**RTC Android SDK接口文档**

1. 类型定义

## 基本常量定义

### URTCSdkAudioDevice

音频设备类型定义

public enum URTCSdkAudioDevice {

//无效设备

URTC\_SDK\_AUDIODEVICE\_NONE,

//扬声器

URTC\_SDK\_AUDIODEVICE\_SPEAKER,

//耳机模式

URTC\_SDK\_AUDIODEVICE\_WIRED\_HEADSET,

//听筒

URTC\_SDK\_AUDIODEVICE\_EARPIECE,

//蓝牙耳机

URTC\_SDK\_AUDIODEVICE\_BLUETOOTH

};

### URTCSdkCameraOrientation

摄像头旋转角度

public enum URTCSdkCameraOrientation {

URTC\_SDK\_CAMERA\_ORIENTATION\_0,

URTC\_SDK\_CAMERA\_ORIENTATION\_90,// 90度

URTC\_SDK\_CAMERA\_ORIENTATION\_180, //180度

URTC\_SDK\_CAMERA\_ORIENTATION\_270 //270度

}

### URTCSdkLogLevel

日志级别定义

public enum URTCSdkLogLevel {

URTC\_SDK\_LogLevelNone, // 不打印

URTC\_SDK\_LogLevelDebug,

URTC\_SDK\_LogLevelInfo,

URTC\_SDK\_LogLevelWarn,

URTC\_SDK\_LogLevelError

}

### URTCSdkMediatype

视频媒体源类型

public enum URTCSdkMediatype {

URTC\_SDK\_MEDIA\_TYPE\_NULL(0),

URTC\_SDK\_MEDIA\_TYPE\_VIDEO(1), //摄像头

URTC\_SDK\_MEDIA\_TYPE\_SCREEN(2), //屏幕

URTC\_SDK\_MEDIA\_TYPE\_FILE(3) ; //文件（暂不支持）

private int result ;

URTCSdkMediatype(int type) {

this.result = type ;

}

public int getResult() {

return this.result ;

}

public static URTCSdkMediatype matchValue(int code) {

for (URTCSdkMediatype value: values()) {

if (value.ordinal() == code) {

return value ;

}

}

return URTC\_SDK\_MEDIA\_TYPE\_NULL ;

}

}

### URTCSdkMode

public enum URTCSdkMode {

URTC\_SDK\_MODE\_NORMAL, //正式环境

RTC\_SDK\_MODE\_TRIVAL //测试环境

}

### URTCSdkScaleType

视频显示类型

public enum URTCSdkScaleType {

URTC\_SCALE\_ASPECT\_FIT, //保持比例

URTC\_SCALE\_ASPECT\_FILL, //平铺

URTC\_SCALE\_ASPECT\_BALANCED // 自动

}

### URTCSdkStreamtype

媒体流类型

public enum URTCSdkStreamtype {

URTC\_SDK\_STREAM\_TYPE\_NULL(0),

URTC\_SDK\_STREAM\_TYPE\_PUB(1), // 发布流

URTC\_SDK\_STREAM\_TYPE\_SUB(2); // 订阅流

private int result ;

URTCSdkStreamtype(int type) {

this.result = type ;

}

public static URTCSdkStreamtype matchValue(int code) {

for (URTCSdkStreamtype value: values()) {

if (value.ordinal() == code) {

return value ;

}

}

return URTC\_SDK\_STREAM\_TYPE\_NULL ;

}

public int getResult() {

return this.result ;

}

}

### URTCSdkTrackType

媒体轨道类型

public enum URTCSdkTrackType {

URTC\_SDK\_TRACK\_TYPE\_NULL(0),

URTC\_SDK\_TRACK\_TYPE\_AUDIO(1), //音频轨道

URTC\_SDK\_TRACK\_TYPE\_VIDEO(2), //视频轨道

URTC\_SDK\_TRACK\_TYPE\_DATA(3); //数据轨道

private int result ;

URTCSdkTrackType(int type) {

this.result = type ;

}

public static URTCSdkTrackType matchValue(int code) {

for (URTCSdkTrackType value: values()) {

if (value.ordinal() == code) {

return value ;

}

}

return URTC\_SDK\_TRACK\_TYPE\_NULL ;

}

public int getResult() {

return this.result ;

}

}

### URTCSdkStreamRole

本地流权限配置

public enum URTCSdkStreamRole {

URTC\_SDK\_STREAM\_ROLE\_PUB, // 发布权限

URTC\_SDK\_STREAM\_ROLE\_SUB,// 订阅权限

URTC\_SDK\_STREAM\_ROLE\_BOTH //全部权限

}

### URTCSdkStreamSt

// 媒体流状态

public enum URTCSdkStreamSt {

URTC\_SDK\_STREAM\_ST\_NULL(0),

URTC\_SDK\_STREAM\_ST\_ADD(1), //媒体流添加

URTC\_SDK\_STREAM\_ST\_REMOVE(2), //媒体流删除

URTC\_SDK\_STREAM\_ST\_UPDATE(3); //媒体流更新

URTCSdkStreamSt(int value) {

this.mValue = value ;

}

public static URTCSdkStreamSt matchValue(int value) {

for (URTCSdkStreamSt en : values()) {

if (en.ordinal() == value) {

return en ;

}

}

return URTCSdkStreamSt.URTC\_SDK\_STREAM\_ST\_NULL ;

}

private int mValue ;

}

### URTCSdkUserSt

//用户状态

public enum URTCSdkUserSt {

URTC\_SDK\_USER\_ST\_NULL(0),

URTC\_SDK\_USER\_ST\_JOIN(1), // 用户加入房间

URTC\_SDK\_USER\_ST\_LEAVE(2), // 用户离开

URTC\_SDK\_USER\_ST\_REJOIN(3); // 用户重新加入

URTCSdkUserSt(int value) {

this.value = value ;

}

public static URTCSdkUserSt matchValue(int value) {

for (URTCSdkUserSt en : values()) {

if (en.ordinal() == value) {

return en ;

}

}

return URTCSdkUserSt.URTC\_SDK\_USER\_ST\_NULL ;

}

private int value ;

}

### URTCSdkVideoProfile

视频分辨率等级

public enum URTCSdkVideoProfile {

UCloudRtcVideoProfile\_320\_180,// 视频配置分辨率 320\*180,对应传输最小码率 100kbps 最大码率 200kbps 初始码率 100kbps

UCloudRtcVideoProfile\_352\_288, // 视频配置分辨率 352\_288,对应传输最小码率 100kbps 最大码率 300kbps 初始码率 200kbps

UCloudRtcVideoProfile\_480\_360,// 视频配置分辨率 480\_360,对应传输最小码率 100kbps 最大码率 400kbps 初始码率 200kbps

UCloudRtcVideoProfile\_640\_360 //视频配置分辨率 640\_360,对应传输最小码率 200kbps 最大码率 400kbps 初始码率 300kbps

UCloudRtcVideoProfile\_640\_480, //视频配置分辨率 640\_480,对应传输最小码率 200kbps 最大码率 500kbps 初始码率 300kbps

UCloudRtcVideoProfile\_1280\_720; //视频配置分辨率 1280\_720,对应传输最小码率 500kbps 最大码率 1000kbps 初始码率 600kbps

public static URTCSdkVideoProfile matchValue(int value) {

for (URTCSdkVideoProfile en : values()) {

if (en.ordinal() == value) {

return en ;

}

}

return URTCSdkVideoProfile.UCloudRtcVideoProfile\_320\_240 ;

}

}

## 通用类定义

### URTCSdkAuthInfo

//登录参数类，参数可以通过set get 方法操作

public class URTCSdkAuthInfo {

String mAppId; //应用标识

String mRoomid; // 房间标识

String mUid; //用户标识

String mToken; //用户登录凭证

public String getmAppId() {

return mAppId;

}

public void setmAppId(String mAppId) {

this.mAppId = mAppId;

}

public String getmRoomid() {

return mRoomid;

}

public void setmRoomid(String mRoomid) {

this.mRoomid = mRoomid;

}

public String getmUid() {

return mUid;

}

public void setmUid(String mUid) {

this.mUid = mUid;

}

public String getmToken() {

return mToken;

}

public void setmToken(String mToken) {

this.mToken = mToken;

}

public URTCSdkAuthInfo()

{

mAppId = "" ;

mRoomid = "" ;

mUid = "" ;

mToken = "" ;

}

}

### URTCSdkMediaOp

订阅或者发布时媒体类型

public class URTCSdkMediaOp {

private String mUid ; // 用户id

private URTCSdkMediatype mMediaType ; //媒体类型

private boolean mEnableVideo ; // 是否打开视频

private boolean mEnableAudio ; //是否打开音频

public URTCSdkMediaOp() {

mUid = "" ;

mMediaType = URTCSdkMediatype.URTC\_SDK\_MEDIA\_TYPE\_VIDEO ;

mEnableAudio = false ;

mEnableVideo = false ;

}

public String getmUid() {

return mUid;

}

public void setmUid(String mUid) {

this.mUid = mUid;

}

public URTCSdkMediatype getmMediaType() {

return mMediaType;

}

public void setmMediaType(URTCSdkMediatype mMediaType) {

this.mMediaType = mMediaType;

}

public boolean ismEnableVideo() {

return mEnableVideo;

}

public void setmEnableVideo(boolean mEnableVideo) {

this.mEnableVideo = mEnableVideo;

}

public boolean ismEnableAudio() {

return mEnableAudio;

}

public void setmEnableAudio(boolean mEnableAudio) {

this.mEnableAudio = mEnableAudio;

}

}

### URTCSdkStats

媒体当前状态

public class URTCSdkStats {

String mUid; //用户id

URTCSdkMediatype mMtype; //媒体类型

URTCSdkTrackType mTracktype; // //轨道类型

int mBitrate = 0; // 码率

int mLostpre = 0; // 丢包率

int mRttMsg = 0 ; //发送的rtt

int mDelayMs = 0 ;// 接受端当前delay

public URTCSdkStats()

{

this.mUid = "" ;

this.mMtype = URTCSdkMediatype.URTC\_SDK\_MEDIA\_TYPE\_NULL ;

this.mTracktype = URTCSdkTrackType.URTC\_SDK\_TRACK\_TYPE\_NULL ;

this.mBitrate = 0 ;

this.mLostpre = 0 ;

this.mRttMsg = 0 ;

this.mDelayMs = 0 ;

}

public String getmUid() {

return mUid;

}

public void setmUid(String mUid) {

this.mUid = mUid;

}

public URTCSdkMediatype getmMtype() {

return mMtype;

}

public void setmMtype(URTCSdkMediatype mMtype) {

this.mMtype = mMtype;

}

public URTCSdkTrackType getmTracktype() {

return mTracktype;

}

public void setmTracktype(URTCSdkTrackType mTracktype) {

this.mTracktype = mTracktype;

}

public int getmBitrate() {

return mBitrate;

}

public void setmBitrate(int mBitrate) {

this.mBitrate = mBitrate;

}

public int getmLostpre() {

return mLostpre;

}

public void setmLostpre(int mLostpre) {

this.mLostpre = mLostpre;

}

public int getmRttMsg() {

return mRttMsg;

}

public void setmRttMsg(int mRttMsg) {

this.mRttMsg = mRttMsg;

}

public int getmDelayMs() {

return mDelayMs;

}

public void setmDelayMs(int mDelayMs) {

this.mDelayMs = mDelayMs;

}

}

### URTCSdkStreamInfo

媒体流信息

public class URTCSdkStreamInfo {

String mUid; //用户id

boolean mHasvideo; // 是否启用视频

boolean mHasaudio; // 是否启用音频

boolean mHasdata; // 是否启用数据流 //暂不支持

URTCSdkMediatype mMediatype; // 媒体流类型

public String getmUid() {

return mUid;

}

public void setmUid(String mUid) {

this.mUid = mUid;

}

public boolean ismHasvideo() {

return mHasvideo;

}

public void setmHasvideo(boolean mHasvideo) {

this.mHasvideo = mHasvideo;

}

public boolean ismHasaudio() {

return mHasaudio;

}

public void setmHasaudio(boolean mHasaudio) {

this.mHasaudio = mHasaudio;

}

public boolean ismHasdata() {

return mHasdata;

}

public void setmHasdata(boolean mHasdata) {

this.mHasdata = mHasdata;

}

public URTCSdkMediatype getmMediatype() {

return mMediatype;

}

public void setmMediatype(URTCSdkMediatype mMediatype) {

this.mMediatype = mMediatype;

}

public URTCSdkStreamInfo()

{

mUid = "" ;

mHasvideo = false ;

mHasaudio = false ;

mHasdata = false ;

mMediatype = URTCSdkMediatype.URTC\_SDK\_MEDIA\_TYPE\_VIDEO ;

}

}

### URTCSdkSurfaceVideoView

// 视频渲染view

public class URTCSdkSurfaceVideoView extends FrameLayout {

private static final String TAG = "==URTCSfVideoView==";

private URTCRenderView mSurface ; // 视频窗口

private Context mContext ; // android 上下文环境

public URTCSdkSurfaceVideoView(Context context) {

super(context);

mContext = context ;

}

public URTCSdkSurfaceVideoView(Context context, AttributeSet attrs) {

super(context, attrs);

mContext = context ;

}

public void init() {

mSurface = new URTCRenderView(mContext, this);

mSurface.init(URTCPeerManger.GetInstance().getRenderContext(), null);

mSurface.setEnableHardwareScaler(true /\* enabled \*/);

setBackgroundColor(Color.BLACK);

addView(mSurface, new LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.MATCH\_PARENT));

}

public URTCRenderView getSurfaceView() {

return mSurface;

}

public void setZOrderMediaOverlay(boolean enable) {

mSurface.setZOrderMediaOverlay(enable);

}

public void setScalingType(URTCSdkScaleType sType) {

if (mSurface != null) {

mSurface.setScaleType(sType);

}

}

public void setMirror(boolean enable) {

if (mSurface != null) {

mSurface.setMirror(enable);

}

}

public void release() {

if (mSurface != null) {

mSurface.release();

mSurface = null;

}

}

public void refrush() {

if (mSurface != null) {

mSurface.clearImage();

}

}

@Override

protected void onLayout(boolean changed, int left, int top, int right, int bottom) {

super.onLayout(changed, left, top, right, bottom);

if (mSurface != null) {

refrush() ;

}

}

## 回调函数定义

Sdk 事件回调接口定义

public interface URTCSdkEventListener {

//self join room

/\*\*

\* server 断开

\*/

public void onServerDisconnect() ;

/\*\*

\* 加入房间回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param roomid 房间标识

\*/

public void onJoinRoomResult(int code, String msg, String roomid) ;

/\*\*

\* 离开房间回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param roomid 房间标识

\*/

public void onLeaveRoomResult(int code, String msg, String roomid) ;

/\*\*

\* 房间重连中

\* @param roomid 房间标识

\*/

public void onRejoiningRoom(String roomid) ;

/\*\*

\* 房间重连成功

\* @param roomid 房间标识

\*/

public void onRejoinRoomResult(String roomid) ;

/\*\*

\* 本地发布结果

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param info 流标识

\*/

public void onLocalPublish(int code, String msg, URTCSdkStreamInfo info) ;

/\*\*

\* 本地取消发布结果

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param info 流标识

\*/

public void onLocalUnPublish(int code, String msg, URTCSdkStreamInfo info) ;

/\*\*

\* 其他用户加入房间

\* @param uid 用户标识

\*/

public void onRemoteUserJoin(String uid) ;

/\*\*

\* 其他用户离开房间

\* @param uid 用户标识

\*/

public void onRemoteUserLeave(String uid,int reson) ;

/\*\*

\* 远端发布媒体流

\* @param info 流标识

\*/

public void onRemotePublish(URTCSdkStreamInfo info) ;

/\*\*

\* 远端取消媒体发布

\* @param info 流标识

\*/

public void onRemoteUnPublish(URTCSdkStreamInfo info) ;

/\*\*

\* 订阅远端媒体流结果回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param info 流标识

\*/

public void onSubscribeResult(int code, String msg, URTCSdkStreamInfo info) ;

/\*\*

\* 取消订阅远端媒体流结果回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param info 流标识

\*/

public void onUnSubscribeResult(int code, String msg, URTCSdkStreamInfo info) ;

/\*\*

\* 本地媒体流mute 结果回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param mediatype 媒体流类型

\* @param tracktype 媒体轨道类型

\* @param mute true 关闭本地 false 打开本地

\*/

public void onLocalStreamMuteRsp(int code, String msg,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute) ;

/\*\*

\* 远端媒体流mute 结果回调

\* @param code int : 0 成功 非0 失败

\* @param msg 失败时消息描述

\* @param uid 用户标识

\* @param mediatype 媒体流类型

\* @param tracktype 媒体轨道类型

\* @param mute true 关闭本地 false 打开本地

\*/

public void onRemoteStreamMuteRsp(int code, String msg, String uid,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute) ;

/\*\*

\* 远端媒体流状态变化通知

\* @param uid 用户标识

\* @param mediatype 媒体流类型

\* @param tracktype 媒体轨道类型

\* @param mute true 关闭 false 打开

\*/

public void onRemoteTrackNotify(String uid,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute) ;

/\*\*

\* 发送端状态回调

\* @param rtstats 主要 码率 丢包率 rtt

\*/

public void onSendRTCStats(URTCSdkStats rtstats) ;

/\*\*

\* 接收端某一路状态通知

\* @param rtstats 发送端状态回调 主要 码率 丢包率 延时

\*/

public void onRemoteRTCStats(URTCSdkStats rtstats) ;

/\*\*

\* 本地音量通知

\* @param volume 音频能量 0 --100

\*/

public void onLocalAudioLevel(int volume) ;

/\*\*

\* 远端某一路音量通知

\* @param uid 用户标识

\* @param volume 音频能量 0 --100

\*/

public void onRemoteAudioLevel(String uid, int volume) ;

/\*\*

\* 自己 被踢出房间回调

\* @param code 踢出原因

\*/

public void onKickoff(int code) ;

/\*\*

\* SDK 内部警告回调

\* @param warn 警告原因

\*/

public void onWarning(int warn) ;

/\*\*

\* SDK 内部错误回调

\* @param error 错误吗

\*/

public void onError(int error) ;

}

1. 功能类描述

## URTCSdkEngine

引擎接口类，完成SDK所有功能操作

### 常量声明

#### 参数定义

申请桌面权限请求码

SCREEN\_CAPTURE\_REQUEST\_CODE = 1000

### 获取引擎类

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取引擎类 |
| 接口 | public static URTCSdkEngine createEngnine(URTCSdkEventListener eventlistener) |
| 返回值 | URTCSdkEngine |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| eventlistener [in] | 消息回调通知类 | URTCSdkEventListener | N |

#### 消息回调

无

### 销毁引擎

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 销毁引擎 |
| 接口 | public static void destory() |
| 返回值 | 无 |

#### 参数说明

无

#### 消息回调

无

#### 消息回调

无

### 获取桌面采集权限

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取桌面采集权限 |
| 接口 | public static void requestScreenCapture（Activity activity） |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| activity | 申请采集桌面的Activity | Activity | N |

#### 消息回调

无

### 获取桌面采集权限结果

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取桌面采集权限结果 |
| 接口 | public static void onScreenCaptureResult(Intent data) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| data [in] | 回调结果 | Intent | N |

#### 消息回调

无

### 获取SDK 版本

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取SDK 版本 |
| 接口 | public String getSdkVersion() |
| 返回值 | 对应的SDK版本号 |

#### 参数说明

无

#### 消息回调

无

### 设置音频设备

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置音频设备 |
| 接口 | public void setAudioDevice(URTCSdkAudioDevice device) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| device [in] | 音频设备类型 | URTCSdkAudioDevice | N |

#### 消息回调

无

### 获取音频默认设备

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取音频默认设备 |
| 接口 | public URTCSdkAudioDevice getDefaultAudioDevice() |
| 返回值 | URTCSdkAudioDevice |

#### 参数说明

无

#### 消息回调

无

### 设置扬声器状态

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 扬声器状态 |
| 接口 | public void setSpeakerOn(boolean on) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| on [in] | true开启扬声器 false 关闭扬声器 | boolean | N |

#### 消息回调

无

### 设置流操作权限

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置流操作权限 |
| 接口 | public int setStreamRole(URTCSdkStreamRole role) |
| 返回值 | 参见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| role[in] | 详见URTCSdkStreamRole说明 | URTCSdkStreamRole | N |

#### 消息回调

无

### 设置自动发布手动发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置自动发布 |
| 接口 | public int setAutoPublish(boolean autoPub) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| autoPub[in] | 是否自动发布流 | boolean | N |

#### 消息回调

无

### 设置自动订阅手动订阅

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置自动订阅 |
| 接口 | public int setAutoSubscribe(boolean autoSub) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| autoSub [in] | 是否自动订阅流 | boolean | N |

#### 消息回调

无

### 设置纯音频模式

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置纯音频模式 |
| 接口 | public int setAudioOnlyMode(boolean audioOnly) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| audioOnly[in] | 纯音频模式 | boolean | N |

#### 消息回调

无

### 设置视频显示等级

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置视频显示等级 |
| 接口 | public int setVideoProfile(URTCSdkVideoProfile profile) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| profile[in] | 视频质量等级 | URTCSdkVideoProfile | N |

#### 消息回调

无

### 加入房间

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 加入房间 |
| 接口 | public int joinChannel(URTCSdkAuthInfo auth) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| auth[in] | 加入房间参数 详见URTCSdkAuthInfo说明 | URTCSdkAuthInfo | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onJoinRoomResult(int code, String msg, String roomid) ;

### 离开房间

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 离开房间 |
| 接口 | public int leaveChannel(); |
| 返回值 | 详见错误码描述 |

#### 参数说明

无

#### 消息回调

参见 消息回调事件接口类对应接口

public void onLeaveRoomResult(int code, String msg, String roomid)

### 切换摄像头

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 切换摄像头 |
| 接口 | public int switchCamera() |
| 返回值 | 详见错误码描述 |

#### 参数说明

无

#### 消息回调

无

### 开启本地预览

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 开启本地预览 |
| 接口 | public int startPreview(URTCSdkMediatype mediatype, URTCSdkSurfaceVideoView view) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mediatype[in] | 详见URTCSdkMediatype参数说明 | URTCSdkMediatype | N |
| view [in] | 详见URTCSdkSurfaceVideoView参数说明 | URTCSdkSurfaceVideoView | N |

#### 消息回调

无

### 停止本地预览

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 停止本地预览 |
| 接口 | public int stopPreview(URTCSdkMediatype mediatype) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mediatype[in] | 详见URTCSdkMediatype参数说明 | URTCSdkMediatype | N |

#### 消息回调

无

### 发布媒体流

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 发布媒体流 |
| 接口 | public int publish(URTCSdkMediatype mtype, boolean hasvideo, boolean hasaudio) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mtype[in] | 发布流类型 详见URTCSdkMediatype | URTCSdkMediatype | N |
| hasvideo[in] | 发布流是否带视频 | boolean | N |
| has audio[in] | 发布流是否带音频 | boolean | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onLocalPublish(int code, String msg, URTCSdkStreamInfo info)

### 停止媒体流发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 停止媒体流发布 |
| 接口 | public int unPublish(URTCSdkMediatype mtype) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mtype[in] | 发布流类型 详见URTCSdkMediatype说明 | URTCSdkMediatype | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onLocalUnPublish(int code, String msg, URTCSdkStreamInfo info)

### 打开关闭本地音频

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 打开关闭本地音频（音频静音包发送） |
| 接口 | public int muteLocalMic(boolean mute) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mute[in] | 是否关闭音频 | boolean | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onLocalStreamMuteRsp(int code, String msg,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute)

### 打开关闭本地视频

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 打开关闭本地视频（黑屏帧发送） |
| 接口 | public int muteLocalVideo(boolean mute, URTCSdkMediatype mtype) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mute[in] | 是否关闭视频 | boolean | N |
| mtype[in] | 流类型 详见URTCSdkMediatype说明 | URTCSdkMediatype | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onLocalStreamMuteRsp(int code, String msg,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute)

### 开启远端渲染

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 开启远端渲染 函数只在自己订阅成功后才能调用 |
| 接口 | public int startRemoteView(URTCSdkStreamInfo info, URTCSdkSurfaceVideoView renderview) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| info[in] | 详见URTCSdkStreamInfo参数说明 | URTCSdkStreamInfo | N |
| view [in] | 详见UCloudVideoCanvas 参数说明 | URTCSdkSurfaceVideoView | N |

#### 消息回调

无

### 停止远端视频渲染

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 停止远端渲染 |
| 接口 | public int stopRemoteView(URTCSdkStreamInfo info) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| info[in] | 详见URTCSdkStreamInfo参数说明 | URTCSdkStreamInfo | N |

#### 消息回调

无

### 订阅远端媒体流

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 订阅远端媒体流 |
| 接口 | public int subscribe(URTCSdkStreamInfo info) |
| 返回值 | 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| info[in] | 详见URTCSdkStreamInfo参数说明 | URTCSdkStreamInfo | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onSubscribeResult(int code, String msg, URTCSdkStreamInfo info)

### 取消订阅远端媒体流

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 取消订阅远端媒体流 |
| 接口 | public int unSubscribe(URTCSdkStreamInfo info) |
| 返回值 | 0 成功 其他失败 详见错误码 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| info[in] | 详见URTCSdkStreamInfo参数说明 | URTCSdkStreamInfo | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onUnSubscribeResult(int code, String msg, URTCSdkStreamInfo info)

### 静音远端音频

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 静音开启远端音频 |
| 接口 | public int muteRemoteAudio(String uid, boolean mute) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| uid[in] | 用户标识 | String | N |
| mute[in] | 是否关闭远端音频 | boolean | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onRemoteStreamMuteRsp(int code, String msg, String uid,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute)

### 打开关闭远端视频

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 打开关闭远端视频 |
| 接口 | public int muteRemoteVideo(String uid, boolean mute) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| uid[in] | 用户标识 | String | N |
| mute[in] | 是否关闭远端视频 | boolean | N |

#### 消息回调

参见 消息回调事件接口类对应接口

public void onRemoteStreamMuteRsp(int code, String msg, String uid,

URTCSdkMediatype mediatype, URTCSdkTrackType tracktype, boolean mute)

### 打开关闭远端桌面

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 打开关闭远端桌面 |
| 接口 | public int muteRemoteScreen(String uid, boolean mute) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| uid[in] | 用户标识 | String | N |
| mute[in] | 是否关闭远端桌面 | boolean | N |

### 配置本地摄像头是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 配置本地摄像头是否发布 |
| 接口 | public int configLocalCameraPublish(boolean enable) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| enable[in] | 是否可以发布视频 | boolean | N |

#### 消息回调

无

### 摄像头是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 摄像头是否发布 |
| 接口 | public boolean isLocalCameraPublishEnabled() |
| 返回值 | true 有 false 没有 |

#### 参数说明

无

#### 消息回调

无

### 配置本地桌面是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 配置本地桌面是否发布 |
| 接口 | public int configLocalScreenPublish(boolean enable) |
| 返回值 | 详见错误码描述 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| enable[in] | 是否可以发布本地桌面 | boolean | N |

#### 消息回调

无

### 桌面是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 桌面是否发布 |
| 接口 | public boolean isLocalScreenPublishEnabled() |
| 返回值 | true 有 false 没有 |

#### 参数说明

无

#### 消息回调

无

### 配置本地音频是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 配置本地音频是否发布 |
| 接口 | public int configLocalAudioPublish(boolean enable) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| enable[in] | 是否可以发布本地音频 | boolean | N |

#### 消息回调

无

### 音频是否发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 音频是否发布 |
| 接口 | public boolean isLocalAudioPublishEnabled() |
| 返回值 | true 有 false 没有 |

#### 参数说明

无

#### 消息回调

无

### 是否为自动发布

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 是否为自动发布模式 |
| 接口 | public boolean isAutoPublish() |
| 返回值 | true 是 false 不是 |

#### 参数说明

无

#### 消息回调

无

### 是否为自动订阅模式

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 是否为自动订阅模式 |
| 接口 | public boolean isAutoSubscribe() |
| 返回值 | true 是 false 不是 |

#### 参数说明

无

#### 消息回调

无

### 是否为纯音频模式

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 是否为纯音频模式 |
| 接口 | public boolean isAudioOnlyMode() |
| 返回值 | true 是 false 不是 |

#### 参数说明

无

#### 消息回调

无

## URTCSdkEnv

SDK 环境初始化功能

### 初始化SDK android 环境

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 初始化SDK android 环境 |
| 接口 | public static void initEnv(Context context, Application application) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| context [in] | Android 上下文环境 | Context | N |
| application[in] | SDK所在的application | Application | N |

#### 消息回调

无

### 销毁引擎环境

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 销毁引擎环境 |
| 接口 | public static void unInitEnv() |
| 返回值 | 无 |

#### 参数说明

无

#### 消息回调

无

### 是否支持桌面采集

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 是否支持桌面采集 |
| 接口 | public static boolean isSuportScreenCapture() |
| 返回值 | Boolean true 支持 false 不支持 |

#### 参数说明

无

#### 消息回调

无

### 设置SDK日志级别

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置SDK日志级别 |
| 接口 | public static void setLogLevel(URTCSdkLogLevel level) |
| 返回值 | 无 |

#### 参数说明

无

#### 消息回调

无

### 获取SDK 所在的应用

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取SDK 所在的应用 |
| 接口 | public static Application getsApplication() |
| 返回值 | Application android 应用 |

#### 参数说明

无

#### 消息回调

无

### 设置SDK模式

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置SDK模式 |
| 接口 | public static void setSdkMode(URTCSdkMode mode) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| mode[in] | sdk模式 | URTCSdkMode | N |

#### 消息回调

无

### 获取SDK模式

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 获取SDK模式 |
| 接口 | public static URTCSdkMode getSdkMode() |
| 返回值 | sdk 模式 |

#### 参数说明

无

#### 消息回调

无

### 设置用户APP秘钥key

#### 接口定义

|  |  |
| --- | --- |
| 接口描述 | 设置用户APPID秘钥key |
| 接口 | public static void setTokenSeckey(String seckey) |
| 返回值 | 无 |

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **说明** | **数据类型** | **可空** |
| seckey[in] | App 秘钥key | String | N |

#### 消息回调

无

1. 接口错误表

## 错误码

public enum URTCSdkErrorCode {

NET\_ERR\_CODE\_OK, //成功

NET\_ERR\_GETSIGNAL\_ADDR\_FAIL,// 无法获取接入地址

NET\_ERR\_SAME\_CMD , // 重复的命令

NET\_ERR\_NOT\_IN\_ROOM,// 没有加入房间

NET\_ERR\_ROOM\_JOINED, // 已加入房间

NET\_ERR\_NETWORK\_ERR, // 网络错误

NET\_ERR\_ROOM\_RECONNECTING, // 服务器重连中

NET\_ERR\_PUB\_TOO\_MORE, // 超过房间最大发布流数

NET\_ERR\_STREAM\_PUBED, // 媒体流已发布

NET\_ERR\_STREAM\_NOT\_PUB, // 媒体流没有发布

NET\_ERR\_STREAM\_SUBED, // 媒体流已订阅

NET\_ERR\_STREAM\_NO\_SUB, // 媒体流没有订阅

NET\_ERR\_SUB\_NO\_USER, // 无对应用户

NET\_ERR\_SUB\_NO\_MEDIA, // 无对应媒体流

NET\_ERR\_USER\_LEAVING, // 正在离开房间

NET\_ERR\_SERVER\_CON\_FAIL,// 服务器连接失败

NET\_ERR\_INVAILD\_ST, // 无效的房间状态

NET\_ERR\_SERVER\_DIS, // 房间断开

NET\_ERR\_NO\_HAS\_TRACK, // 没有对应的媒体轨道

NET\_ERR\_MSG\_TIMEOUT, // 消息请求超时

NET\_ERR\_AUTO\_PUB , //自动发布模式

NET\_ERR\_AUTO\_SUB, //自动订阅模式

NET\_ERR\_NOT\_INIT, //SDK没有初始化

NET\_ERR\_NOT\_PUB\_TRACK, //对应的媒体轨道没有启用

NET\_ERR\_INVAILED\_PARGRAM, //无效参数

NET\_ERR\_INVAILED\_WND\_HANDLE, //无效的视频渲染窗口

NET\_ERR\_INVAILED\_MEDIA\_TYPE, //无效的媒体类型

NET\_ERR\_SUB\_ONEMORE, // 重复订阅

NET\_ERR\_NO\_PUB\_ROLE, // 没有发布权限

NET\_ERR\_NO\_SUB\_ROLE, // 没有订阅权限

NET\_ERR\_CAM\_NOT\_ENABLE, // 摄像头没有启用

NET\_ERR\_SCREEN\_NOT\_ENABLE, // 桌面没有启用

NET\_ERR\_AUDIO\_MODE, // 当前为纯音频模式

NET\_ERR\_SECKEY\_NULL,// 无效的秘钥key

NET\_ERR\_GEN\_TOKEN\_FAIL // 生成token 失败

}

# **SDK接入指导**

## 友商接口对比

Ucloud RTC SDK 采用和声网接近的SDK接口设计，采用事件机制编程，方便用户快速上手。

## 开发语言以及系统要求

* 开发语言：java
* 系统要求：android4.4以上版本

## 开发环境

* android studio 需要android SDK

## 搭建开发环境

1. 下载urtc android SDK包，SDK包为aar 格式名称为urtclib-release.aar
2. 将aar 文件拷贝到自己的lib 目录下，然后添加到lib 中

打开工程目录下的 build.gradle，确保已经添加了如下依赖，如下所示：

dependencies {

implementation (name: 'urtclib-release', ext: 'aar')

}

如果项目混淆，请在混淆中添加一下urtc 混淆规则

-keep class com.urtclib.sdkengine.\*\*{\*;}

-keep class com.urtclib.sdkengine.define.\*{\*;}

-keep enum com.urtclib.sdkengine.define.\*{\*;}

-keepclassmembers class com.urtclib.sdkengine.URTCSdkEnv {

public static <methods>;

}

-keepclassmembers interface com.urtclib.sdkengine.URTCSdkEngine {

public <methods>;

public static <methods>;

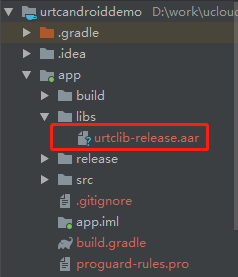
}

-keep class org.webrtc.\*\* {

\*;

}

3 添加成功后，工程大概目录如下：



4 配置权限

<uses-feature android:name="android.hardware.camera" />  
<uses-feature android:name="android.hardware.camera.autofocus" />  
<uses-feature android:glEsVersion="0x00020000" android:required="true" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.RECORD\_AUDIO" />  
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.FLASHLIGHT" />  
<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.WAKE\_LOCK" />  
<uses-permission android:name="android.permission.BLUETOOTH" />

在 Android 6.0 (API 23) 开始，用户需要在应用运行时授予权限，而不是在应用安装时授予，并分为正常权限和危险权限两种类型。在实时音视频 SDK 中，用户需要在进入音视频通话房间前动态申请 CAMERA、RECORD\_AUDIO、WRITE\_EXTERNAL\_STORAGE 权限，具体可参考 [Android 官方文档](https://developer.android.com/training/permissions/requesting?hl=zh-cn)。

## 初始化

### 引擎环境初始化：

主要配置android context sdkmode以及APPid key

public class URTCApplication extends Application {  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 URTCSdkEnv.*initEnv*(getApplicationContext(), this);  
 URTCSdkEnv.*setLogLevel*(URTCSdkLogLevel.*URTC\_SDK\_LogLevelInfo*) ;  
 URTCSdkEnv.*setSdkMode*(URTCSdkMode.*RTC\_SDK\_MODE\_TRIVAL*);  
 URTCSdkEnv.*setTokenSeckey*(CommonUtils.*SEC\_KEY*);  
 WindowManager windowManager = (WindowManager) getSystemService(Context.*WINDOW\_SERVICE*);  
 DisplayMetrics outMetrics = new DisplayMetrics();  
 windowManager.getDefaultDisplay().getMetrics(outMetrics);  
 CommonUtils.*mItemWidth* = outMetrics.widthPixels / 3;  
 CommonUtils.*mItemHeight* = CommonUtils.*mItemWidth*;  
 }  
  
  
}

### 继承实现URTCSdkEventListener 实现事件处理

URTCSdkEventListener eventListener = new URTCSdkEventListener() {  
 @Override  
 public void onServerDisconnect() {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 ToastUtils.*shortShow*(RoomActivity.this, " 服务器已断开");  
 stopTimeShow();  
 onMediaServerDisconnect() ;  
 }  
 });  
 }  
  
 @Override  
 public void onJoinRoomResult(int code, String msg, String roomid) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 if (code == 0) {  
 ToastUtils.*shortShow*(RoomActivity.this, " 加入房间成功");  
 startTimeShow();  
 }else {  
 ToastUtils.*shortShow*(RoomActivity.this, " 加入房间失败 "+  
 code +" errmsg "+ msg);  
 Intent intent = new Intent(RoomActivity.this, ConnectActivity.class);  
 onMediaServerDisconnect() ;  
 startActivity(intent) ;  
 finish();  
 }  
  
 }  
 });  
 }

……

### 获取SDK 引擎 并进行基础配置

sdkEngine = URTCSdkEngine.createEngnine(eventListener) ;

sdkEngine.setAudioOnlyMode(true) ; // 设置纯音频模式  
sdkEngine.configLocalCameraPublish(false) ; // 设置摄像头是否发布  
sdkEngine.configLocalAudioPublish(true) ; // 设置音频是否发布，用于让sdk判断自动发布的媒体类型  
sdkEngine.configLocalScreenPublish(false) ; // 设置桌面是否发布，作用同上

sdkEngine.setStreamRole(URTCSdkStreamRole.*URTC\_SDK\_STREAM\_ROLE\_BOTH*);// 流权限  
sdkEngine.setAutoPublish(true) ; // 是否自动发布  
sdkEngine.setAutoSubscribe(true) ;// 是否自动订阅  
sdkEngine.setVideoProfile(URTCSdkVideoProfile.*matchValue*(mVideoProfile)) ;// 摄像头输出等级

## 加入房间

URTCSdkAuthInfo info = new URTCSdkAuthInfo() ;  
info.setmAppId(mAppid);  
info.setmToken(mRoomToken);  
info.setmRoomid(mRoomid);  
info.setmUid(mUserid);  
Log.*d*(*TAG*, " roomtoken = " + mRoomToken) ;  
sdkEngine.joinChannel(info)

## 自动/手动发布

如果配置了自动发布无需调用发布视频接口，SDK会在用户成功加入房间后自动发布，只需要监听事件调用渲染接口即可。

如果配置了手动发布需要调用sdkEngine引擎的publish接口 配置手动/自动发布

sdkEngine.setAutoPublish(mPublishMode == CommonUtils.*AUTO\_MODE* ? true : false);

### 媒体发布类型

现在的类型包括两大类，需要传入publish接口的mtype,hasvideo,hasaudio参数各不相同，混合类型是单一类型的组合，具体代码可参阅urtcdemo的RoomActvity中的处理。

1. 混合类型

* 音频+屏幕捕捉
* 视频+屏幕捕捉

1. 单一类型

* 音频 （mtype:urtc\_sdk\_media\_type\_video,hasvideo:false,hasaudio:true）
* 视频 （mtype:urtc\_sdk\_media\_type\_video,hasvideo:true,hasaudio:true）
* 屏幕捕捉 （mtype:urtc\_sdk\_media\_type\_screen,hasvideo:true,hasaudio:false）

### 手动发布媒体流

sdkEngine.publish(URTCSdkMediatype mtype, boolean hasvideo, boolean hasaudio)

回调事件

public void onLocalPublish(int code, String msg, URTCSdkStreamInfo info)

### 渲染媒体流

在onLocalPublish 回调成功后，再函数中可以调用视频渲染

localrenderview.setBackgroundColor(Color.*TRANSPARENT*);  
sdkEngine.startPreview(info.getmMediatype(), localrenderview);

//不想渲染时可以调用停止渲染接口

sdkEngine.stopPreview(URTCSdkMediatype mediatype)

### 取消发布媒体流

sdkEngine.unPublish(URTCSdkMediatype mtype)

//回调事件

public void onLocalUnPublish(int code, String msg, URTCSdkStreamInfo info)

## 订阅

如果配置了自动发布无需调用发布视频接口，SDK会自动发布，只需要监听事件调用渲染接口即可。

### 订阅媒体流

sdkEngine.subscribe(URTCSdkStreamInfo info)

//回调事件

public void onSubscribeResult(int code, String msg, URTCSdkStreamInfo info)

### 渲染媒体流

在onSubscribeResult回调成功后，再函数中可以调用视频渲染

sdkEngine. startRemoteView(URTCSdkStreamInfo info, URTCSdkSurfaceVideoView renderview)

//不想渲染时可以调用定制渲染接口

sdkEngine.stopPreview(URTCSdkMediatype mediatype)

### 取消订阅媒体流

sdkEngine. subscribe(URTCSdkStreamInfo info)

//回调事件

public void onUnSubscribeResult(int code, String msg, URTCSdkStreamInfo info)

## 权限控制

权限分为发布，订阅，全部权限，全部权限包括了发布和订阅

//接口

public int setStreamRole(UCloudRtcSdkStreamRole role)

//调用

sdkEngine.setStreamRole(mRole);

## 离开房间

sdkEngine.leaveChannel() ;

## 编译运行 运行demo