/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

This application is very basic implementation of various features like audio call m 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)

1. Connect 0 -> 1

Connect 1 -> 0

1. Connect 2->0

Connect 0->2

1. Connect 1->2

Connect2 ->1

/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

**Recording conf / 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.

1. 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

1. 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**