**一 PC浏览器接口说明**

1 请使用Google Chrome浏览器（或者IE，要安装插件）

下载链接：<http://pan.baidu.com/share/link?shareid=500175&uk=772255636>

2使用方法

* 将lw2fzd.min.js文件包含在页面中，请使用utf-8编码方式，例如

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

* Livecom公司提供的demo文件为lw2fzd.min.js，放到webserver相应目录下，比如script目录下，在服务器页面中，包含lw2fzd.min.js文件

<script src="script/lw2fzd.min.js" type="text/javascript"></script>

* 正常情况，此时浏览器会提示是否允许麦克风，按照窗口的提示，点击浏览器上面的允许按钮。

3 接口说明如下：

lwStartVoip(cid, phone, opts)

其中cid为主叫标识，phone为被叫号码。

opts 是一个对象数组 （可选参数）

{

userclass: // “registered” or “test” 用户类别：注册或者测试用户

ringing: function(){ } // 被叫振铃的回调事件

talking:function(){ } // 被叫接通时响应事件

peerhangup: function(){ } // 对端挂断时响应的回调事件

}

lwStopVoip()

挂断

例如：

lwStartVoip ('13800000000','13800000001', {

userclass: “registered”,

ringing: function(){

console.log(ringing,handle');

},

talking:function(){

console.log(talking,handle');

},

'peerhangup':function(){

console.log('peerhangup,handle');

},

qos\_report:function(data){

}

})

其中qos\_report函数参数data的结构如下：

{pr:968,pl:0,lrate,0.0,jitter:18,jquality:"good",rtt:1,rttquality:"perfect"}

qos\_report回调调用的时候，请显示以上信息。

**二 手机客户端接口说明**

=================================SDK导入=================================

1.eclipse新建android工程后会自动生成一个libs的文件夹

2.在libs文件夹上点击右键，选择import，将webrtc.jar导入

3.在libs文件夹上点击右键，选择Build Path->Configure Build Path

4.在弹出的界面上左侧栏选择Java Build Path

5.在右侧的菜单中选择Libraries

6.选择Add JARS，添加webrtc.jar

7.将armeabi和armeabi-v7a文件夹拖入libs

8.在AndroidManifest.xml文件中增加如下三句话

<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.INTERNET" />

=================================SDK API=================================

提供了一个类CallMaker和一个interface UserCallback

import org.webrtc.videoengineapp.CallMaker;

import org.webrtc.videoengineapp.UserCallback;

UserCallback定义了三个回调函数 ie.

public class MyCallback implements UserCallback

{

public void ring(){};

public void talking(){};

public void hangup(){};

}

三个回调分别是在振铃，接通，对方挂机时被调用

CallMaker类提供如下API

CallMaker(Context ctxt,UserCallback api)

void release()

void setVolumeLevel(int vol)

void setLoudSpeaker(boolean bool)

String startCall(String phone,String userclass,String selfPhone)

String stopCall()

=============================================

CallMaker(Context ctxt,UserCallback api)

void release()

负责构造CallMaker类和释放，如果构造或者释放失败，会抛出RuntimeException

=============================================

设置本端听筒音量，默认值是204

void setVolumeLevel(int vol)

设置是否使用免提，默认值是不使用

void setLoudSpeaker(boolean bool)

=============================================

String startCall(String phone,String userclass,String selfPhone)

String stopCall()

发起呼叫和停止呼叫，phone为呼叫号码，selfPhone为用户号码，userclass为用户类型，目前是两个值有效分别是"registered","unregistered"

startCall和stopCall成功返回"ok"，失败则返回原因

此外，startCall和stopCall都调用了发送ajax请求的http接口，android的权限限制，要求UI类中不能调用这种接口，防止被吊死，所以如果想在UI类中调用

需要另起一个线程来执行startCall和stopCall

=============================================

demo代码

package com.example.test;

import android.os.Bundle;

import android.widget.Button;

import android.view.View;

import android.widget.EditText;

import android.app.Activity;

import android.view.Menu;

import android.os.AsyncTask;

import org.webrtc.videoengineapp.CallMaker;

import org.webrtc.videoengineapp.UserCallback;

public class MainActivity extends Activity implements View.OnClickListener,UserCallback

{

private CallMaker call = null;

private Button btCall;

private Button btStop;

private EditText txtPhone;

public void ring(){};

public void talking(){};

public void hangup(){};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

btCall = (Button) findViewById(R.id.call);

btStop = (Button) findViewById(R.id.stop);

txtPhone = (EditText) findViewById(R.id.phone);

btCall.setOnClickListener(this);

btStop.setOnClickListener(this);

call = new CallMaker(this,this);

}

@Override

protected void onDestroy() {

call.release();

call = null;

super.onDestroy();

}

public void onClick(View v){

switch (v.getId()){

case R.id.call:

new CallStarter().execute("");

break;

case R.id.stop:

new CallStoper().execute("");

break;

default:

break;

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

public class CallStarter extends AsyncTask<String, Integer, String>{

@Override

protected String doInBackground(String... data){

return call.startCall(txtPhone.getText().toString(),"registered","11111111111");

}

@Override

protected void onPostExecute(String rt){

super.onPostExecute(rt);

}

}

public class CallStoper extends AsyncTask<String, Integer, String>{

@Override

protected String doInBackground(String... data){

return call.stopCall();

}

@Override

protected void onPostExecute(String rt){

super.onPostExecute(rt);

}

}

}

一、p2p voip语音接口说明

1 lwforfzdVoices.get\_free\_room(success\_cb,fail\_cb);

success\_cb/fail\_cb回调函数原型如下：

success\_cb(data); //data: {room:room}

fail\_cb(err);

2 lwforfzdVoices.enter\_room: function(paras){

//paras:{uuid:uuid,room:room,cbs:{waiting:w, incoming:i,media\_ok:m,media\_fail:m1,peerleave:p,roomrelease:r,err:e, request\_talking:r}}

cbs里面的成员是回调函数：

waiting: 房间有人，需要排队

incoming：表示对端媒体准备好

media\_ok: 本端媒体准备好

media\_fail:本端媒体失败，比如拒绝。

peerleave: 对方离开

release：用户已释放

request\_talking(data): 通知客服和用户请求通话,用户或客服需要选择是否接听。

err：各种错误

以上回调函数的参数不用关心。

3 lwforfzdVoices.leave\_room();

4 lwforfzdVoices. waiting2talk();

排队结束用户点击界面允许通话

5 客服拒接对方电话或者挂断对方lwforfzdVoices.kickout(data)，如果客户调用则相当于挂机。data来自于request\_talking.

二、 获取uuid列表的房间状态, lwforfzdVoices.get\_room\_infos(UUIDS,cb,fb)

uuids输入参数为所有客服的uuid的列表。

cb为成功的回调函数，原型为function(data), data的结构为tuple的列表，[{uuid:uuid, room:room,member\_num:m, waiting\_num:w}}]，其中member\_num为房间里面的通话的人数，waiting\_num为等待的人数。如果客服没有进入房间，将不会在此列表中。

fb为失败的回调函数.