeafCall* leaf_init(void);
```

輸入參數 : 無
回傳參數 : LeafCall 指標

在使用Leaf mediastreamer時需要先呼叫此函式完成初始化動作，並回傳初始化資料。

3.2 通話 API

3.2.1 Video stream Initialization

```
void leaf_init_video_streams (LeafCall *call , unsigned short port);
```

輸入參數 : LeafCall 指標及UDP封包接收阜號
回傳參數 : 無

初始化video stream。

3.2.2 Video start stream

```
void leaf_start_video_stream(LeafCall *call, const char *addr, int port);
```

輸入參數 : LeafCall 指標、封包來源位址及UDP封包接收阜號
回傳參數 : 無

開始接收及處理影像網路封包。

3.2.3 Video stop stream

```
void leaf_stop_media_streams(LeafCall *call);
```

輸入參數 : LeafCall 指標

回傳參數 : 無

停止接收及處理影像網路封包。

3.2.4 Audio stream Initialization

```
void leaf_init_audio_streams (LeafCall *call, unsigned short port);
```

輸入參數 : LeafCall 指標及UDP封包接收阜號
回傳參數 : 無

初始化audio stream。

3.2.5 Audio start stream

```
void leaf_start_audio_stream(LeafCall *call, const char *addr, int port);
```

輸入參數 : LeafCall 指標、UDP封包來源位址及UDP封包接收阜號
回傳參數 : 無

開始接收及處理語音網路封包。

3.2.6 Audio stop stream

```
void leaf_stop_media_streams(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止接收及處理語音網路封包。

3.2.7 Audio mic/spk level set

```
void leaf_set_play_level(LeafCall *call,int level);
void leaf_set_rec_level(LeafCall *call,int level);
```

輸入參數 : LeafCall 指標及Level(0~100)
回傳參數 : 無

設置通話音量大小。

設置麥克風收音大小。

3.3 鈴聲 API

3.3.1 Audio start play sound

```
void leaf_start_sound_play(LeafCall *call, char *file, int sec,SoundPlayCallback func);
```

輸入參數 : LeafCall 指標 、file path、撥放長度sec(秒)及回調函數(Callback function)不使用設為NUL
回傳參數 : 無

撥放.wav檔(支援ulaw、alaw編碼)，sec秒後自動停止撥放，sec = 0重複撥放，sec < 0 檔案結束停止。
撥放.svm檔(speex壓縮檔)

3.3.2 Audio stop play sound

```
void leaf_stop_sound_play(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止撥放.wav檔。

3.3.3 Audio ring level set

```
void leaf_set_ring_level(LeafCall *call,int level);
```

輸入參數 : LeafCall 指標及Level(0~100)
回傳參數 : 無

設置鈴聲音量大小。

3.4 留影留言 錄製API

3.4.1 Start video memo record

```
int leaf_start_video_memo_record(LeafCall *call,  char *file);
```

輸入參數 : LeafCall 指標及錄製檔案路徑和名稱
回傳參數 : 1 代表成功
其他值代表失敗

開始錄製留影留言。

3.4.2 Stop video memo record

```
void leaf_stop_video_memo_record(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止錄製留影留言。

3.4.3 Check video memo is recording or not

```
bool leaf_video_memo_is_recording(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 1 代表正在進行錄影
0 代表沒有影片正在錄影

檢查是否正在進行錄影

3.4.4 Take video snapshot

```
int leaf_take_video_snapshot(LeafCall *call, char *file, FileWriterCallback func );
```

輸入參數 : LeafCall 指標、錄製檔案路徑和名稱和 回調函數(callback function)
回傳參數 : 1 代表成功
其他值代表失敗

通話拍照

如果輸入參數*func*為NULL，表示同步處理，此API會執行完拍照和寫檔後才會返回AP。
如果輸入參數*func*不為NULL，表示非同步處理，此API設定完相關參數即返回AP，再經由回調函數通知上層AP，拍照和寫檔已完成。

3.4.5 Start voice memo record

```
void leaf_start_voice_memo_record(LeafCall *call, char *file);
```

輸入參數 : LeafCall 指標及錄製檔案路徑和名稱
回傳參數 : 無

開始錄音
經語音壓縮(speex壓縮)非wav檔(pcm)。副檔名為.svm
錄製wav檔可修改 sdk\share\mediastreamer2\audiostream.c
voice_memo_start_record()。
int encode_type = 0; //0:unknow(speex) 1:PCM 6:a-law 7:u-law
預設0:speex壓縮，1:不壓縮，6:a-law，7:u-law

3.4.6 Stop voice memo record

```
void leaf_stop_voice _memo_record(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止錄音。

3.5 留影留言播放API

3.5.1 Start video memo playback

```
int leaf_start_video_memo_playback(LeafCall *call,  char *file);
```

輸入參數 : LeafCall 指標及錄製檔案路徑和名稱

回傳參數 : 1 代表成功
其他值代表失敗

播放留影留言。

3.5.2 Stop video memo playback

```
void leaf_stop_video_memo_playback(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止播放留影留言。

3.5.3 Pause video memo playback

```
void leaf_pause_video_memo_playback();
```

輸入參數 : 無
回傳參數 : 無

當影片正在播放時，呼叫此API會進入暫停狀態
當影片暫停時，呼叫此API會回到播放狀態

3.5.4 Check video memo play finished or not

```
bool leaf_video_memo_is_play_finished(void);
```

輸入參數 : 無
回傳參數 : 1 代表影片播放結束
0 代表影片尚未播放結束

檢查留影留言是否已經播放結束。

3.5.5 Get video memo current time

```
int leaf_get_video_memo_current_time();
```

輸入參數 : 無
回傳參數 : -1代表無法取得時間
其他值代表影片目前播放時間，單位為秒

取得影片目前播放時間

3.5.6 Get video memo total time

```
int leaf_get_video_memo_total_time();
```

輸入參數 : 無
回傳參數 : -1代表無法取得時間
其他值代表影片總時長，單位為秒

取得影片總時長

3.5.7 Get audio total time

```
double  leaf_get_wav_time(*char file);
```

輸入參數 : 檔案路徑
回傳參數 :wav檔總時長，單位為秒。

取得wav檔總時長。

3.5.8 Show snapshot

```
int leaf_show_snapshot(LeafCall *call,  int width,  int height,  char *file);
```

輸入參數 : LeafCall 指標及錄製檔案路徑和名稱
回傳參數 : 1 代表成功
其他值代表失敗

播放JPEG檔案。

3.5.9 Start voice memo playback

~~void leaf_start_voice _memo_playback(LeafCall *call, char *file);~~
void leaf_start_sound_play(LeafCall *call, char *file, int sec,SoundPlayCallback func);

輸入參數 : LeafCall 指標 、file path、撥放長度sec(秒)及回調函數(Callback function)不使用設為NUL
回傳參數 : 無
撥放錄音檔，sec秒後自動停止撥放，sec = 0重複撥放，sec < 0 檔案結束停止。

3.5.10 Stop voice memo playback

~~void leaf_stop_voice _memo_playback(LeafCall *call);~~
void leaf_stop_sound_play(LeafCall *call);

輸入參數 : LeafCall 指標
回傳參數 : 無

停止撥放錄音檔。

3.5.11 Check audio play finished or not

bool leaf_audio_sound_is_playing(void);

輸入參數 : 無
回傳參數 : 1 代表音檔播放結束
0 代表音檔尚未播放結束

檢查音檔是否已經播放結束。

3.5.12 Check audio memo finished or not

bool leaf_audio_memo_is_recording(void);

輸入參數 : 無
回傳參數 : 1 代表音檔錄音結束
0 代表音檔尚未錄音結束

檢查錄音是否已經結束。

3.6 IP CAM API

3.6.1 IPCam start stream

```
void leaf_start_ipcam_stream(LeafCall *call, const char *addr, int port);
```

輸入參數 : LeafCall 指標、IPCAM封包來源位址及IPCAM封包接收阜號
若該IPCAM有設定帳號密碼，則封包來源位址格式必須為 “id:pwd@xxx.xxx.xxx.xxx”
回傳參數 : 無

開始接收及處理IPCAM網路封包。

3.6.2 IPCam stop stream

```
void leaf_stop_ipcam_stream(LeafCall *call);
```

輸入參數 : LeafCall 指標
回傳參數 : 無

停止接收及處理IPCAM網路封包。

4. 簡單通話範例

使用者除了調用Leaf mediastreamer API，還有幾個地方需要使用者自行修改相關代碼，在說明如何修改相關代碼之前，我們先來看iTE SDK提供的範例，然後藉由範例的說明，讓使用者更清楚如何修改。

4.1 範例說明

iTE SDK 提供的簡單的通話範例，以UDP封包傳送語音和視頻資料，傳送端(大廳機)開機後會自動透過UDP封包傳送語音和視頻資料，視頻為單向傳輸，語音則可以相向傳輸。

傳送端(大廳機)的專案為doorbell_lobby_ex，本專案只是單純開機後自動傳送語音與視頻UDP封包。
接收端(室內機) 的專案為doorbell_indoor_ex，相關代碼可以參考

project/doorbell_indoor_ex/layer_called.c
處理UI相關操作及Leaf mediastream API的調用
目前在layer_called.c有定義以下三種define

```
#define TEST_CALL_INCOMING          1       //for testing call incoming
#define TEST_IPCAM_STREAMING        0       //for testing ipcam streaming
#define TEST_PLAYBACK_VIDEO         0       //for testing playback video
```

分別表示一般通話測試、IPCAM測試、 Video Playback播放測試
要測試哪種功能必須自行修改define的值

sdk/share/leaf/leaf_mediastream.c
Leaf mediastream API的實作

sdk/share/me/mediastream_udp .c
mediastrem2 語音視頻相關filter的建立及進入點

sdk/share/mediastreamer2/msudp.c
接收網路封包和處理A/V封裝格式
目前在msdup.c有FILE_MODE 的定義
當#define FILE_MODE 1 則從USB隨身碟中讀取test.264檔案播放
當#define FILE_MODE 0 則實際收UDP網路封包

4.2 視頻的顯示

藉由第三節API介紹，使用者應該會發現相關API並沒有任何參數和最後視頻顯示位置和大小相關的資訊，這就是使用者需要自行修改的第一個地方。首先你需要知道你們設計的UI layer,哪個物件是最後要顯示視頻的區域，以我們提供的範例layer_called.c中 *calledRemoteBackground*物件是視頻顯示的區域和位置，知道哪個物件後，在 layer_called.c中Enter function中加上下面代碼，這些代碼主要是設定視頻顯示的位置和大小

```
ituWidgetGetGlobalPosition(calledRemoteBackground, &x, &y);
width = ituWidgetGetWidth(calledRemoteBackground);
height = ituWidgetGetHeight(calledRemoteBackground);
itv_set_video_window(x, y, width, height);
```

然後你需要為這個物件掛上回調函數(callback function)

```
calledRemoteBackground = ituSceneFindWidget(&theScene, "calledRemoteBackground");
assert(calledRemoteBackground);
```

增加下面代碼

```
ituWidgetSetDraw(calledRemoteBackground, calledRemoteBackgroundDraw);
```

這樣就把回調函數安排給*calledRemoteBackground*物件，當系統畫到此物件是就會調用 *calledRemoteBackgroundDraw()*

最後就是補上回調函數，函數作用就是將解碼出的視頻畫到螢幕上。

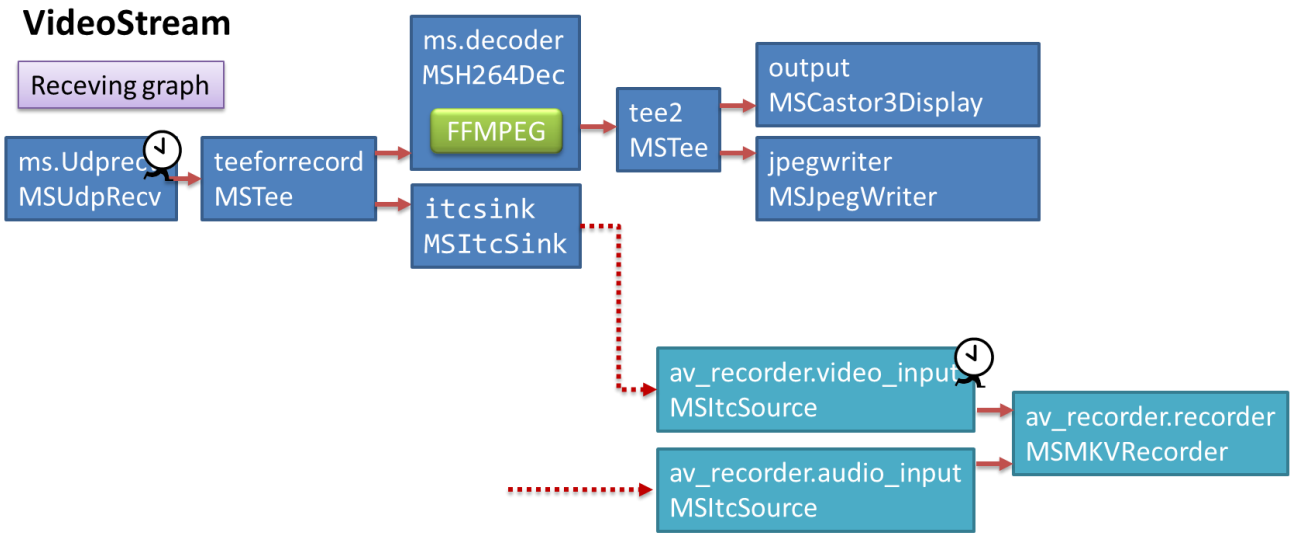
```
static void calledRemoteBackgroundDraw(ITUWidget* widget, ITUSurface* dest, int x, int y, uint8_t alpha)
{
    int destx, desty;
    ITURectangle* rect = (ITURectangle*) &widget->rect;

    destx = rect->x + x;
    desty = rect->y + y;

    ituDrawVideoSurface(dest, destx, desty, rect->width, rect->height);
}
```

4.3 video stream

使用者第二個需要修改的就是需要根據使用者平台修改網路相關協議及A/V封裝格式，接下來章節將會介紹相關觀念及如何修改。

首先來了解video stream filter架構，使用者調用*leaf_start_video_stream()*後，video 相關filter就會自動建立，並等待傳送端傳送資料，下圖MSUdpRecv就是接收網路封包和處理A/V封裝格式的filter，MSUdpRecv相關代碼請參考sdk/share/mediastreamer2/msudp.c。

基於mediastreamer2架構所建立的video stream graph如下圖，包含視頻對講、通話拍照、留影留言。



相關filter模組說明如下：

MSUdpRecv : 處理網路封包的模組
MSTee : 複製video stream的模組
MSH264Dec : 處理H264硬體解碼的模組
MSCastor3Display : 處理將Decode出來的影像資料顯示到LCD螢幕上
MSJpegWriter : 處理JPEG硬件編碼的模組
MSMKVRecorder : 處理儲存AV stream到在地儲存裝置

4.4 msudp.c介紹

msudp.c是接收網路封包和處理A/V封裝格式的filter相關代碼，iTE SDK提供兩個範例

範例一：
為了讓客戶可以快速驗證H.264 stream相容性及開發UI，iTE SDK提供開機後會自動讀取USB隨身碟內的H.264 Stream播放的範例，使用者只需將H.264 NAL stream檔名改成test.264，放入USB隨身碟中。
使用者使用此範例需要修改 msdup.c的定義 #define FILE_MODE 1

範例二：
是以UDP協議接收A/V資料，使用者使用此範例需要修改 msdup.c的定義 #define FILE_MODE 0
接收影像封包流程如下：
當大廳機播打室內機時，室內機AP會調用 *leaf_start_video_stream()*啟動接收影像封包

leaf_start_video_stream()	leaf_mediastream.c
⇩	
video_stream_udp_start()	mediastream_udp.c
⇩	
udp_receiver_set_para()	msdup.c

在*udp_receiver_set_para()*函式中最重要的是
(a). 建立網路socket，接收網路封包
(b). 建立video 執行緒 *video_receive_thread()*，持續接收socket送來的資料，解析封包格式，並將完整的NAL frame 往下一級filter傳送

使用者只需修改*video_receive_thread()*內相關代碼，將影像封裝格式轉成H.264 byte stream format。
iTE提供的傳輸範例，其中影像格式為
(a). video payload最大為10*1024

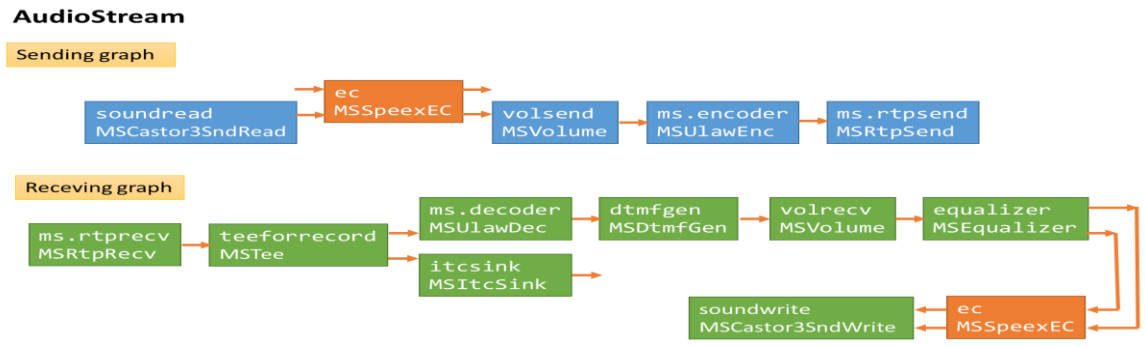
(b). video package format :
start_code [4]+ framesize[4] + H.264 byte stream format
start_code[4] = {0x00, 0x00, 0x01, 0xfc};
framesize[4] 是指H.264 byte stream format長度，不包括start_code和 frame size欄位

video_receive_thread() 會持續調用recvfrom()從socket接收資料，再經過重新封裝後將一幀完整的video 資料透過調用ms_queue_put()將資料送到下一級的filter。

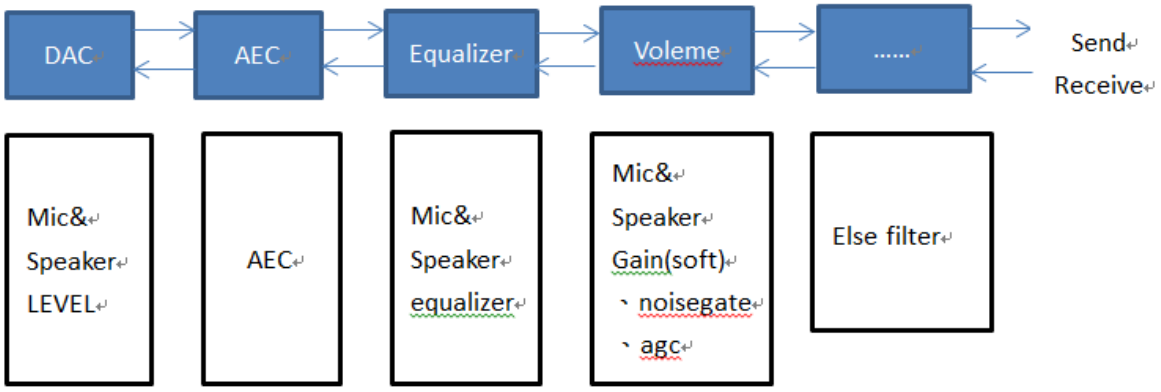
4.5 Audio stream

Audio stream filter架構，使用者調用leaf_start_audio _stream()後，對講相關filter就會自動建立，並等待傳送端傳送資料。Filter 建立過程請參考 sdk/share/me/mediastream_udp .c audio_stream_udp_start_full()，建立之後對講所需的參數由audio_stream_post_configure()下至filter。

在audio_stream_udp_start_full()這個函式中所代入的第7個參數和相對應的audio_stream_udp_stop()函式中所代入的第2個參數, 將決定這次建立的audio stream可以進行雙向錄影或是單向錄影，若該參數為AudioFromSoundRead，則進行通話錄影時，可以同時錄下發送端和本機端的聲音，若該參數為AudioFromUdpRecv，則進行通話錄影時，只能錄下來自發送端的聲音，目前預設為AudioFromSoundRead，開發者可依需求自行調適。



以下為主要對講filter對聲音執行之DSP功能



當通話過程中雙方聲音互相傳播，聲音的大小調整、回音消除、等化器、語音壓縮、混入聲音等可藉由audiostream filter模組達成，詳情請查閱。

聲音大小調整　:/sdk/share/mediastreamer2/msvolume.c
等化器　:/sdk/share/mediastreamer2/equalizer.c
混入聲音　:/sdk/share/mediastreamer2/mixvoice.c
語音壓縮　:/sdk/share/mediastreamer2/ulaw.c、alaw.c
回音消除　:/sdk/share/mediastreamer2/sbc_aec.c

語音的部分使用者也一樣只需修改audio*_receive_thread()*內相關代碼，將語音資料從網路封包取出。
iTE提供的語音傳輸範例，格式為每筆固定128byte大小作傳送和接收,並無特別封裝, 使用者可依實際狀況自行做修改。