/****************************************************************************/
This application is very basic implementation of various features like audio call, conference call and recording of audio call including conference call.People can use this for their own application and enhance its features according to their own requirements

/___________________________________________________________/

## MAIN API USED FOR AUDIO CALL

1. pj_status_t pjsua_call_make_call (pjsua_acc_id acc_id,
   const pj_str_t *dst_uri,
   const pjsua_call_setting *opt,
   void *user_data,
   const pjsua_msg_data *msg_data,
   pjsua_call_id *p_call_id)

Make outgoing call to the specified URI using the specified account.

Parameters->

| | |
|---|---|
| acc_id | The account to be used. |
| dst_uri | URI to be put in the To header (normally is the same as the target URI). |
| opt | Optional call setting. This should be initialized using pjsua_call_setting_default(). |
| user_data | Arbitrary user data to be attached to the call, and can be retrieved later. |
| msg_data | Optional headers etc to be added to outgoing INVITE request, or NULL if no custom header is desired. |
| p_call_id | Pointer to receive call identification. |

Returns->

| | |
|---|---|
| PJ_SUCCESS | on success, or the appropriate error code. |

2. pj_status_t pjsua_call_answer(pjsua_call_id call_id,
   unsigned code,
   const pj_str_t * reason,
   const pjsua_msg_data * msg_data
   )

Send response to incoming INVITE request. Depending on the status code specified as parameter, this function may send provisional response, establish the call, or terminate the call. See also pjsua_call_answer2().

Parameters->

| | |
|---|---|
| call_id | Incoming call identification. |

code Status code, (100-699). // 200 for answering call
reason Optional reason phrase. If NULL, default text will be used.
msg_data Optional list of headers etc to be added to outgoing response message. Note that this message data will be persistent in all next answers/responses for this INVITE request.
Returns->
PJ_SUCCESS on success, or the appropriate error code.

3. void pjsua_call_hangup_all (void)

Terminate all calls. This will initiate pjsua_call_hangup() for all currently active calls.

/___________________________________________________________________/

## Audio Conference call

When multiple call happens, then in callback function **on_call_media_state** , conf_slot of each all gets allocated a sequential number stating from 0, suppose you cann USER1, then in call info ,conf_slot will be 1 and self conf_slot will be 0.
and you make another call(USER2) while ongoing 1st call, conf_slot for him will be 2.

All you need to do is connect these slots with each other for merging call with each other using following API bidirectional.

**pjsua_conf_connect(pjsua_conf_port_id source, pjsua_conf_port_id sink);**

**Example ->** there are 3 users , then combination of all voice transfers will be 3C2 = 3 combination. Conf_slot will be
self-> 0
User1-> 1, user2-> 2 ( I am a programmer, my counting starts from 0 …HaHa)

A) Connect 0 -> 1
Connect 1 -> 0

B) Connect 2->0
Connect 0->2

C) Connect 1->2
Connect2 ->1

/___________________________________________________________________/

## Recording conference / audio call

API used for recording purpose

1. For creating recorder

```
PJ_DECL(pj_status_t)      pjsua_recorder_create(const pj_str_t *filename,
                                                 unsigned enc_type,
                                                 void *enc_param,
                                                 pj_ssize_t max_size,
                                                 unsigned options,
                                                 pjsua_recorder_id *p_id);
```

2 for getting port number assocaiated to recorder

```
PJ_DECL(pjsua_conf_port_id) pjsua_recorder_get_conf_port(pjsua_recorder_id id);.
```

For recording voice of simple and conf call, connect all conf_slot to recorder port using same API that we used in conf call.

```
PJ_DECL(pj_status_t) pjsua_conf_connect(pjsua_conf_port_id source,
                                        pjsua_conf_port_id sink);
```

/************************************************************
************************************************************/

**Steps to use my application for above 3 scenarios**.

1. simple audio call,

run application(go inside release folder, run SampleVoiceCall.exe file) on both application (USER1 and USER2)

IP of user1 -> 192.148.34.45

IP of user2-> 193.148.34.44;

Put string “sip:100@192.148.34.44:5060” in user1’s application. You will get incoming call info in user2’s application.

2. In conf call , suppose IP of USER3->193.148.34.43,

Put string “sip:100@192.148.34.43:5060” in user1’s application. You will get incoming call info in user3’s application. Now all 3 users are connected to each other.5060 is port number that is being used in calling

3. If you wanna record conf call , press “record ” button , file will be created as “recording.wav”. you can use any media application for playing it back such as VLC player.

# PEACE GUYS