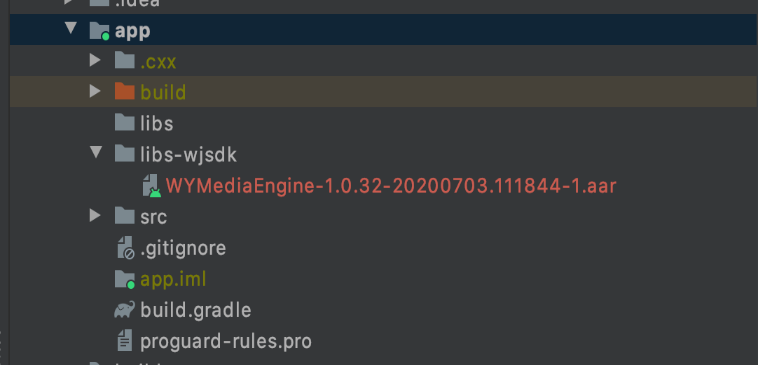
关于媒体引擎音频直播sdk库的使用说明

1. 关于sdk库的集成方法

1、采用直接复制aar 包到工程libs目录的方式引入。

首先获取到aar文件 WYMediaEngine-x.x.x.aar。把aar文件复制到工程的目录下.如图：



2、通过build.gradle脚本引入工程。比如 我们把aar库放入到目录libs-wjsdk时，

implementation fileTree(dir: 'libs-wjsdk', include: ['\*.aar'])

3、同步一下工程确认已经被引入工程。

1. 关于sdk库的初始化

1、配置sdk 参数。

配置声音采样率与声道 ， 默认为44100hz，2声道即可

/\*\*

 \* 设置录音参数

 \* @param sampleRate

 \* @param channel

 \*/

public static void setRecordingAudioFrameParameters(int sampleRate, int channel){

    config.sampleRate = sampleRate;

    config.channels = channel;

}

然后初始化sdk.

示例：

private AudioEngine mAudioEngine;

AudioEngine.setRecordingAudioFrameParameters(44100, 2);

AudioEngine.setLogFileDir(logfileDir);

AudioEngine.setDebugMode(true);//设置调试模式

mAudioEngine = WJMediaEngine.createAudioEngine(getApplicationContext(), appId, audioEngineHandler);

注：目前appId设置成“8kJzRafedna967WbqyeyrPCLYsmxi6A1”可以直接使用。

2、sdk初始化之后，生成媒体引擎实例子，通过实例就可以调用相关API：

mAudioEngine.xxxxxx();

1. 关于sdk的接口
2. sdk方法

/\*\*

 \* 设置log 目录

 \* @param logFileDir

 \*/

public static void setLogFileDir(String logFileDir){

    config.logDir = logFileDir;

}

/\*\*

 \* 设置录音参数

 \* @param sampleRate

 \* @param channel

 \*/

public static void setRecordingAudioFrameParameters(int sampleRate, int channel){

    config.sampleRate = sampleRate;

    config.channels = channel;

}

/\*\*

 \* 设置是否调式模式

 \* @param isDebug

 \*/

public static void setDebugMode(boolean isDebug){

    config.DebugMode = isDebug;

}

 /\*\*

 \* 设置连接正式、测试环境域名及备份ip，注意！！！，若不调用则使用默认配置，否则，必须在init方法调用前调用

 \* @param proEnvDomain      :正式环境域名

 \* @param proEnvBackupIp    :正式环境备份ip

 \* @param betaEnvDomain     :测试环境域名

 \* @param betaEnvBackupIp   :测试环境备份ip

 \*

 \*/

public static void setConDomainIP(String proEnvDomain,String proEnvBackupIp,String betaEnvDomain,String betaEnvBackupIp){

    config.proEnvDomain     = proEnvDomain;

    config.proEnvBackupIp   = proEnvBackupIp;

    config.betaEnvDomain    = betaEnvDomain;

    config.betaEnvBackupIp  = betaEnvBackupIp;

}

/\*\*

 \* 设置隐藏参数

 \* @param key

 \* @param value

 \*/

public abstract void setParams(String key, String value);

/\*\*

 \* 该方法设置接收到的语音从听筒或扬声器出声

 \* @param enable true：默认从外放（扬声器）出声 false：默认从听筒出声

 \*/

public abstract boolean setDefaultAudioRoutetoSpeakerphone(boolean enable);

/\*\*

 \* 开启频道内谁说话回调

 \* @param interval

 \* @param smooth

 \*/

public abstract void enableAudioVolumeIndication(int interval, int smooth);

/\*\*

 \* 进入频道，进入频道后，不设置角色就是无角色状态，不推流也不拉流。

 \* @param channel

 \* @param uid

 \* @param extUid app的账号系统id

 \*/

public abstract void joinChannel(long channel, long uid, final String extUid);

/\*\*

 \* 离开频道

 \*/

public abstract void leaveChannel();

/\*\*

 \* 设置当前用户角色，在进入频道成功后才能设置成功，

 \* 不在频道中调用报错：{@link ErrorCode#AUDIO\_CHANGE\_ROLE\_NOT\_IN\_CHANNEL\_ERROR}

 \* @param role {@link AudioRole}

 \*/

public abstract void setRole(AudioRole role);

/\*\*

 \* 是否静音当前麦克风

 \* @param mute

 \*/

public abstract void muteLocalAudioStream(boolean mute);

/\*\*

 \* 是否静音喇叭

 \* @param mute

 \*/

public abstract void muteAllRemoteAudioStreams(boolean mute);

/\*\*

 \* 调节麦克风音量

 \* @param volume

 \* @return

 \*/

public abstract int adjustRecordingSignalVolume(int volume);

/\*\*

 \* 调节播放音量

 \* @param volume

 \* @return

 \*/

public abstract int adjustSpeakerVolume(int volume);

/\*\*

 \* 开始音乐播放

 \* @param filePath mp3 文件绝对路径

 \* @return

 \*/

public abstract int startAudioMixing(java.lang.String filePath);

/\*\*

 \* 停止音乐播放

 \* @return

 \*/

public abstract int stopAudioMixing();

/\*\*

 \* 暂停音乐播放

 \* @return

 \*/

public abstract int pauseAudioMixing();

/\*\*

 \* 恢复音乐播放

 \* @return

 \*/

public abstract int resumeAudioMixing();

/\*\*

 \* 调整音乐的音量

 \* @param volume

 \* @return

 \*/

public abstract int adjustAudioMixingVolume(int volume);

/\*\*

 \* 获取背景音乐总时长

 \* @return

 \*/

public abstract long getAudioMixingDuration();

/\*\*

 \* 获取背景音乐当前时长

 \* @return

 \*/

public abstract long getAudioMixingCurrentPosition();

/\*\*

 \* 移动到背景音乐某个位置播放

 \* @param pos 位置 0 < pos < getAudioMixingDuration()

 \* @return

 \*/

public abstract long setAudioMixingPosition(long pos);

/\*\*

 \* 打开cdn流

 \* @param userId appid

 \* @param userId 用户id

 \* @param url cdn 地址

 \*/

public abstract void openCdn(String appid, long userId, String url);

/\*\*

 \* 关闭cdn

 \*/

public abstract void closeCdn();

/\*\*

 \* 开始语音录制

 \* @param path 要保存的文件路径

 \* @param quality 录音质量

 \*/

public abstract boolean startAudioRecording(String path, AudioRecordQuality quality);

/\*\*

 \* 结束语音录制

 \* @return

 \*/

public abstract boolean stopAudioRecording();

/\*\*

 \* vad 展示方式，

 \* true 播放音乐一定设置vad标记

 \* false 播放音乐不设置vad 标记

 \*/

public abstract void setAudioMixingVAD(boolean enable);

/\*\*

 \* VAD 检测方式 ，true 如果有噪音也回触发vad，

 \* false 只有人声出现的时候才有vad

 \*/

public abstract void setEnergyVadFlag(boolean enable);

/\*\*

 \* 设置驱动模式，

 \* @param mode 0:表示由硬件驱动，1:表示由内部定时器驱动

 \*/

public abstract void setDriveMode(int mode);

/\*\*

 \* 设置降噪等级

 \* @param level 范围[0,100] 闭区间

 \*/

public abstract void setDenioseLevel(int level);

/\*\*

 \* 设置回声消除模式

 \* @param mode  {@link AECMode} //MODE\_HARDWARE,MODE\_SOFTWARE

 \*/

public abstract void setAECMode(AECMode mode);

1. sdk回调

public abstract class AudioEngineHandler {

    /\*\*

     \* sdk 异常回调

     \* @param err

     \*/

    public void onError(int err) { }

    /\*\*

     \* 加入频道成功

     \* @param channel

     \* @param uid

     \* @param elapsed

     \*/

    public void onJoinChannelSuccess(long channel, long uid, int elapsed) {}

    public void onRejoinChannelSuccess(long channel, long uid, int elapsed) {}

    /\*\*

     \* 离开频道回调

     \*/

    public void onLeaveChannel() {}

    /\*\*

     \* 网络质量回调

     \* @param quality 表示丢包率，范围是 0～100， 推荐大于50才toast网络质量不好。

     \*/

    public void onLastmileQuality(int quality) {}

    /\*\*

     \* 链接丢失

     \*/

    public void onConnectionLost() {}

    /\*\*

     \* 说话的uid 回调

     \* @param speakers

     \*/

    public void onAudioVolumeIndication(AudioVolumeInfo[] speakers) {}

    /\*\*

     \* 静音麦克风回调

     \* @param uid

     \* @param muted

     \*/

    public void onUserMuteAudio(long uid, boolean muted) {}

    /\*\*

     \* 用户角色变更回调

     \* @param oldRole

     \* @param newRole

     \*/

    public void onClientRoleChanged(AudioRole oldRole, AudioRole newRole) {}

    /\*\*

     \* 本地音乐每首歌结束回调

     \*/

    public void onAudioMixingFinished() {}

    /\*\*

     \* PC 端开播回调

     \* @param isBroadcast

     \*/

    public void onPCBroadcast(boolean isBroadcast){

    }

    /\*\*

     \* 被服务器踢出

     \* @param code

     \* @param reason

     \*/

    public void onBeKicked(int code, String reason){};

    /\*\*

     \* 打电话状态

     \* @param inInCall 是否在打电话

     \*/

    public void onReceivePhoneCall(boolean inInCall){}

    /\*\*

     \* 收到更新MicUniqueID状态

     \* @param micUniqueID 更新的micUniqueID

    \*/

    public void OnMicUniqueIDUpdate(String micUniqueID){}

    /\*\*

     \* 收到为主播时被服务器更新为观众的消息

     \* @param

     \*/

    public void OnKickedToAudience(){}

    /\*\*

     \* 收到音量数据的消息

     \* @param volume 音量大小 【0，100】

     \*/

    public void OnCaptureVolumeData(int  volume){}

}

注：初始化后调用进房，

public abstract void joinChannel(long channel, long uid, final String extUid);

进房成功后，要把当前用户的角色设置下来，默认角色为无角色，设置为主播时，可说可听，设置为观众时，仅听。

public abstract void setRole(AudioRole role);