URTC Android sdk开发指导

# Sdk版本

目前sdk大致分为2类，这两类都目前都在维护更新，一类master分支版本，这类符合大部分手机sdk开发用户，一类dev扩展版本，适合各类使用定制机或者开发板的有特定开发需求的客户，目前一部分功能譬如rtsp拉流只在dev扩展版本中有实现，并且dev扩展版本将会定期更新master分支上的最新修改功能。

# 加入房间

## Token获取

请参考<https://docs.ucloud.cn/urtc/sdk/token>。

Sdk 通过UCloudRtcSdkEnv.setSdkMode()设置测试或者正式模式，测试模式请赋予

CommonUtils 里的String APP\_ID/String SEC\_KEY 用户相关的appid和appkey

## Classtype

加入房间前需要调用setClassType(UCloudRtcSdkRoomType classType)设置教室类型，small为小班课，否则为直播课。小班课适合叫少的人连麦，直播适用于大型直播场景

## 加入准备

加入房间需要做一些准备工作，大致包括

// 创建sdk事件回调

UCloudRtcSdkEventListener eventListener = new UCloudRtcSdkEventListener(){

...

}

// 创建引擎并注册回调对象

UCloudRtcSdkEngine sdkEngine = UCloudRtcSdkEngine.createEngine(eventListener);

sdkEngine.setAutoPublish(true); // 设置是否自动发布

sdkEngine.setAutoSubscribe(true); // 设置是否自动订阅

sdkEngine.setAudioOnlyMode(true) ; // 设置是否纯音频模式

sdkEngine.configLocalAudioPublish(true) ; // 设置是否发布音频

UCloudRtcSdkAuthInfo info = new UCloudRtcSdkAuthInfo();

info.setAppId(mAppid); // appid

info.setToken(mRoomToken); // token

info.setRoomId(mRoomid); // 房间号

info.setUId(mUserid); // 用户名

sdkEngine.joinChannel(info); //加入房间

## 加入时序图



# 发布

## 手动发布和自动发布

setAutoPublish 控制手动或者自动发布。

手动发布代表当sdk加入房间后，sdk会根据sdk当前的设置自动选择合适的publish接口发布对应的音频以及音视频，屏幕流。

选择流程大致为如果设置了audioonly，则根据configLocalAudioPublish选择是否发布音频流，如果不是audioonly，则根据configLocalCameraPublish 和 configLocalScreenPublish 的设置选择发布摄像头流或者屏幕流或者两者都发布。

自动发布的局限性在于无法灵活组合，譬如想要发布一路音频流加一路屏幕流，好处在于如果是比较普遍的需求即只需要发布一路摄像头或者音频流这种可以不用在调用publish接口相关代码。

## 发布权限

setStreamRole 接口控制了sdk是否具备发布或者订阅或者两项能力兼具。当然，sdk内部也做了android上camera部分的权限检测功能，如果具备上行发布能力，configLocalCameraPublish也是true，但是app没有赋予camera权限，也是会发布失败的。

## 媒体类型

发布的媒体类型sdk目前设有2种，video/screen，语音推流类型，语音+摄像头属于video，屏幕捕捉 属于 screen。Master版本的sdk，video类型包含了语音通道，screen不含语音通道。Dev版本的screen类型不在是屏幕捕捉，而是rtsp，相关的Dev版本的rtsp功能请参考dev 版本相关文档，此处不在赘述。

## 通道源数据

UCloudRtcSdkEnv.setCaptureMode（）决定了当媒体类型为video时，通道内数据的来源方式，master版本支持camera和外部自定义的数据采集，dev版本支持camera和 rtsp混合切换通道内数据切换，具体请看dev版本相关文档。

## 发布前预览

Sdk提供了发布前预览功能，目前只支持手机摄像头发布前预览。使用方通过调用

startCameraPreview接口即可实现，此时并未推流，后续可以通过调用publish推流，在onlocalpublish回调中无需在调用renderlocalview接口渲染视图。

## 发布相关参数和接口

UCloudRtcSdkEnv.setCaptureMode()

setAutoPublish(boolean autoPub)

setAudioOnlyMode(boolean audioOnly)

configLocalAudioPublish(boolean enable)

configLocalCameraPublish(boolean enable)

configLocalScreenPublish(boolean enable)

setStreamRole(UCloudRtcSdkStreamRole role)

renderLocalView(…)

startCameraView(…)

## 发布时序图

一个简单的发布流程大致基于以下图示：



Sdk通过onlocalpublish告诉调用者发布结果，调用需要通过结果判断是否发布成功，发布成功是否需要回显（renderlocalview接口），具体的渲染步骤请参考下面的渲染模块说明。

# 订阅

## 自动订阅和手动订阅

setAutoSubscribe(boolean autoSub)控制订阅或者自动订阅。

如果sdk设置了自动订阅，sdk会在远端推流上来的时候帮助用户自动调用subscribe接口订阅任何新推上来的流，前提是权限设置了sdk拥有拉流权限。

和手动订阅的差异在于自动订阅会把收到的新的流都订阅一下，有时候这并不是用户所期望的，可能用户期望的是要订阅一部分流而忽略另外一部分，那这个时候就需要用到手动订阅了。

## 拉流权限

同样由setStreamRole 接口来控制sdk是否具有拉流权限

## 监听远端发布消息

Sdk 通过void onRemotePublish(UCloudRtcSdkStreamInfo info)接口监听了远端发布流的信息。其中info中包含了远端发布流的关键信息，包括用户id，流的媒体类型等。用户可以借助这些信息在手动订阅模式下选择是否需要订阅这条流

## 调用订阅

Sdk通过subscribe(UCloudRtcSdkStreamInfo info)接口手动订阅那些需要被用户订阅的流。这里的info入参就是onRemotePublish带出来的输出参数。

## 订阅回调

调用订阅后需要去监听onSubscribeResult(…)接口去监听到底是否订阅成功，即code = 0。订阅成功后需要去根据自己的界面需求调用startRemoteView()去渲染，

## 订阅相关参数接口

void onRemotePublish(UCloudRtcSdkStreamInfo info)

subscribe(UCloudRtcSdkStreamInfo info)

onSubscribeResult(…)

startRemoteView(…)

## 订阅时序图

一个简单的订阅流程大致基于以下图示：



# 渲染

## Surfaceview和Textureview

Sdk目前支持两种方式的渲染，一种surfaceview，一种textureview。关于surfaceview和textureview的差异可以参考相关google android dev说明。Ucloud提供了UCloudRtcRenderView支持surfaview，UCloudRtcRenderTextureView支持textureview。当构造这两个Ucloud提供的render的时候，方式有所不同，其中UCloudRtcRenderView继承自surfaview，而UCloudRtcRenderTextureView需要通过构造函数包含用户自己的textureview，构造完毕后请调用init()函数初始化egl环境后，在把UCloudRtcRenderView或者UCloudRtcRenderTextureView参数传给startremoteview或者randerlocalview接口即可。

另外有几点请注意，传入的render相关的view需要被attach到viewroot上，否则无法渲染出该视图。其次当确定该view不在需要被使用时，调用render的release函数释放egl环境，否则可能造成不必要的内存泄漏。

## 渲染方式

Sdk为surfaview和textureview提供了几种渲染效果，目前包括aspect\_fit/aspect\_fill/fill三种。前两种都是保持纵横比的，区别在于aspect\_fill是类似于android中的centercrop效果，aspect\_fit类似于fit效果，而fill则是不保持纵横比的拉伸图像以显示整个view，这个一般用的较少。

# 混音

混音是指播放本地或者在线音乐文件，同时让频道内的其他人听到此音乐。混音方法主要用来播放比较长的背景音，比如直播的时候播放的音乐，同时只可以有一个文件播放。目前仅支持wav和mp3格式。

## 相关参数接口

播放：

startPlayAudioFile(String filePath, boolean isRemotePlay, boolean isLoop);

filePath:音乐文件的路径或网址。

isRemotePlay:是否远端一起播放。

isLoop:是否循环播放。

停止：

stopPlayAudioFile();

暂停：

pauseAudioFile ();

恢复：

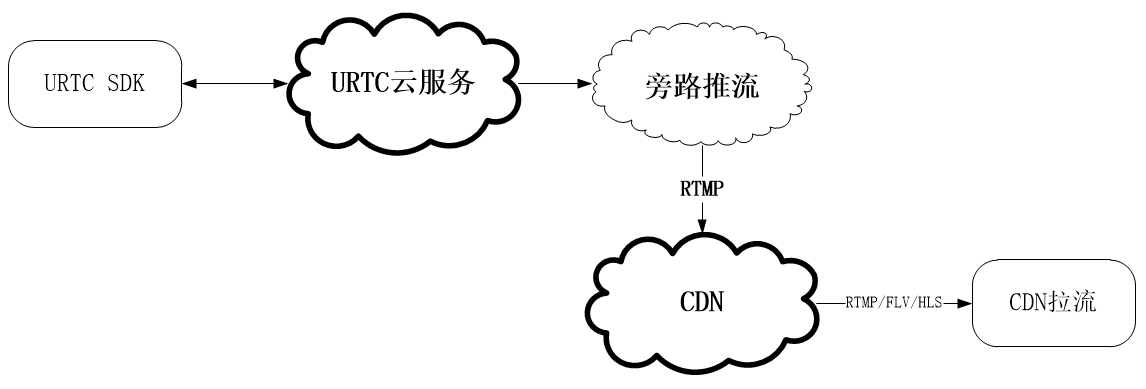
resumeAudioFile ();

播放结束回调：

onAudioFileFinish();

# 转推

URTC 推流，支持将音视频会议、直播的内容，推流到直播CDN。用户无需安装APP，可以直接通过Web浏览器拉CDN直播流，观看会议、直播的内容



在推流过程中，可以推单路流到直播CDN；也可以把音视频会议、多个主播的音视频混流转码，再推流到直播CDN。混流会将多个音视频流混合成单个流，并设置音视频参数、混流风格。

## 开始转推

UCloudRtcSdkMixProfile mixProfile = UCloudRtcSdkMixProfile.getInstance().assembleMixParamsBuilder()

.type(UCloudRtcSdkMixProfile.MIX\_TYPE\_RELAY)

//画面模式

.layout(UCloudRtcSdkMixProfile.LAYOUT\_CLASS\_ROOM\_2)

//画面分辨率

.resolution(1280, 720)

//背景色

.bgColor(0, 0, 0)

//画面帧率

.frameRate(15)

//画面码率

.bitRate(1000)

//h264视频编码

.videoCodec(UCloudRtcSdkMixProfile.VIDEO\_CODEC\_H264)

//编码质量

.qualityLevel(UCloudRtcSdkMixProfile.QUALITY\_H264\_CB)

//音频编码

.audioCodec(UCloudRtcSdkMixProfile.AUDIO\_CODEC\_AAC)

//主讲人ID

.mainViewUserId(mUserid)

//主讲人媒体类型

.mainViewMediaType(UCLOUD\_RTC\_SDK\_MEDIA\_TYPE\_VIDEO.ordinal())

//加流方式手动

.addStreamMode(UCloudRtcSdkMixProfile.ADD\_STREAM\_MODE\_MANUAL)

//添加流列表，也可以后续调用MIX\_TYPE\_UPDATE 动态添加

.addStream(mUserid,UCLOUD\_RTC\_SDK\_MEDIA\_TYPE\_VIDEO.ordinal())

//设置转推cdn 的地址

.addPushUrl("rtmp://rtcpush.ugslb.com/rtclive/" + mRoomid)

.build();

sdkEngine.startRelay(mixProfile); // 开始转推

## 停止转推

sdkEngine.stopRelay(null);

## 转推状态回调

public void onRelayStatusNotify(UCloudRtcSdkMediaServiceStatus status, int code, String msg, String userId, String roomId, String mixId, String[] pushUrls) {

status:转推状态

code:回调返回值

msg:回调提示信息

userId:用户id

roomId:房间号

mixId:混流号

pushUrls:转推地址

## 转推状态更新

在转推模式开启后，可以动态更新推流的状态。另外，updateMixConfig接口还可以用于添加新流。

sdkEngine.updateMixConfig(mixProfile); // 更新转推状态

当需要添加新的流到转推地址时，mixProfile参数的设置方法和转推开启相同，只是其中的type和addStream需要做修改，type需改成UCloudRtcSdkMixProfile MIX\_TYPE\_UPDATE，addStream的userId参数改成需要推流的用户id。