场景融合
视频直播
Android 2.X

2024-08-30

视频直播概述

更新时间:2024-08-30

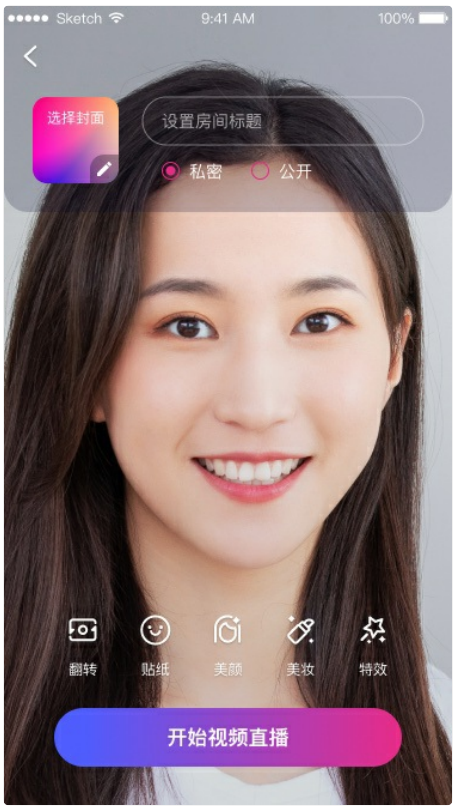
RCLiveVideoLib 是融云针对视频直播场景设计的 SDK，API 设计参考了市面主流视频直播 App 的功能和场景特点。SDK 融合 **RongIMLib** 和 **RongRTCLib** 实现视频直播场景，将复杂逻辑（订阅和发布流、上下麦、连麦布局等）进行了封装，降低开发者的接入难度。相较于开发者自己接入低延迟直播和即时通讯，SDK 具备以下优势：

1. **贴近业务：** API 调用和命名贴近视频直播业务场景与客户端功能特性；
2. **使用简单：** 核心 API 数量少（房主 3 个、观众 2 个和常规操作 13 个），学习成本低，可快速上手；
3. **扩展性强：** 支持房间自定义属性，提高扩展性，不管是游戏直播、社交直播，还是电商直播等场景均可覆盖；
4. **资源丰富：** 包含丰富的文档和全功能的线上开源项目 RCRTC，不管是开发还是上线，都有充足的参考资源。

主要功能

视频直播： RCLiveVideoLib 只需两步即可开启直播：

1. 准备视频直播：设置视频信息，输出视频流，用于视频流预处理，比如：美颜；
2. 开启视频直播：加入房间并发布视频直播流。
3. 可以根据自身需要选择官方美颜或相芯美颜。
   1. 官方美颜：提供基础的美颜能力，包括美妆及特效，可通过 RTCLib SDK 直接使用官方美颜插件。
   2. 相芯美颜：Demo 中已直接集成 RTCLib 相芯美颜插件并展示相芯美颜能力（试用）。正式使用相芯美颜能力需单独付费。购买咨询详见云市场相芯美颜特效页面。插件使用详见 RTCLib 文档美颜处理 · 相芯美颜插件。



视频连麦： RCLiveVideoLib 支持邀请上麦、申请上麦、自由上麦等连麦方式

连麦布局： SDK 封装了 7 种连麦布局，并支持自定义布局

跨房间 PK： 基于连麦布局，视频直播支持跨房间连麦进行 PK

房间管理： 视频直播包含丰富的房间属性，RCLiveVideoLib 支持自定义属性

功能列表

RCLiveVideoLib 的功能包括但不限于以下内容：

功能	描述	支持版本
视频流预处理	直播推流前可以对视频流进行处理，比如：美颜	2.0.0

功能	描述	支持版本
支持1对1连麦	连麦用户支持小窗事件点击和区域位置通知	2.0.0
内置 7 种连麦布局	一键切换连麦布局	2.0.0
支持自定义连麦布局	支持自定义任意连麦布局	2.0.0
申请连麦	轻松处理申请机制，特定用户允许才可上麦	2.0.0
取消连麦申请	观众取消连麦申请，房主接收回调	2.0.0
同意连麦申请	房主同意连麦申请，并与观众开始连麦	2.0.0
拒绝连麦申请	房主拒绝连麦申请，观众接收回调	2.0.0
查询连麦申请	查询当前房间上麦申请信息	2.0.0
邀请上麦	房主邀请观众进行视频连麦	2.0.0
取消上麦邀请	房主取消连麦邀请，观众接收到回调	2.0.0
同意上麦邀请	观众同意连麦邀请，并与房主开始连麦	2.0.0
拒绝上麦邀请	观众拒绝连麦邀请，房主接收到回调	2.0.0
抱用户下麦	将指定用户抱下麦位	2.0.0
自定义房间属性	丰富的房间扩展属性，轻松构建不同类型的视频直播房间	2.0.0
发起 PK 邀请	邀请其他房间主播连麦进行 PK	2.1.0
取消 PK 邀请	取消以发起 PK 邀请	2.1.0
接受 PK 邀请	接受其他房间主播发起的 PK 邀请	2.1.0
拒绝 PK 邀请	拒绝其他房间主播发起的 PK 邀请	2.1.0

开通融云 CDN 服务

更新时间:2024-08-30

欢迎前往融云官网了解直播 CDN 产品。本文介绍如何开始配置融云 CDN 服务。

开通服务

融云 CDN 是付费增值服务，且需要开通后才能使用。您可以访问控制台的融云 CDN 页面开通服务。

开通融云 CDN 服务后，可看到域名配置、证书管理、防盗链配置。

选择推拉流模式

融云 CDN 支持三种不同的推拉流地址模式：

- **开播自动推流**：在后台配置后，所有直播房间的直播流都会自动推流到 CDN，观众可在客户端加入房间时获取 CDN 信息，或者使用客户端 SDK 提供的监听房间内 CDN 资源的方法。如果您希望直接从融云服务端获取拉流地址，可从 App 服务端调用融云服务端 API，详见获取融云 CDN 拉流地址。
- **开播手动推流**：在直播主播开始推流后，您需要根据产品设计决定何时调接口让融云服务开始或停止推流到 CDN。观众可在客户端加入房间时获取 CDN 信息，或者使用客户端 SDK 提供的监听房间内 CDN 资源的方法。如果您希望直接从融云服务端获取拉流地址，可从 App 服务端调用融云服务端 API，详见获取融云 CDN 拉流地址。
- **自行生成推拉流地址**：融云服务端不负责生成推拉流地址，您需要在您的应用服务器自行生成 CDN 推拉流地址。

自行拼接推拉流地址

在自行生成推拉流地址模式下，可自行拼接生成 CDN 推拉流地址。需要拼接的字段包括推/拉流域名、自定义的 {AppName}

和自定义的 {StreamName}。您还可以通过在 CDN 拉流地址中添加分辨率、码率参数 {with}、{height}、{fps}，拉取 CDN 转码流。

CDN 推流地址拼接规则

融云 CDN 仅支持 RTMP 协议的推流：

RTMP：`rtmp://推流域名/{AppName}/{StreamName}`

拼接推流地址后，可调用客户端 SDK 或服务端 API 的旁路推流接口，传入 CDN 推流地址进行推流。

CDN 拉流地址拼接规则

融云 CDN 支持使用 RTMP、FLV、HLS 协议进行拉流。

如需使用 CDN 流的原始分辨率、帧率：

- **RTMP**：`rtmp://拉流域名/{AppName}/{StreamName}`
- **FLV**：`http(s)://拉流域名/{AppName}/{StreamName}.flv`
- **HLS**：`http(s)://拉流域名/{AppName}/{StreamName}.m3u8`

如需拉取不同分辨率的 CDN 直播流（拉取非原始分辨率、码率的流会触发 CDN 转码服务，产生转码费用）：

- **RTMP**：`rtmp://拉流域名/{AppName}/{StreamName}_{with}_{height}_{fps}`
- **FLV**：`http(s)://拉流域名/{AppName}/{StreamName}_{with}_{height}_{fps}.flv`
- **HLS**：`http(s)://拉流域名/{AppName}/{StreamName}_{with}_{height}_{fps}.m3u8`

{with}、{height} 参数为要拉取的 CDN 转码流的分辨率宽高，参考下方枚举值。{fps} 为帧率参数，支持 10/15/24/30。

宽高限制列表：

```
[
"176*144", "180*180", "256*144", "240*180", "320*180", "240*240", "320*240",
"360*360", "480*360", "640*360", "480*480", "640*480", "720*480", "848*480",
"960*720", "1280*720", "1920*1080", "144*176", "144*256", "180*240", "180*320",
"240*320", "360*480", "360*640", "480*640", "480*720", "480*848", "720*960",
"720*1280", "1080*1920"
]
```

例如，拉取 flv 协议的 CDN 流，并指定宽x高为 720*920，帧率为 15，则拼接后的地址如下：

`http(s)://yourdomain/apppkey/roomid_720_960_15.flv`

推荐使用融云 CDN 播放器进行播放。

> **注意：**

- 如在控制台启用了防盗链，则您拼接的推拉流地址中还必须携带防盗链信息。防盗链地址的具体拼接规则请参见下文「防盗链配置」。
- 2022 年 6 月前开通融云 CDN 服务的客户，如获取到的 HLS 拉流地址无法正常播放，请提交工单咨询。

域名配置

在域名配置标签下配置并保存了推流域名和拉流域名后，会看到完整的配置界面。

配置推拉流域名

推拉流域名都需要填写二级域名，请确保此域名没有没有在别的 App Key 下配置过。

- 域名是成对的，推流和拉流有绑定关系，新增和修改时需要一并修改。
- 您需要确保一级域名已经备案过。如果因域名没有备案或涉及非法业务等原因导致推流或者拉流域名不可用，由此产生的任何后果由您自身承担。

推流与拉流域名配置生效后，系统会自动给该域名分配一个 CNAME。使用您的推流或拉流域名进行直播之前，需要需要您公司开发或服务运维人员在域名解析配置中添加对应的 CNAME 记录，让您的域名指向 CNAME。具体如何配置可以咨询域名供应商。

HTTPS 设置

HTTPS 设置是可选项。FLV/HLS 协议的直播流默认使用 HTTP，配置 SSL 证书后可以使用 HTTPS。

如果您没有在融云平台上传过证书，请点击**新增证书**，将证书上传。下图展示了提交证书的界面。

防盗链

防盗链配置是可选项。配置后会提升推拉流域名的安全。

开启后一定要配置推流和拉流地址有效期时间和加密 URL 的密文 KEY。

防盗链开启后，如果观众端遇到弱网（融云 SDK 内部帮您在网络恢复后重新订阅成功）会比没有防盗链多出一些恢复订阅时间。

自行生成防盗链推拉流地址

如果您选择的推拉流模式为**自行生成推拉流地址**，在您的服务端生成推拉流地址时在地址中加入防盗链相关信息。

相比普通推拉流地址，生成防盗链推拉流地址还需要用到以下防盗链参数：

防盗链参数	描述	补充说明
wsTime	生成推拉流地址时的 UNIX 时间。防盗链地址中直接携带该参数。	格式为 16 进制 UNIX 时间。防盗链地址中直接携带该参数。
KEY	MD5 计算方式的密钥，在控制台配置，仅用于计算 `wsSecret`。	可以自定义。
wsSecret	播放 URL 中的加密参数。防盗链地址中直接携带该参数。	值是通过将 `KEY`，`URI`，`wsTime` 依次拼接的字符串进行 MD5 加密算法得出，即 wsSecret = MD5（wsTime+URI+KEY）。

防盗链的推拉流地址生成示例：

假设原始 url（已包含推/拉流域名 + AppName + StreamName）为：http://cdn.rongcloud.cn/myappname/stream.flv

1. 获取在控制台**防盗链配置**填入的 KEY，假设 KEY 为：`rongcloud`。

2. 获取 wsTime，即生成推拉流地址时的 UNIX 时间，假设为 1601026312，转换成 16 进制5f6db908。在计算 wsSecret 时需要用到。防盗链 URL 中也会直接携带该参数。

3. 计算 wsSecret。防盗链 URL 中会直接携带该参数。

    1. 获取计算 `wsSecret` 需要用到的 `URI` 参数值。需要用到您自定义的 AppName 与 StreamName。拼接规则为 `/{AppName}/{StreamName}`。从本例的原始 URL 可得出 URI 为 `/app/stream.flv`。
    2. 依次拼接 `wsTime` + `URI` + `KEY`，获取签名字符串。在本例中该拼接字符串为：5f6db908/app/stream.flvrongcloud
    3. 对签名字符串计算 md5hash，得到 `wsSecret`。即，`wsSecret` = md5sum("5f6db908/app/stream.flvrongcloud") = 79aead3bd7b5db4adeffb93a010298b5

4. 使用上述步骤中得到的 wsSecret 与 wsTime 参数，拼接成防盗链 URL：

    http://cdn.rongcloud.cn/app/stream.flv?wsSecret=79aead3bd7b5db4adeffb93a010298b5&wsTime=5f6db908

当用户发起带时间戳防盗链的 url 请求后，CDN 服务器会对 url 内容进行校验，判断时间有效性及 MD5 值，两个值其中有一个不符合要求，返回 403。

生成防盗链代码示例（golang）

```
package main

import (
"crypto/md5"
"encoding/hex"
"fmt"
"strconv"
"strings"
"time"
)

func main() {
secret, t := GenWsSecret("/live/abc", "key")
fmt.Println(secret)
fmt.Println(t)
}

func GenWsSecret(uri, key string) (string, string) {
uri = fmt.Sprintf("/%s", strings.TrimLeft(uri, "/"))

t := strconv.FormatInt(time.Now().Unix(), 16)

ori := fmt.Sprintf("%s%s%s", t, uri, key)

h := md5.New()
h.Write([]byte(ori))

return hex.EncodeToString(h.Sum(nil)), t
}
```

> **注意事项：**
>
> 1. 在 20：00 至次日 2：00 进行的配置不会立即生效，会在 2 点后依次配置生效。在此时间段外进行配置会

在 30 分钟后生效。

2. 新配置生效后，原有的配置即刻作废。建议开发者避开业务高峰时段，并给还在使用老配置的房间用户发送提醒通知，重新订阅新地址。

快速集成 环境要求

更新时间:2024-08-30

- **Android Studio**：4.0+
- **Android**：5.0 及以上
- **Java**：java 8

开通音视频服务

您在融云创建的应用默认不会启用音视频服务。在使用融云提供的任何音视频服务前，您需要前往控制台，为应用开通音视频服务。

使用语聊房业务要求开通「音视频直播」服务，具体步骤请参阅 开通音视频服务 。

> 服务开通、关闭等设置完成后 30 分钟后生效。

集成直播房 SDK

- gradle 远程依赖

远端依赖

1. 在 project 的 build.gradle 添加融云仓库

```
allprojects {
repositories {
// 融云 maven 仓库
maven { url "https://maven.rongcloud.cn/repository/maven-releases/" }
}
}
```

2. module 的 build.gradle 文件中的 dependencies 节点添加如下代码：

```
dependencies {
......
// 此处以 IMLib 为例，建议使用最新版本。
implementation 'cn.rongcloud.sdk:im_lib:5.1.7'
// RTCLib，音视频能力库
implementation 'cn.rongcloud.sdk:rtc_lib:5.1.14'
// 融云 CDN 插件 + 混音网络资源文件（可选）。插件版本必须和 RTCLib 版本保持一致。
implementation 'cn.rongcloud.sdk:player:5.1.14'
// 视频直播 SDK，注意替换为最新版本
implementation 'cn.rongcloud.sdk:liveroom_lib:2.1.0.8'
......
}
```

3. **gradle 配置完成后，重新** build **项目。**

混淆配置

- **根据实际项目的依赖情况，选项对应的混淆配置：**
  1. **项目中依赖** imlib **和** rtclib**，则添加** imlib**，**rtc **和 视频直播** SDK **的混淆配置即可**
  2. **项目中依赖** imkit **和** rtclib**，则添加** imkit**，**rtc **和 视频直播** SDK **的混淆配置即可**
- **混淆配置有交叉重复的部分，可适当去重。**

1. IMLib **的混淆配置**

```
# imkit 的混淆配置
-keepattributes Exceptions,InnerClasses
-keepattributes Signature
-keep class io.rong.** {*;}
-keep class cn.rongcloud.** {*;}
-keep class * implements io.rong.imlib.model.MessageContent {*;}
-dontwarn io.rong.push.**
-dontnote com.xiaomi.**
-dontnote com.google.android.gms.gcm.**
-dontnote io.rong.**

# 当代码中有继承 PushMessageReceiver 的子类时，需 keep 所创建的子类广播
# 把 io.rong.app.DemoNotificationReceiver 改成子类广播的完整类路径即可.
-keep class io.rong.app.DemoNotificationReceiver {*;}

-ignorewarnings
```

2. IMKit **的混淆配置**

```
# imkit 的混淆配置
-keepattributes Exceptions,InnerClasses
-keepattributes Signature
-keep class io.rong.** {*;}
-keep class cn.rongcloud.** {*;}
-keep class * implements io.rong.imlib.model.MessageContent {*;}
-dontwarn io.rong.push.**
-dontnote com.xiaomi.**
-dontnote com.google.android.gms.gcm.**
-dontnote io.rong.**

# 下方混淆使用了Location包时才需要配置，可参考高德官网的混淆方
式:https://lbs.amap.com/api/android-sdk/guide/create-project/dev-attention
-keep class com.amap.api.**{*;}
-keep class com.amap.api.services.**{*;}
-keep class com.autonavi.**{*;}

# 当代码中有继承 PushMessageReceiver 的子类时，需 keep 所创建的子类广播
# 把 io.rong.app.DemoNotificationReceiver 改成子类广播的完整类路径即可.
-keep class io.rong.app.DemoNotificationReceiver {*;}

-ignorewarnings
```

3. RTC 混淆配置

```
# rtc 混淆配置
-keepattributes Exceptions,InnerClasses
-keepattributes Signature

-keep class io.rong.** {*;}
-keep class cn.rongcloud.** {*;}
-keep class * implements io.rong.imlib.model.MessageContent {*;}
-dontwarn io.rong.push.**
-dontnote com.xiaomi.**
-dontnote com.google.android.gms.gcm.**
-dontnote io.rong.**

# 下方混淆使用了Location包时才需要配置，可参考高德官网的混淆方
式:https://lbs.amap.com/api/android-sdk/guide/create-project/dev-attention
-keep class com.amap.api.**{*;}
-keep class com.amap.api.services.**{*;}
-keep class com.autonavi.**{*;}

# 当代码中有继承 PushMessageReceiver 的子类时，需 keep 所创建的子类广播
# 把 io.rong.app.DemoNotificationReceiver 改成子类广播的完整类路径即可.
-keep class io.rong.app.DemoNotificationReceiver {*;}

-ignorewarnings
```

4. 视频直播的混淆配置

```
# liveroom 混淆配置
-keep class cn.rongcloud.liveroom.api.** {*;}
-keep class cn.rongcloud.liveroom.manager.** {*;}
-keep class cn.rongcloud.liveroom.utils.** {*;}
-keep class cn.rongcloud.liveroom.weight.** {*;}
-keep class cn.rongcloud.liveroom.messager.** {*;}
```

集成后的包大小

集成视频直播 SDK 后，会增加一定的包大小。

- Android 平台： 增量大约 7 MB（包含 IMLib、RTCLib 和 player 依赖库）
- IOS 平台： 增量大约 200 KB（不包含依赖库）

版本依赖说明

RCLiveRoomLib 依赖融云 IMLib、RTCLib 和 player，依赖版本如下。

依赖组件	版本
IMLib	5.1.7 及以上
RTCLib	5.1.15 及以上
player	必须和 RTCLib 保持一致

初始化

更新时间:2024-08-30

视频直播 SDK 的初始化依赖于融云 IMLib 或者 IMKit 的初始化。

RCLiveRoomLib 初始化

> 在初始化前，请确保已完成以下操作：
>
> - 您已开通融云开发者账号，并申请了融云 App Key。
> - 您已为 App Key 开通音视频服务。使用视频直播 SDK 业务要求开通「音视频直播」服务。
> - 建议在 Application 中初始化，视频直播 SDK 依赖于 IM 的初始化，您可以使用 IMLib 或 IMKit 初始化，具体可以根据当前项目的实际依赖选择使用。

```
/**
* 初始化 AppKey
* 注意：直播房依赖于 IM 的初始化，您可以使用 im_lib 或 im_kit 初始化，具体可以根据当前项目的实际依赖选择使用 imlib 还是
imkit。
*
* @param context 当前应用的 application
* @param appKey 开发者申请的 AppKey
*/
void init(Application context, String appKey);

// imlib 初始化代码示例
String process = UIKit.getCurrentProcessName();
if (!getPackageName().equals(process)) {
// 非主进程不初始化 避免过度初始化
return;
}
Log.d(TAG, "initLiveRoom:process : " + process);
RongCoreClient.init(this, APP_KEY);
```

连接融云服务

> 视频直播 SDK 依赖于 IMLib 或 IMKit。您可以使用 IMLib 或 IMKit 的连接方法，具体可以根据当前项目的实际依赖选择使用。

```
// imlib 中 RCCoreClient 的连接接口说明
/**
* 连接融云服务器
* 注意：如果使用 RCCoreClient
*
* @param appToken 从服务器获取的 imToken
* @param connectCallback 结果回调
*/
RongCoreClient connect(String appToken, ConnectCallback connectCallback)

// 具体调用示例
RongCoreClient.connect(accout.getToken(), new IRongCoreCallback.ConnectCallback() {
@Override
public void onSuccess(String t) {
KToast.show("connect success");
}

@Override
public void onError(IRongCoreEnum.ConnectionErrorCode e) {
String info = "connect fail：\n【" + e.getValue() + "】" + e.name();
Log.e("ConnectActivity", info);
KToast.show(info);
setTitle(accout.getName() + " 连接失败");
}

@Override
public void onDatabaseOpened(IRongCoreEnum.DatabaseOpenStatus code) {

}
}
);
```

开启和结束视频直播

更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 开启和结束视频直播。

注意：SDK 不具备创建和关闭视频直播房间的功能，请调用业务服务器接口实现。

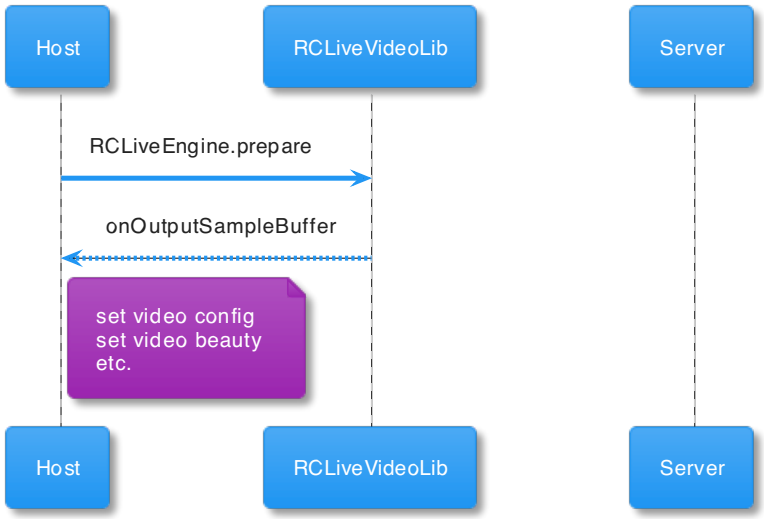
相关调用主要包括：

准备视频直播 · 开启视频直播 · 结束视频直播

准备视频直播

准备直播，时序图参考右边：

- Host 主持人(管理员)
- Audience 观众
- RCLiveVideoLib SDK



设置视频信息

根据业务需求设置视频信息：分辨率、码率和帧率等。

```
///用户可自定义，如果不设置的话，那么默认值会采用以下示例代码的
RCRTCVideoStreamConfig config = RCRTCVideoStreamConfig.Builder
.create()
.setMinRate(250)
.setMaxRate(2200)
.setVideoFps(RCRTCParamsType.RCRTCVideoFps.Fps_15)
.setVideoResolution(RCRTCParamsType.RCRTCVideoResolution.RESOLUTION_720_1280)
.build();
// 通过此方法设置，会覆盖 SDK 内部默认参数（可选）
RCLiveEngine.getInstance().setVideoConfig(config, callback);
// 此方法仅更改主播发流的分辨率（可选）
RCLiveEngine.getInstance().setVideoResolution(RCRTCParamsType.RCRTCVideoResolution.RESOLUTION_720_1280,
callback);
// 此方法仅更改主播发流的帧率（可选）
RCLiveEngine.getInstance().setVideoFps(RCRTCParamsType.RCRTCVideoFps.Fps_15, callback);
```

设置直播间监听方法

用于监听视频直播相关的业务回调（二次视频流处理美颜、布局代理等）。

```
/**
* 设置房间事件监听
* 注意：离开房间后设置监听回置空，因此加入房间时需要重新设置房间监听
* 建议：先设置监听在加入房间。
*
* @param listener 事件监听
*/
RCLiveEngine.getInstance().setLiveEventListener(this);
```

添加视频预览

将视频直播画面view添加到要显示的控制器视图，准备。

```
//获取预览 videoView
RCLiveView videoView = RCLiveEngine.getInstance().preview();
//获取到容器
flLiveView = (FrameLayout) getView().findViewById(R.id.fl_live_view);
flLiveView.removeAllViews();
//将 videoView 绑定到容器里面
videoView.attachParent(flLiveView, null);
```

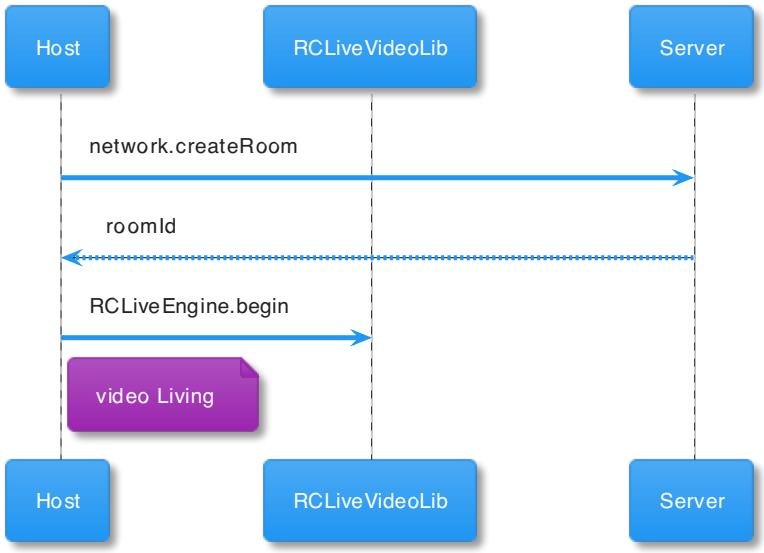
处理视频流输出回调

视频输出回调，可以在此接口做视频流二次开发，例如：美颜。

```
/**
* 处理美颜
*
* @param frame 视频流采样数据
*/
@Override
public void onOutputSampleBuffer(RCRTCVideoFrame frame) {
//示例代码
int render = MhDataManager.getInstance().render(frame.getTextureId(), frame.getWidth(),
frame.getWidth());
frame.setTextureId(render);
}
```

开启视频直播

> 开启直播，时序图参考右边：



客户业务服务器创建房间

调用业务服务器的接口创建一个房间，获得 roomId。

```
      if (result.ok()){
        // 创建直播房间成功，开启直播，需要传入roomId，roomId为业务服务器返回的对象解析得到
        String roomId= result.get();
        // 开启直播(如需向三方 CDN 推流则传 publishURL，否则不传或传 null)
        RCLiveEngine.getInstance().begin(roomId, publishURL, new RCLiveCallback(){
          .......
        });
      }

    }
    @Override
    public void onError(int code, String msg) {
      ///创建直播房间失败
    }
  });
```

开播后添加三方 CDN 推流地址

开始直播后可随时添加三方 CDN 推流地址，需要注意以下几点：

1. 必须开通音视频服务和直播服务。
2. 设置的 CDN 地址不能为空。
3. 最多设置 5 个 CDN 地址，超出会抛出 RTCErrorCode.RongRTCCodeCDNCountReachToLimit 错误。
4. 如果多次设置相同的地址，会直接返回成功。

```
// 添加三方 CDN 推流地址
RCLiveEngine.getInstance().addCDNPublishStream(publishURL, callback);
```

移除配置过的直播 CDN 地址

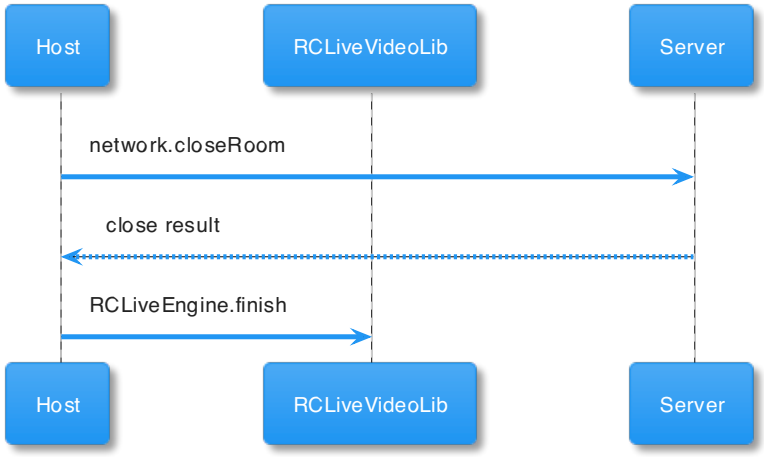
开始直播成功之后，主播可选择移除一个设置过的 CDN 推流地址，有以下几点要求：

1. 必须开通音视频服务和直播服务。
2. 移除的 CDN 地址不能为空。
3. 如果移除的地址，之前没有设置过，会直接返回成功。

```
// 移除三方 CDN 推流地址
RCLiveEngine.getInstance().removeCDNPublishStream(publishURL, callback);
```

结束视频直播

> 关闭直播，时序图参考右边：



客户业务服务器处理

调用业务服务器接口关闭房间。

```
///调用业务服务器关闭房间接口，让服务端去删除房间
OkApi.get(VRApi.deleteRoom(mVoiceRoomBean.getRoomId()), null, new WrapperCallBack() {
@Override
public void onResult(Wrapper result) {

}

@Override
public void onError(int code, String msg) {

}
});
```

房间被关闭,房间被销毁接口方法回调

在 onRoomDestroy 中取消监听和注销房间资源等。

```
@Override
public void onRoomDestroy() {
//房间被销毁，处理资源

}
```

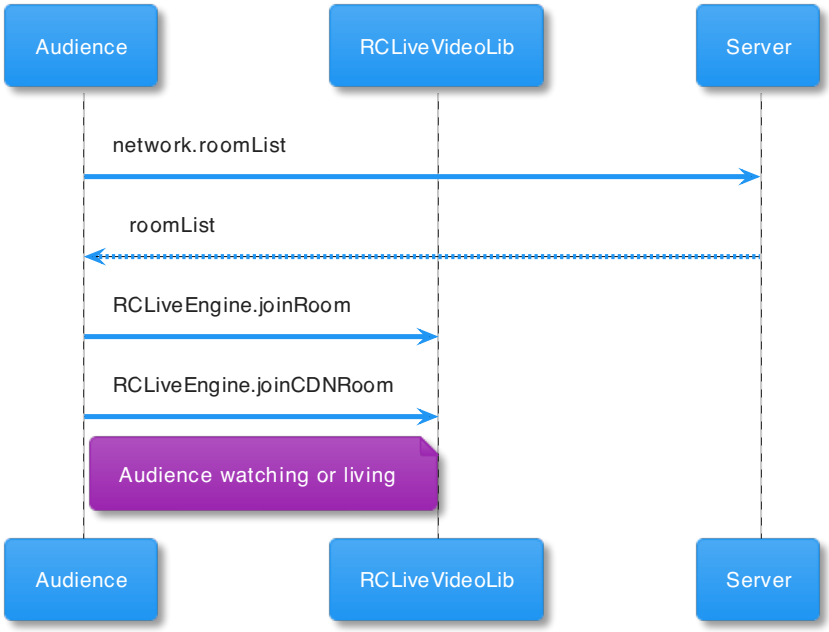
加入和离开视频直播

更新时间:2024-08-30

本节介绍观众端如何使用 RCLiveVideoLib 加入和离开视频直播。

加入视频直播

观众仅可加入已存在的视频直播房间，可订阅低延迟视频流或 CDN 视频流，分别对应不同的加入房间接口



设置直播间监听方法

用于监听视频直播相关的业务回调

```
///可根据需求设置相对于的监听方法
//设置房间事件监听
RCLiveEngine.getInstance().setLiveEventListener(this);
//设置连麦事件监听
RCLiveEngine.getInstance().getLinkManager().setLiveLinkListener(this);
//设置麦位变化监听
RCLiveEngine.getInstance().getSeatManager().setLiveSeatListener(this);
```

加入房间并订阅低延迟视频流

观众通过订阅低延迟视频流的方式加入房间，调用接口如右边所示：

```
      //拿到视频直播画面View
      RCLiveView videoView=RCLiveEngine.getInstance().preview();
      //拿到容器view
      flLiveView = (FrameLayout) getView().findViewById(R.id.fl_live_view);
      flLiveView.removeAllViews();
      //将视频画面添加到容器中呢
      videoView.attachParent(flLiveView, null);

    }
    @Override
    public void onError(int code, RCLiveError error) {
      //加入房间失败
    }
  });
```

已复制

复制以上代码

订阅融云 CDN

以订阅 CDN 视频流的方式加入视频直播。

> 准备工作：
>
> - 开通融云 CDN 服务。
> - 依赖融云 CDN 插件 `player`。

可选择订阅默认 CDN 视频流 或 设置订阅流的分辨率和帧率。

加入房间并订阅融云 CDN 视频流

加入房间，订阅 CDN 拉流时，观众端默认按 CDN 服务输出的视频格式拉流。

您也可以设置 CDN 订阅流的分辨率和帧率
要注意这种情况，需要提交工单，客服会为您开通转码服务。

```
    //拿到视频直播画面View
    RCLiveView videoView = RCLiveEngine.getInstance().preview();
    //拿到容器view
    flLiveView = (FrameLayout) getView().findViewById(R.id.fl_live_view);
    flLiveView.removeAllViews();
    //将视频画面添加到容器中呢
    videoView.attachParent(flLiveView, null);

  }
  @Override
  public void onError(int code, RCLiveError error) {
    //TODO code
  }
});
```

已复制

复制以上代码

订阅三方 CDN

开发者如果不使用融云 CDN，可以自行接入的第三方 CDN 服务。如果开播时传入了三方 CDN 地址，则观众端加入房间时可以订阅您自行接入的 CDN 服务地址。

> 注意：您需要实现 CDN 播放器，并通过第三方 CDN 服务的拉流地址自行实现播放逻辑。

构建自己的播放器

开发者需实现 ICDNPlayer 接口，将自己的 VidoView 返回给 SDK。之后可以通过 start(String roomId) 和 stop() 的回调控制播放和停止。具体实现可参见 RC RTC Demo 项目。

```
// 这里继承了 ViewGroup 并实现了 ICDNPlayer 接口
public class ThirdCDNPlayer extends FrameLayout implements ICDNPlayer {
// 这里采用的是七牛的播放器
private PLVideoView mVideoView;

public ThirdCDNPlayer(Context context) {
this(context, null);
}

public ThirdCDNPlayer(Context context, AttributeSet attrs) {
this(context, attrs, 0);
}

public ThirdCDNPlayer(Context context, AttributeSet attrs, int defStyleAttr) {
this(context, attrs, defStyleAttr, 0);
// 初始化播放器
initVideoView();
}

private void initVideoView() {
// 初始化 VideoView 并添加到当前 ViewGroup 中
mVideoView = new PLVideoView(mContext);
this.addView(mVideoView);
......
}

@Override
public View getView() {
// 返回当前view，供SDK内部使用
return this;
}

// SDK 内部调用了开始
@Override
public void start(String roomId) {
// 设置拉流地址
mVideoView.setVideoPath(pullURL);
// 开始播放
mVideoView.start();
}
// SDK 内部调用停止播放，一般是退出直播房间时回调
@Override
public void stop() {
```

```
public void stop() {
mVideoView.stop();
mVideoView = null;
}

// 这个方法是视频 view 裁切示例
// 如果多人连麦时画面显示区域可能是不规则的，通过该方法根据麦位布局把黑色部分裁切掉，可以露出房间背景
@Override
protected void dispatchDraw(Canvas canvas) {
int realWidth = getMeasuredWidth();
int realHeight = getMeasuredHeight();
// 画布裁剪，根据麦位摆放把视频不规则黑色部分裁掉，以露出背景
RCRect[] rcRects = LayoutManager.get().getFrames();
Path path = new Path();
for (int i = 0; i < rcRects.length; i++) {
RCLiveSeatInfo rcLiveSeatInfo = SeatManager.get().getSeatByIndex(i);
if (null == rcLiveSeatInfo || TextUtils.isEmpty(rcLiveSeatInfo.getUserId()) ||
!rcLiveSeatInfo.isEnableVideo()) {
continue;
}
Rect rect = WeightUtil.getClipRect(rcRects[i], realWidth, realHeight, 0, 0);
path.moveTo(rect.left, rect.top);
path.lineTo(rect.right, rect.top);
path.lineTo(rect.right, rect.bottom);
path.lineTo(rect.left, rect.bottom);
path.close();
}
canvas.clipPath(path);
super.dispatchDraw(canvas);
}
}
```

加入房间并设置直播类型

设置 RCLiveInfo.RCLiveType.THIRD_CDN 后，SDK内部直播 VideoView 则不再生效，通过设置 setThirdCDNPlayer 方法将自己实现的 VideoView 传入到 SDK 内部。

```
    //拿到视频直播画面View
    RCLiveView videoView = RCLiveEngine.getInstance().preview();
    //拿到容器view
    flLiveView = (FrameLayout) getView().findViewById(R.id.fl_live_view);
    flLiveView.removeAllViews();
    //将视频画面添加到容器中呢
    videoView.attachParent(flLiveView, null);

  }
  @Override
  public void onError(int code, RCLiveError error) {
    //TODO code
  }
});
```

已复制

复制以上代码

离开视频直播

观众离开视频直播房间，不再观看视频直播。

调用leaveRoom

观众离开视频直播房间需调用 leaveRoom 接口。

```
/// 离开视频直播房间
public void leaveRoom(LeaveRoomCallBack callback) {
RCLiveEngine.getInstance().leaveRoom(new RCLiveCallback() {
@Override
public void onSuccess() {
//离开房间成功，可以做一些反注册之类的操作
}

@Override
public void onError(int code, RCLiveError error) {
//离开失败，可以根据错误信息来做操作
}
});
}
```

加入和结束连麦

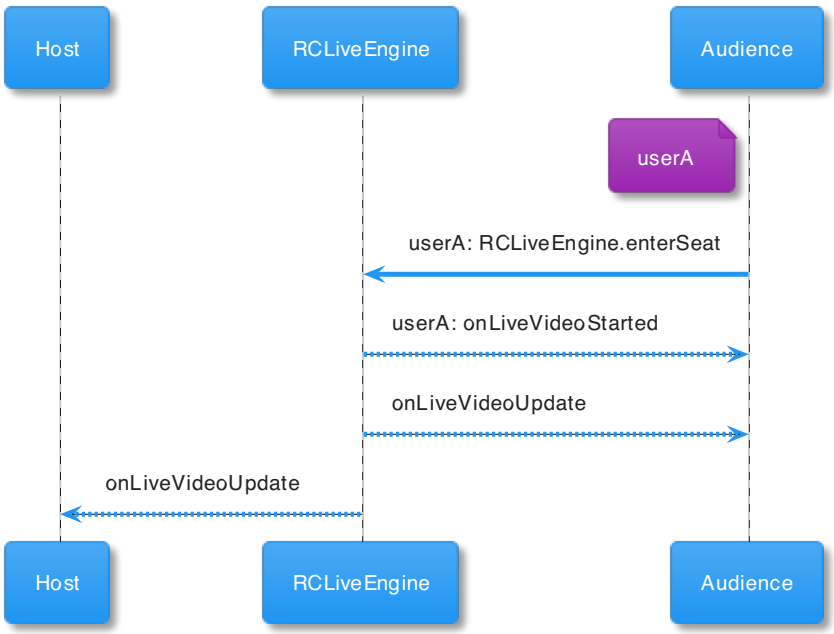
更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 主动上麦、主动下麦、踢人下麦。

> **注意：**
> 在视频直播 SDK 中，并没有权限的概念，也就是说，房间中任何人都可以加入和离开视频直播连麦，也可以抱(踢)其他用户下麦，您可以根据自己业务的需求决定用户权限，比如：
>
> - 自由上麦模式，任何用户都可以自由上麦
> - 房主、管理员可以抱(踢)其他用户下麦

主动上麦

上麦时序图，时序图参考右边：



主动上麦

房间内的观众都可以调用 enterSeat 接口加入视频直播连麦

```
/// 加入视频直播连麦
RCLiveEngine.getInstance().enterSeat(index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

连麦成功

上麦成功，该观众会接收到 onLiveVideoStarted 回调

```
/// 连麦开始
@Override
public void onLiveVideoStarted() {
//TODO code
}
```

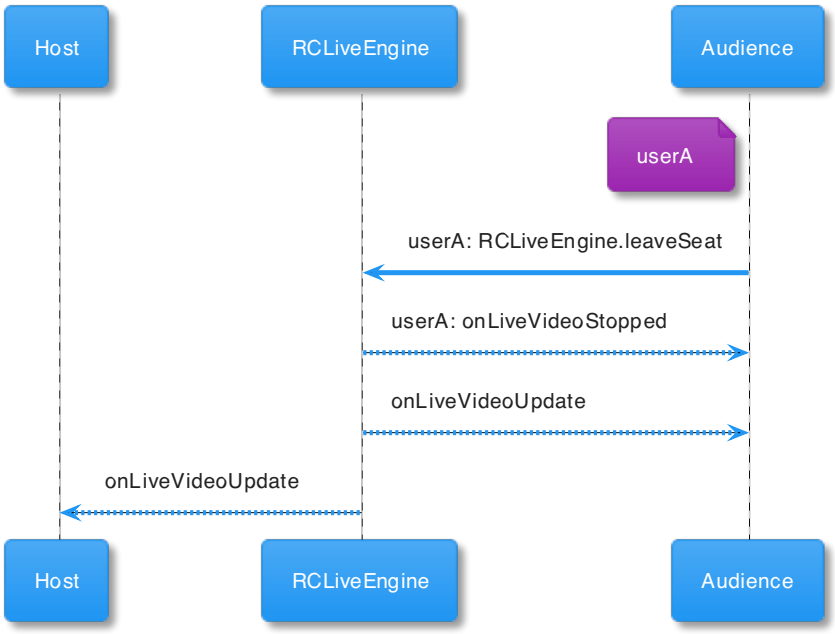
麦位集合发生变化

麦位变化，房间所有用户收到 onLiveVideoUpdate 回调

```
/**
* 连麦用户集合
*
* @param lineMicUserIds 连麦的用户集合
*/
@Override
public void onLiveVideoUpdate(List<String> lineMicUserIds) {
//TODO code
}
```

主动下麦

上麦时序图，时序图参考右边：



主动下麦

正在进行视频直播连麦的用户，随时可以结束视频直播连麦。调用 leaveSeat 接口结束视频直播连麦

```
///下麦
RCLiveEngine.getInstance().leaveSeat(new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

下麦成功

下麦成功，该观众会接收到 onLiveVideoStopped 回调

```
///连麦结束
@Override
public void onLiveVideoStopped() {
//TODO code
}
```

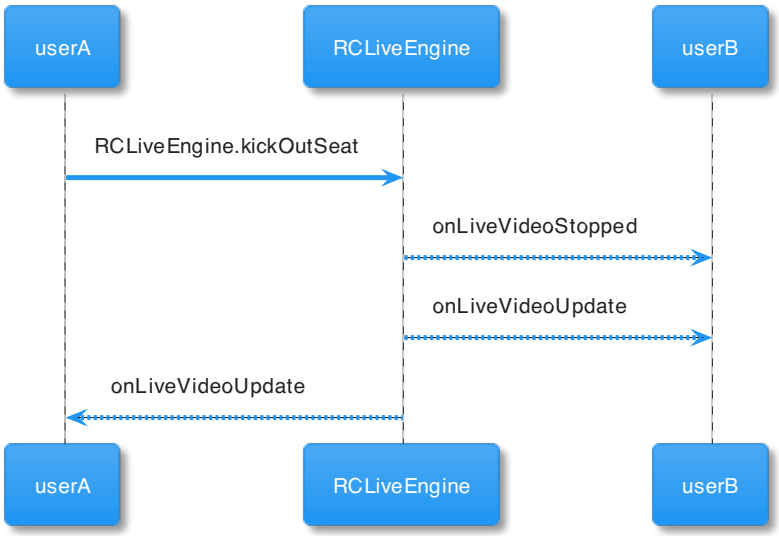
麦位集合发生变化

麦位变化，房间所有用户收到 onLiveVideoUpdate 回调

```
/**
* 连麦用户集合
*
* @param lineMicUserIds 连麦的用户集合
*/
@Override
public void onLiveVideoUpdate(List<String> lineMicUserIds) {
//TODO code
}
```

踢(抱)人下麦

上麦时序图，时序图参考右边：



踢人下麦

在房间内的用户，调用 kickOutSeat 接口将正在进行视频直播连麦的用户抱下麦：

```
///踢人下麦
RCLiveEngine.getInstance().kickOutSeat(user.getUserId(), new RCLiveCallback() {
@Override
public void onSuccess() {
//TOOD code
}

@Override
public void onError(int code, RCLiveError error) {
//TOOD code
}
});
```

被踢下麦

被踢下麦成功，该观众会接收到 onLiveVideoStopped 回调

```
/// 连麦结束
@Override
public void onLiveVideoStopped() {
//TODO code
}
```

麦位集合发生变化

麦位变化，房间所有用户收到 onLiveVideoUpdate 回调

```
/**
* 连麦用户集合
*
* @param lineMicUserIds 连麦的用户集合
*/
@Override
public void onLiveVideoUpdate(List<String> lineMicUserIds) {
//TODO code
}
```

申请连麦

更新时间:2024-08-30

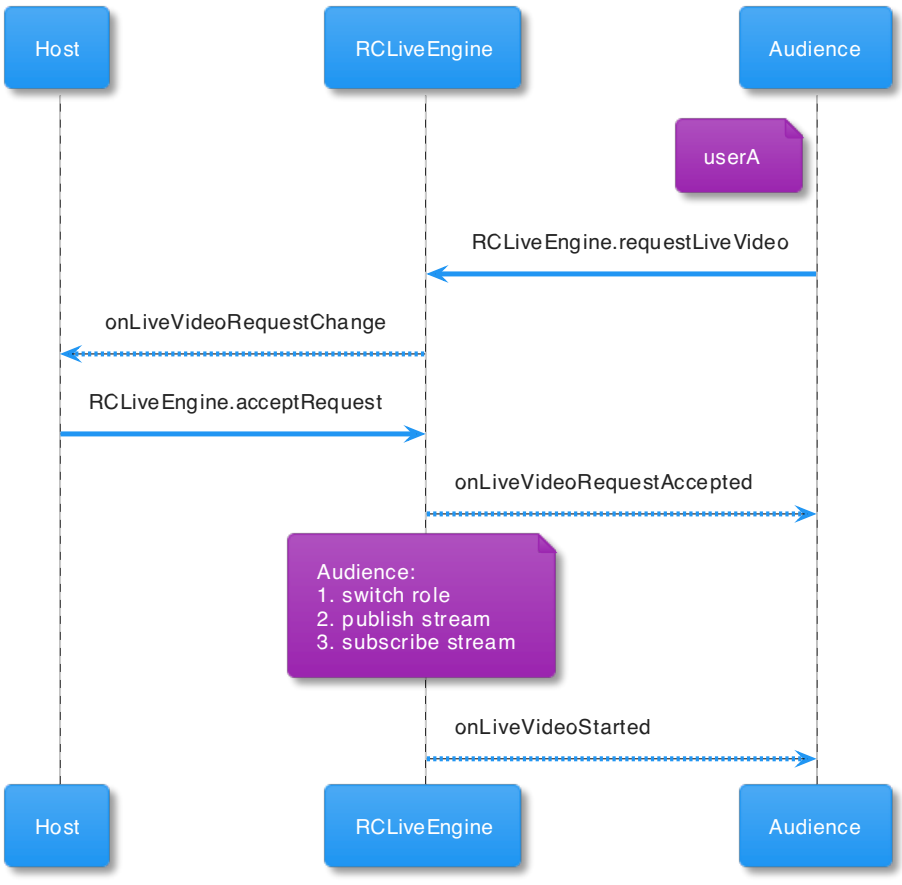
本节介绍如何使用 RCLiveVideoLib 申请视频直播连麦。

注意：

- 申请上麦最多支持 10 人等待，当申请人数达到10人后，再次调用申请上麦接口会报错 `RCLiveVideoLinkMicReqeustFull`；
- 申请上麦请求被同意后，如果上麦的麦位已被占用，SDK 会自动查询第一个空麦位；
- 在视频直播 SDK 中，并没有权限的概念。也就是说当房间某个用户申请上麦时，任何人都可以接收申请麦位变化的回调。您需要根据自己业务的需求，确定哪些人可以处理申请，比如：房主、管理员等。
- 调用 `getRequestLiveVideoIds` 接口查询上麦申请。

观众申请上麦，房主(管理员)同意

时序图参考右边：



观众申请上麦

观众调用 requestLiveVideo 申请上麦。index 参数为观众期待加入的麦位。麦位总数由连麦类型决定，比如：一对一场景，麦位数量为 1；九宫格，麦位数量为 9。如果麦位传入 -1，SDK 会自动查询第一个空麦位：

```
/**
* 发起申请
* @param index 申请想要上的麦位坐标
*/
RCLiveEngine.getInstance().getLinkManager().requestLiveVideo(index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

房主端收到申请

房主触发回调 onLiveVideoRequestChange ：

```
        /// userID为申请人的ID
        RCLiveEngine.getInstance().getLinkManager().acceptRequest(userIds.get(i), new RCLiveCall
          @Override
          public void onSuccess() {
            //TODO code
          }
          @Override
          public void onError(int code, RCLiveError error) {
            //TODO code
          }
        });

      }
      @Override
      public void onError(int code, RCLiveError error) {
        //获取申请人列表失败
      }
    });
}
```

已复制

复制以上代码

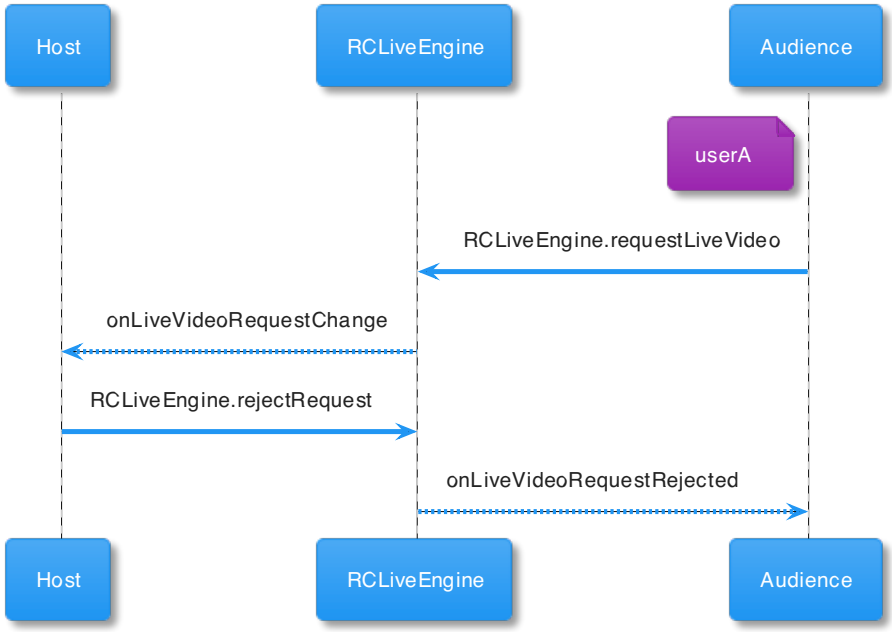
观众端收到同意回调

观众端触发回调 onLiveVideoRequestAccepted ：

```
/**
* 申请上麦被同意：只有申请者收到回调
*/
@Override
public void onLiveVideoRequestAccepted() {
//TODO code
}
```

观众申请上麦，房主(管理员)拒绝

时序图参考右边：



观众申请上麦

观众调用 requestLiveVideo 申请上麦。index 参数为观众期待加入的麦位。麦位总数由连麦类型决定，比如：一对一场景，麦位数量为 1；九宫格，麦位数量为 9。如果麦位传入 -1，SDK 会自动查询第一个空麦位：

```
/**
* 发起申请
* @param index 申请想要上的麦位坐标
*/
RCLiveEngine.getInstance().getLinkManager().requestLiveVideo(index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

房主端收到申请

房主触发回调 onLiveVideoRequestChange ：

```
        /// userID为申请人的ID
        RCLiveEngine.getInstance().getLinkManager().rejectRequest(userIds.get(i), new RCLiveCall
          @Override
          public void onSuccess() {
            //TODO code
          }
          @Override
          public void onError(int code, RCLiveError error) {
            //TODO code
          }
        });

      }
      @Override
      public void onError(int code, RCLiveError error) {
        //获取申请人列表失败
      }
    });
}
```

已复制

复制以上代码

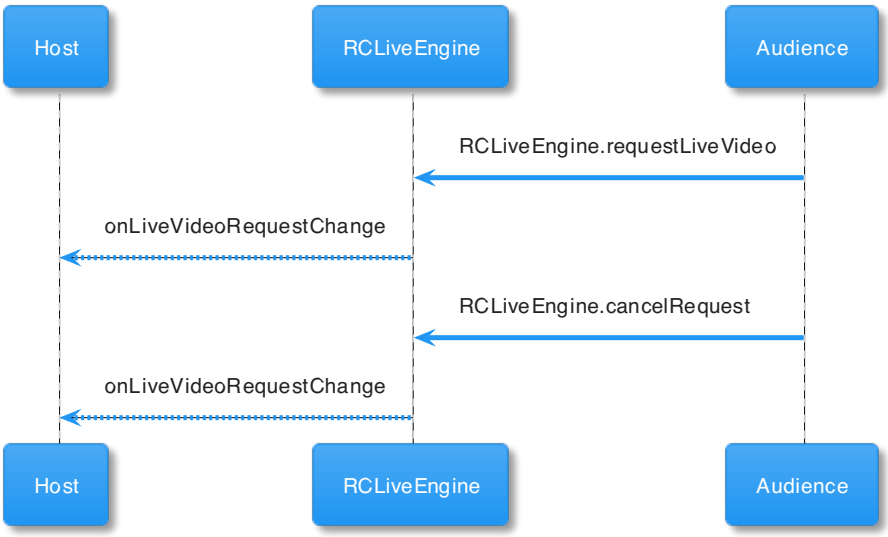
观众端收到拒绝回调

观众端触发回调 onLiveVideoRequestRejected ：

```
/**
* 申请上麦被拒绝：只有申请者收到回调
*/
@Override
public void onLiveVideoRequestRejected() {
//TODO code
}
```

观众申请上麦，然后撤销申请

时序图参考右边：



观众申请上麦

观众调用 requestLiveVideo 申请上麦。index 参数为观众期待加入的麦位。麦位总数由连麦类型决定，比如：一对一场景，麦位数量为 1；九宫格，麦位数量为 9。如果麦位传入 -1，SDK 会自动查询第一个空麦位：

```
/**
* 发起申请
* @param index 申请想要上的麦位坐标
*/
RCLiveEngine.getInstance().getLinkManager().requestLiveVideo(index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

房主端收到申请

房主触发回调 onLiveVideoRequestChange ：

```
/// 连麦申请变化回调
public void onLiveVideoRequestChange() {
//通过“getRequestLiveVideoIds”获取上麦的列表,更新UI
}
```

观众撤销申请

观众调用接口 cancelRequestSeat ：

```
/// 取消申请
RCLiveEngine.getInstance().getLinkManager().cancelRequest(new RCLiveCallback() {
@Override
public void onSuccess() {
//TOOD code

}

@Override
public void onError(int code, RCLiveError error) {
//TOOD code
}
});
```

房主端收到申请

房主触发回调 onLiveVideoRequestChange ：

```
/// 连麦申请变化回调
public void onLiveVideoRequestChange() {
//通过“getRequestLiveVideoIds”获取上麦的列表,更新UI

}
```

邀请连麦

更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 邀请视频直播连麦。

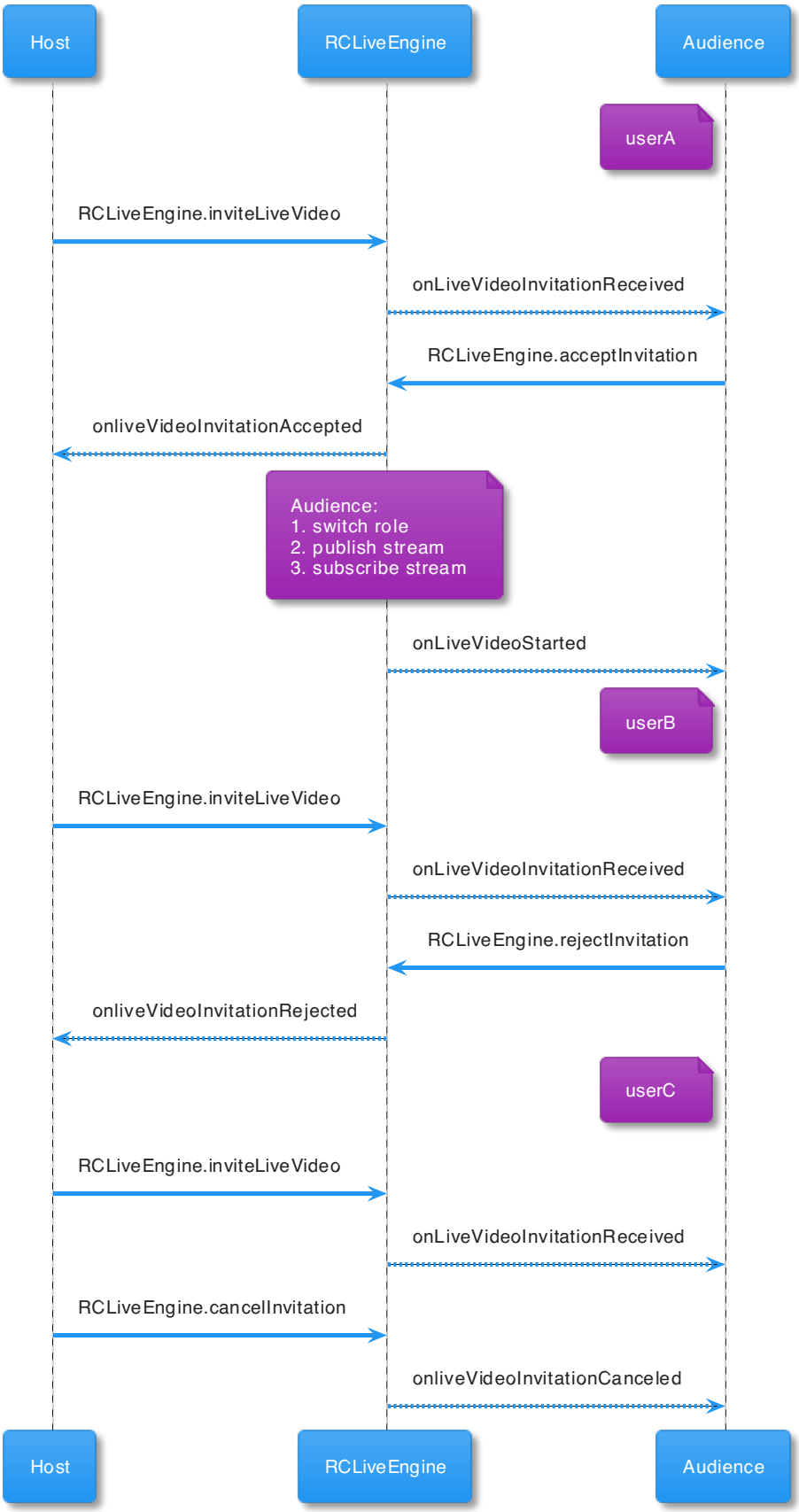
注意：

- 邀请上麦请求被同意后，如果上麦的麦位已被占用，SDK 会自动查询第一个空麦位；
- 在视频直播 SDK 中，并没有权限的概念，也就是说，加入房间的任何人都可以发起邀请上麦，您需要根据自己业务的需求，确定哪些人可以发起邀请，比如：房主、管理员等；

邀请连麦

邀请连麦，时序图参考右边:

- Host 主持人(管理员)
- Audience 观众
- RCLiveEngine SDK



主播（管理员）发起邀请连麦

房主调用 `inviteLiveVideo(String userId, int index, RCLiveCallback callback)` 邀请用户上麦。您可以使用 `index` 参数为受邀者指定麦位。麦位总数由连麦类型决定，比如：一对一场景，麦位数量为 1；九宫格，麦位数量为 9。如果麦位传入 -1，SDK 会自动查询第一个空麦位：

```
///主播（管理员）发起邀请
RCLiveEngine.getInstance().getLinkManager().inviteLiveVideo(userId, index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

被邀请观众收到邀请通知

观众触发回调 onLiveVideoInvitationReceived：

```
/**
* 观众收到连线邀请
*/
@Override
public void onLiveVideoInvitationReceived(String userId, int index) {
//TODO user action
}
```

观众同意上麦,主播收到观众同意邀请

观众处理邀请调用 acceptInvitation

主播端触发回调 onliveVideoInvitationAccepted

```
/// 观众同意邀请
RCLiveEngine.getInstance().getLinkManager().acceptInvitation(userId, index, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
}
/// 主播（管理员）收到同意邀请回调
@Override
public void onliveVideoInvitationAccepted(String userId) {
//TODO code
}
```

观众拒绝上麦,主播收到观众拒绝邀请

观众处理邀请调用 rejectInvitation

主播端触发回调 onliveVideoInvitationRejected

```
/// 拒绝邀请
RCLiveEngine.getInstance().getLinkManager().rejectInvitation(userId, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
}
/// 主播（管理员）收到拒绝邀请回调
@Override
public void onliveVideoInvitationRejected(String userId) {
//TODO code
}
```

主播 (管理员)取消邀请连麦

房主发出邀请上麦后，在观众处理之前，可以调用接口 cancelInvitation 取消上麦邀请：

观众端触发回调 onliveVideoInvitationCanceled

```
/**
* 主播（管理员）取消邀请回调
*/
@Override
public void onliveVideoInvitationCanceled() {
//TODO user action
}
/**
* 观众收到 取消邀请回调
*/
@Override
public void onliveVideoInvitationCanceled() {
//TODO user action
}
```

连麦布局

更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 设置视频直播连麦布局。

> 介绍
>
> - SDK 支持自定义麦位布局view，可自定义比如：用户名称的位置，头像，礼物等等
> - SDK 提供设置内置视频合流布局方法 `setMixType` 和设置自定义连麦布局方法 `setCustomerMixType`
> - SDK 内置了7种主流的视频合流布局模式，也支持自定义模式。布局类型 `RCLiveVideoMixType` 是枚举类型：

```
public enum RCLiveMixType {
// 自定义布局类型,当自定义时选择该类型
RCMixTypeDefault(0),
// 全屏-小窗口类型
RCMixTypeOneToOne(1),
//悬浮窗类型
RCMixTypeOneToSix(2),
//二分屏类型
RCMixTypeGridTwo(3),
//三宫格类型
RCMixTypeGridThree(4),
//四宫格类型
RCMixTypeGridFour(5),
//七宫格类型
RCMixTypeGridSeven(6),
//八宫格类型
RCMixTypeGridNine(7);
}
```

部分内置连麦布局（全部类型，可运行RCRTC APK观看）

- 默认模式：SDK 默认为一对一模式展示，房主全屏显现，连麦用户以浮窗形式显示：

- 房主模式：以房主为核心，房主麦位全屏或大屏显示，其它麦位以小窗口显示，比如浮窗模式：

- 格子模式：所有连麦用户（含房主）麦位大小一致，相互独立显示，比如九宫格模式：

准备

添加视频直播显示视图View 到界面上

调用 `previewView` 获取当前视频渲染 View ，添加到业务视图中展示:

```
//获取预览videoView
RCLiveView previewView = RCLiveEngine.getInstance().preview();
//获取到容器
flLiveView = (FrameLayout) getView().findViewById(R.id.fl_live_view);
flLiveView.removeAllViews();
//将videoView绑定到容器里面
previewView.attachParent(flLiveView, null);
```

设置麦位布局提供者SeatViewProvider

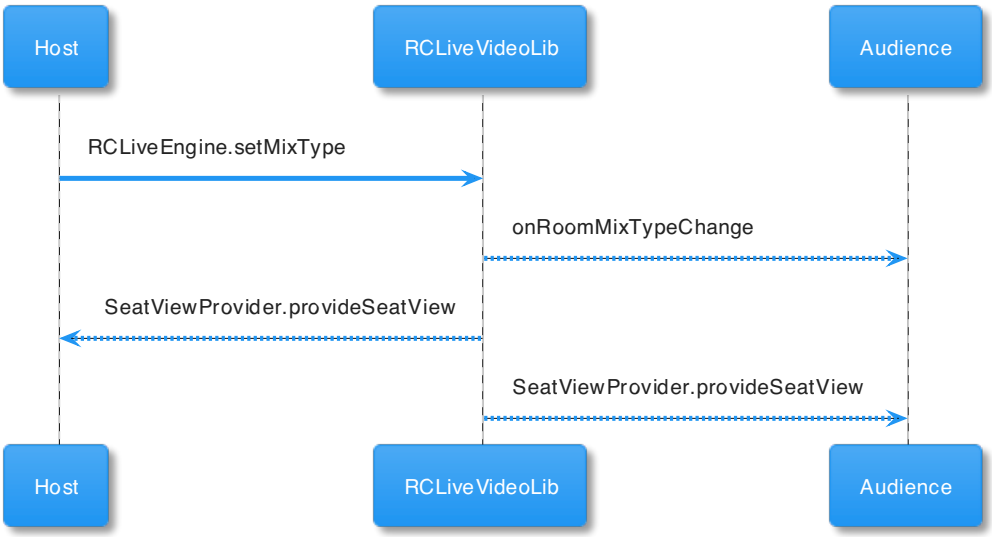
- 调用 `setSeatViewProvider(SeatViewProvider viewProvider)` 接口设置麦位布局提供者，SDK提供了SeatViewProvider的实现类 RCLiveSeatViewProvider，该类实现了基本的麦位view复用逻辑，如不满足需求，可直接设置SeatViewProvider，自定义复用逻辑
- 当视频布局类型变化，麦位信息变化的时候，房间内的观众和房主都触发回调 `SeatViewProvider.provideSeatView`，根据回调返回来的麦位信息自定义麦位view和设置点击事件等操作:

```
///当麦位信息类型发生变化的时候，回调会被触发

///假如 setSeatViewProvider的参数为 RCLiveSeatViewProvider
RCLiveEngine.getInstance().setSeatViewProvider(new RCLiveSeatViewProvider() {

@Override
public void convert(RCHolder holder, RCLiveSeatInfo seat, RCParamter parameter) {
/// 将麦位数据绑定到holder上
}

@Override
public View inflate(RCLiveSeatInfo seatInfo, RCParamter rcParamter) {
///根据麦位数据构建麦位view
return null;
}

@Override
public void onListenerHolds(SparseArray<RCHolder> rcHolderSparseArray) {
///得到当前所有的麦位holder，用户可拿到想要的view，自主刷新view:比如礼物,音频动画等
}
});

///假如 setSeatViewProvider的参数为 SeatViewProvider，建议做缓存策略，避免频繁创建view导致内存异常
RCLiveEngine.getInstance().setSeatViewProvider(new SeatViewProvider() {
@Override
public View provideSeatView(RCLiveSeatInfo seatInfo, RCParamter rcParamter) {
///麦位信息变化，将根据麦位信息构建好的view返回去
return null;
}
});
```

修改默认布局类型

> 修改布局类型，时序图参考右边：
>
> - Host 主持人(管理员)
> - Audience 观众

- RCLiveVideoLib SDK



修改布局类型 （默认布局类型）

- 调用 `setMixType` 设置默认的布局类型，注意，该接口仅支持设置默认类型

```
///主播（管理员）修改布局类型
RCLiveEngine.getInstance().setMixType(rcLiveMixType, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
Log.e(TAG, "onError: " + error.getMessage());
}
});
```

房间其他用户触发回调 onRoomMixTypeChange

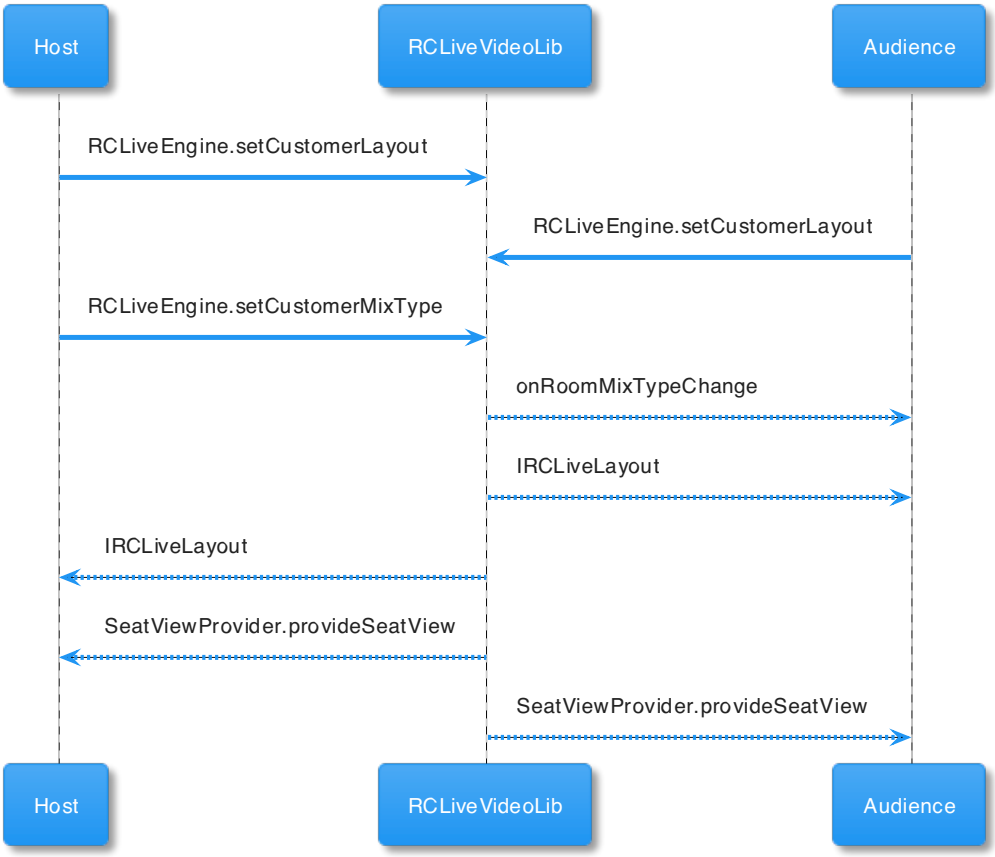
麦位类型变化，远端用户会收到 onRoomMixTypeChange 回调：

```
///远端用户触发回调
@Override
public void onRoomMixTypeChange(RCLiveMixType mixType, int customerType) {
//TODO code
}
```

设置自定义布局

> 修改布局类型，时序图参考右边：

- Host 主持人(管理员)
- Audience 观众
- RCLiveVideoLib SDK



设置自定义布局的画布大小，麦位位置

调用 `setCustomerLayout` 接口，设置一个`IRCLiveLayout`对象，重写该对象的下面几个方法。

```
//设置自定义连麦布局的画布大小等信息
RCLiveEngine.getInstance().setCustomerLayout(new IRCLiveLayout() {
@Override
public int createrIndex(int mixType) {
//房主视图所在的位置
return 0;
}

@Override
public RCLiveCanvas configCanvas(int mixType) {
//画布尺寸
return new RCLiveCanvas(720, 1280);
}

@Override
public RCRect[] seatFrameRects(int mixType) {
//麦位的位置
return new RCRect[]{
new RCRect(0, 0, 1, 1),
new RCRect(1.0f * (width - size - right) / width,
1.0f * (height - size - bottom) / height,
1.0f * size / width,
1.0f * size / height)
};
}

@Override
public void onBindVideoView(int mixType, String roomOwnerUserId, List<RCLiveSeatInfo> seatInfos,
List<RCLiveVideoWrapperView> videoList, RCRTCVideoOutputStream localStram, Map<String,
RCRTCVideoInputStream> remoteVideoStream) {
//用于针对当前的麦位信息等做自定义的UI操作
}
});
```

修改自定义布局类型

- 调用 `setCustomerMixType` 设置自定义布局类型

```
/// 设置自定义模式
RCLiveEngine.getInstance().setCustomerMixType(8, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

房间其他用户触发回调 onRoomMixTypeChange

连麦类型设置成功后，远端用户会收到 onRoomMixTypeChange 回调：

```
///远端用户触发回调
@Override
public void onRoomMixTypeChange(RCLiveMixType mixType, int customerType) {
//TODO code
}
```

麦位管理

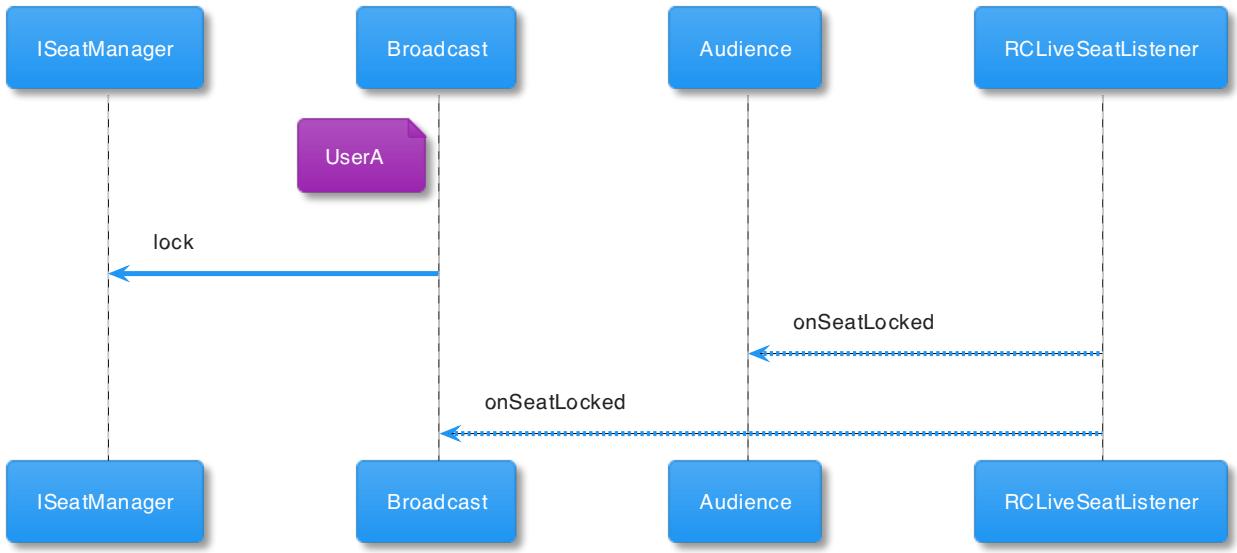
更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 管理和同步麦位信息。

锁麦

锁麦，时序图参考右边:

- Broadcast 主持人(管理员)
- Audience 观众
- RCLiveEngine SDK



麦位锁定

调用 lock 接口更新麦位锁定状态

```
/**
* 锁定麦位
*
* @param index 麦位索引
* @param lock 是否锁定
* @param callback 回调
*/
RCLiveEngine.getInstance().getSeatManager().lock(index, isClose, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

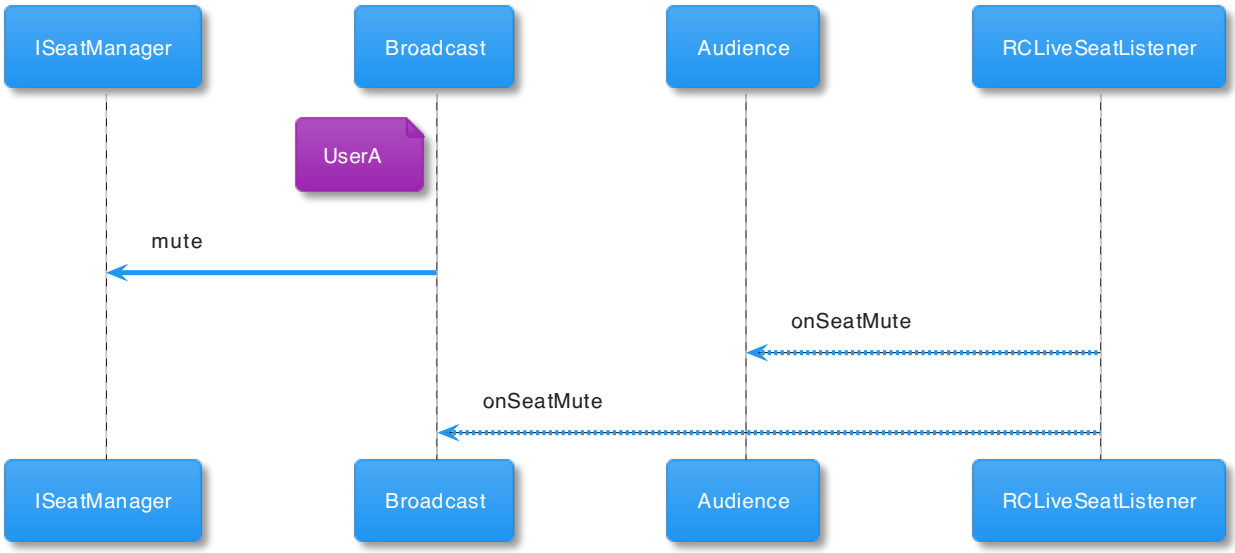
房间内的用户触发 onSeatLocked

房间内用户收到 onSeatLocked 回调，如果麦位锁定，观众不能在该麦位上麦：

```
/**
* 麦位锁定回调,房间内用户收到回调
*
* @param seatInfo 麦位信息
* @param locked 是否锁定
*/
public void onSeatLocked(RCLiveSeatInfo seatInfo, boolean locked) {
//TODO code
}
```

静麦

> **静麦，时序图参考右边:**
>
> - Broadcast 主持人(管理员)
> - Audience 观众
> - RCLiveEngine SDK



麦位静音

调用 mute 接口更新麦位静音状态

```
/**
* 静音
*
* @param index 麦位索引
* @param mute 是否静音
* @param callback 回调
*/
RCLiveEngine.getInstance().getSeatManager().mute(index, isMute, new RCLiveCallback() {
@Override
public void onSuccess() {
//TOOD code
}

@Override
public void onError(int code, RCLiveError error) {
//TOOD code
}
});
```

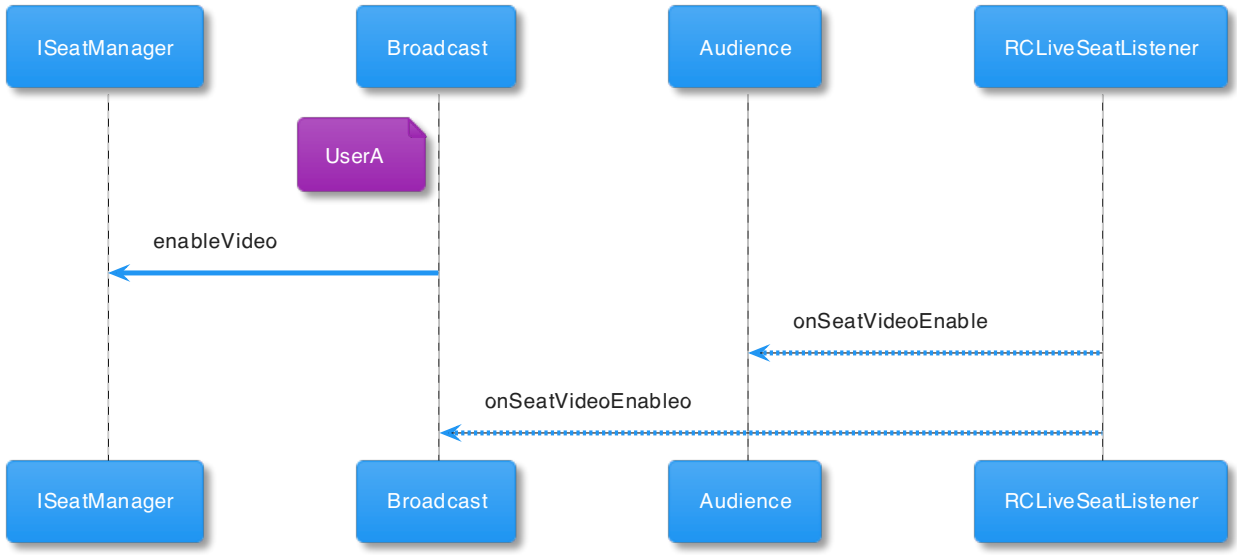
房间内的用户触发 onSeatMute

房间内用户收到 onSeatMute 回调，麦位静音状态变化不影响观众上下麦

```
/**
* 麦位静音回调
*
* @param seatInfo 麦位
* @param mute 是否静音
*/
/// 房间内用户收到回调
public void onSeatMute(RCLiveSeatInfo seatInfo, boolean mute) {
//TODO code
}
```

切换麦位用户视频状态

> 切换麦位视频状态，时序图参考右边:
>
> - Broadcast 主持人(管理员)
> - Audience 观众
> - RCLiveEngine SDK



切换麦位视频状态

调用 enableVideo 接口更新麦位用户是否开启视频

```
/**
* 切换视频
*
* @param index 麦位索引
* @param enable 是否开启视频
* @param callback 回调
*/
RCLiveEngine.getInstance().getSeatManager().enableVideo(index, isVideo, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

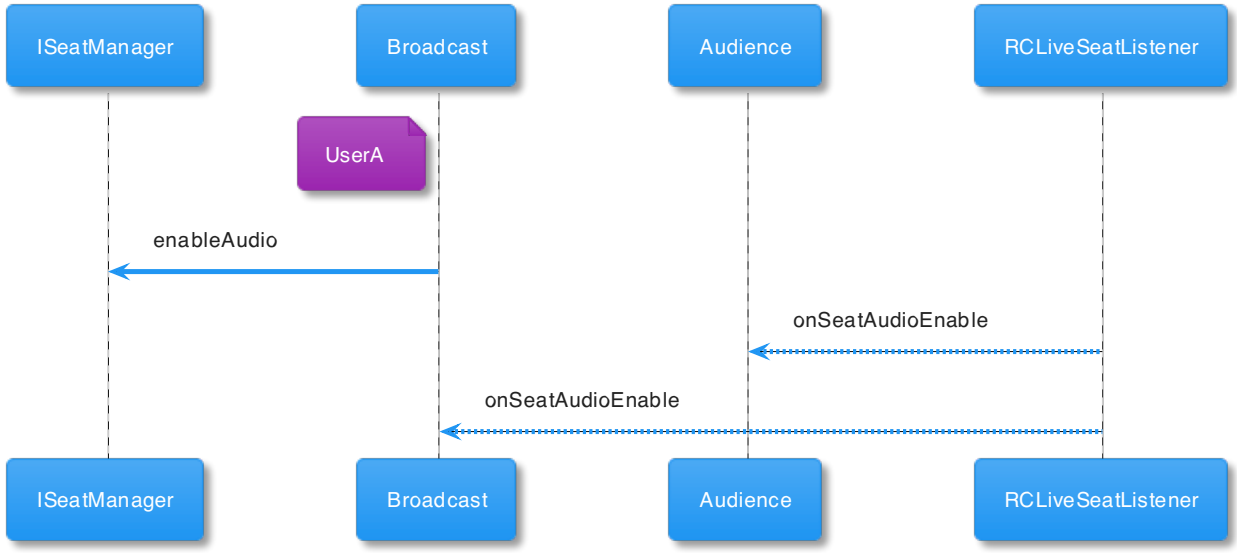
房间内的用户触发 onSeatVideoEnable

房间内用户收到 onSeatVideoEnable 回调

```
/**
* 视频切换状态回调
*
* @param seatInfo 麦位
* @param enable 是否启用
*/
public void onSeatVideoEnable(RCLiveSeatInfo seatInfo, boolean enable) {
//TODO code
}
```

切换麦位用户音频状态

> **切换麦位音频状态，时序图参考右边:**
>
> - Broadcast 主持人(管理员)
> - Audience 观众
> - RCLiveEngine SDK



切换麦位音频状态

调用 `enableAudio` 接口更新麦位用户是否开启音频

```
/**
* 切换麦位音频状态
*
* @param index 麦位索引
* @param enable 是否开启音频
* @param callback 回调
*/
RCLiveEngine.getInstance().getSeatManager().enableAudio(index, !isMute, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {
//TODO code
}
});
```

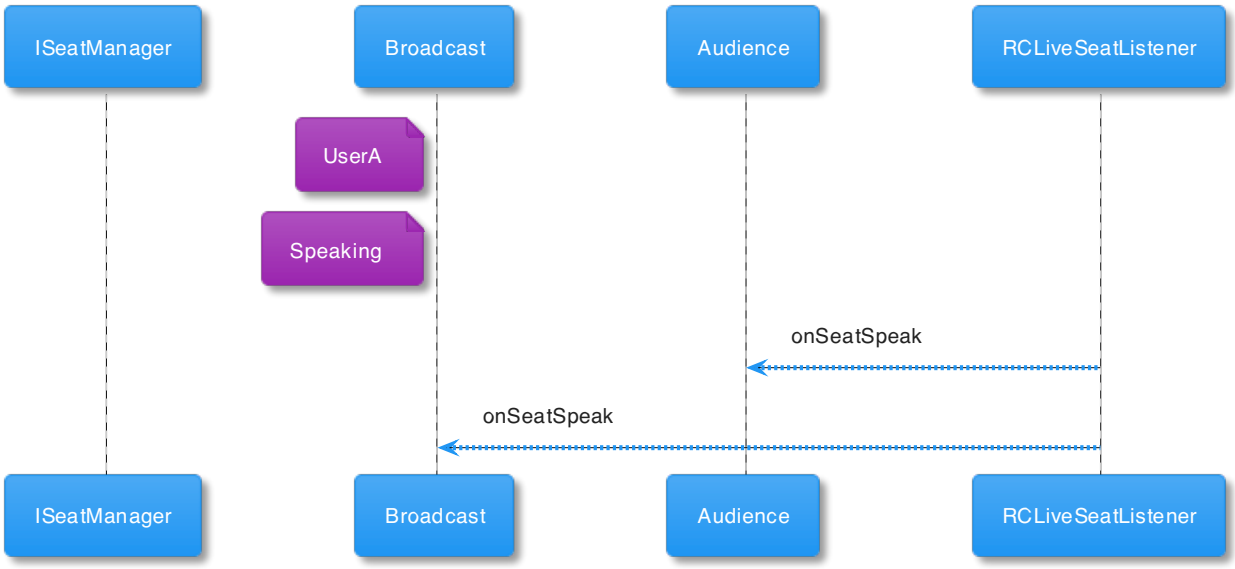
房间内的用户触发 onSeatAudioEnable

房间内用户收到 onSeatAudioEnable 回调：

```
/**
* 音频状态切换回调
*
* @param seatInfo 麦位
* @param enable 是否启用
*/
public void onSeatAudioEnable(RCLiveSeatInfo seatInfo, boolean enable) {
//TODO code
}
```

麦位声音

> **麦位音量变化，时序图参考右边:**
>
> - Broadcast **主持人(管理员)**
> - Audience **观众**
> - RCLiveEngine SDK



麦位音频回调

麦上用户讲话，房间内用户触发 onSeatSpeak 回调：

```
/**
* 说话音量回调
*
* @param audioLevel 音量
*/
void onSeatSpeak(RCLiveSeatInfo seatInfo, int audioLevel);
```

大小流订阅

多人音视频通话过程中，为了减少下行带宽占用，可以开启大小流模式，每个连麦用户会上传一大一小两个视频流，接收方可以根据显示需要来选择接收大流或是小流。小流分辨率保持在 176X144 上下，帧率为 15 FPS。您可以设置 enableTiny 设置麦位订阅流的是否是小流，该属性默认为 true，除非特殊场景，不建议您改变该属性：

设置大小流订阅方式

调用 enableTiny 来控制指定麦位订阅的流的大小

```
/**
* 是否开启小流，默认：开启小流
*
* @param index 麦位索引
* @param enable 是否开启
* @param callback 回调
*/
RCLiveEngine.getInstance().getSeatManager().enableTiny(index, enable, new RCLiveCallback() {
@Override
public void onSuccess() {

}

@Override
public void onError(int code, RCLiveError error) {

}
});
```

跨房间 PK

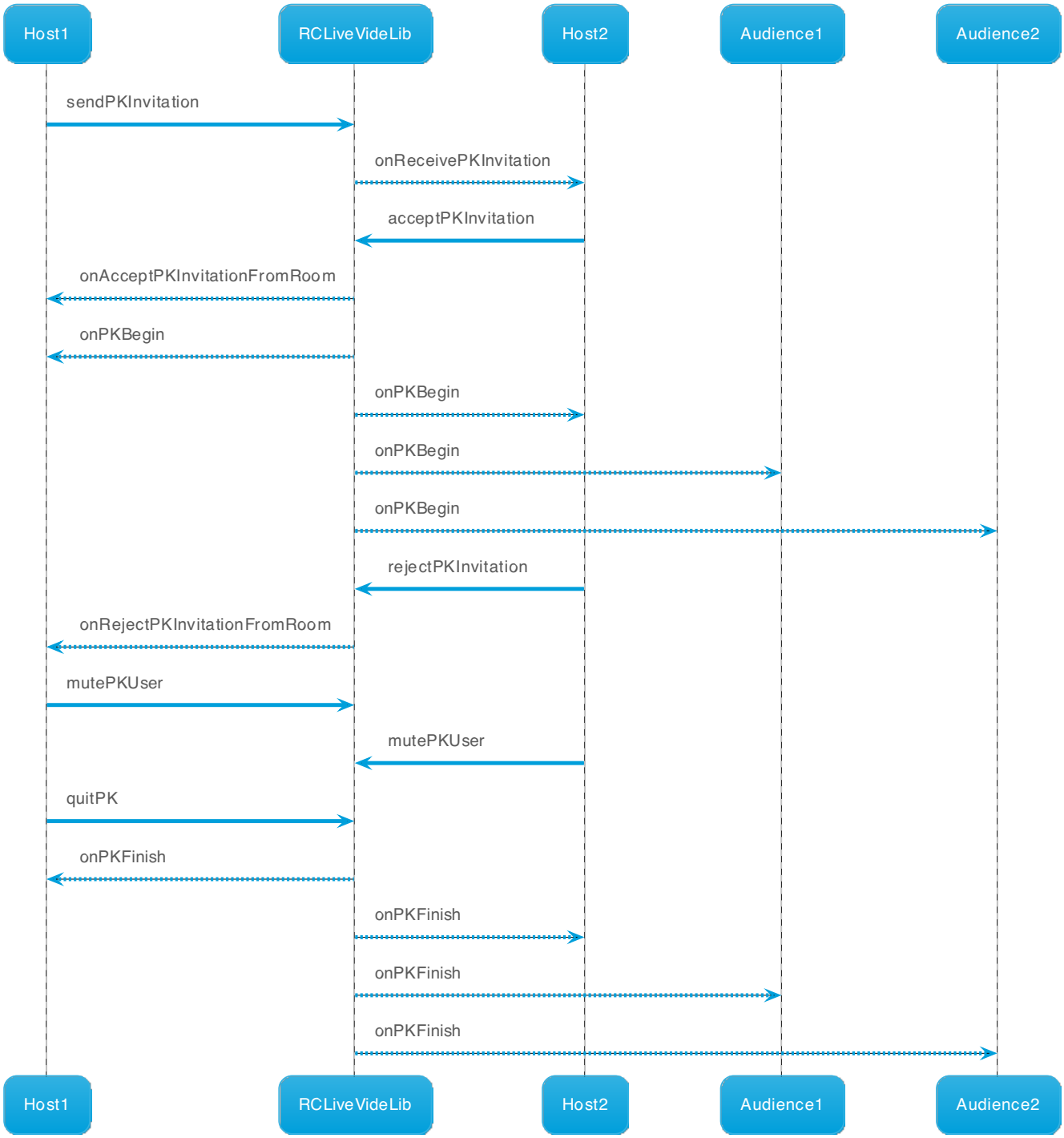
更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 进行跨房间连麦 PK。

> **注意：**
>
> - SDK 目前仅支持 1V1 的PK方案

1v1 PK流程

> PK 时序图参考右边:
>
> - Host1 主播1
> - Audience1 主播1的房间内的观众
> - Host2 主播2
> - Audience2 观众 主播2的房间内的观众
> - RCLiveEngine SDK



主播1 发起 PK 邀请

主播1调用 `sendPKInvitation` 接口邀请其他主播连麦 PK

```
/**
* 发送PK邀请
*
* @param inviteeRoomId 被邀请用户所在的房间id
* @param inviteeUserId 被邀请人的用户id
* @param callback 结果回调
*/
RCLiveEngine.getInstance().sendPKInvitation(inviteeRoomId, inviteeId, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {

}
});
```

主播2 收到 PK 邀请

被邀请的主播2会收到 onReceivePKInvitation 回调；

```
/**
* 收到邀请PK邀请回调
*
* @param inviterRoomId 邀请者的房间id
* @param inviterUserId 邀请者的用户id
*/
@Override
public void onReceivePKInvitation(String inviterRoomId, String inviterUserId) {
/// 一般可以这样操作
/// 1、弹出提示框，是否接受 PK 邀请
/// 2、如果同意，调用 `acceptPKInvitation` 接口
/// 3、如果拒绝，调用 `rejectPKInvitation` 接口
/// 如果有其他原因无法：超时、忙碌等，调用 `rejectPKInvitation` 接口
}
```

主播2 同意 PK 邀请

被邀请的主播2调用 acceptPKInvitation 接口同意 PK 邀请，发起邀请的主播1接收到 onAcceptPKInvitationFromRoom ，两个房间的所有用户收到onPKBegin 回调

```
/**
* 同意 PK 邀请
* @param inviterRoomId 邀请人所在的房间id
* @param inviterUserId 邀请人id
* @param callback 结果回调
*/
RCLiveEngine.getInstance().acceptPKInvitation(inviterRoomId, inviterUserId, new RCLiveCallback() {
@Override
public void onSuccess() {

}

@Override
public void onError(int code, RCLiveError error) {

}
});
/**
* PK邀请被同意回调
*
* @param inviteeRoomId 被邀请人的房间id
* @param inviteeUserId 被邀请人的用户id
*/
@Override
public void onAcceptPKInvitationFromRoom(String inviteeRoomId, String inviteeUserId) {
//TODO code
}

///PK开始
@Override
public void onPKBegin(RCLiveVideoPK rcLiveVideoPK) {
//TODO code
}
```

主播2 拒绝 PK 邀请

被邀请的主播2调用 `rejectPKInvitation` 接口同意 PK 邀请，发起邀请的主播1接收到 `onRejectPKInvitationFromRoom`

```
/**
* 拒绝 PK 邀请
* @param inviterRoomId 邀请人所在的房间id
* @param inviterUserId 邀请人所在的房间id
* @param reason 拒绝原因
* @param callback 结果回调
*/
RCLiveEngine.getInstance().rejectPKInvitation(inviterRoomId, inviterUserId, reason ,new RCLiveCallback()
{
@Override
public void onSuccess() {

}

@Override
public void onError(int code, RCLiveError error) {

}
});
/**
* PK 邀请被拒绝回调
*
* @param inviteeRoomId 被邀请人的房间id
* @param inviteeUserId 被邀请人的用户id
* @param reason 拒绝原因
*/
@Override
public void onRejectPKInvitationFromRoom(String inviteeRoomId, String inviteeUserId,, String reason) {
//TODO code
}
```

主播1 撤销 PK 邀请

发起邀请的主播1调用 cancelPKInvitation 接口取消已经发送的 PK 邀请，被邀请的主播2会收到 onPKInvitationCanceled 回调；

```
/**
* 取消pk邀请
*
* @param inviteeRoomId
* @param inviteeId
* @param resultBack
*/
RCLiveEngine.getInstance().cancelPKInvitation(inviteeRoomId, inviteeId, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code 撤销成功
}

@Override
public void onError(int code, RCLiveError error) {

}
});

/**
* pk邀请被邀请者取消回调
*
* @param inviterRoomId 邀请者的房间id
* @param inviterUserId 邀请者的用户id
*/
@Override
public void onPKInvitationCanceled(String inviterRoomId, String inviterUserId) {
//TODO code
}
```

任何一方主播主动结束 PK

任何一方主播都可以调用 quitPK 接口主动结束 PK，对方主播接收到 onPKFinish 回调

```
/**
* 退出PK
*
* @param callback 结果回调
*/
RCLiveEngine.getInstance().quitPK(new RCLiveCallback() {
@Override
public void onSuccess() {
}

@Override
public void onError(int code, RCLiveError error) {
}
});

///PK结束
@Override
public void onPKFinish() {
//TODO code
}
```

恢复 PK

在 PK 期间，任何一方主播如果异常退出等原因导致PK中断，调用 resumePk 接口来恢复 PK ，然后自己会收到 onPKBegin

的回调

```
/**
* 恢复PK
* 注意：
* 1、此方法跳过邀请，直接进入 PK 阶段
* 2、执行该方法以后，只能执行 quitPk 退出 PK
* 3、该方法要求当前必须已经在 PK 状态中
*/
RCLiveEngine.getInstance().resumePk(new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {

}
});
///PK 开始
@Override
public void onPKBegin(RCLiveVideoPK rcLiveVideoPK) {

}
```

声音控制

在 PK 期间，主播可以调用 mutePKUser 接口来关闭和打开对方的声音：

```
/// 控制 PK 声音
RCLiveEngine.getInstance().mutePKUser(isMute, new RCLiveCallback() {
@Override
public void onSuccess() {
//TODO code
}

@Override
public void onError(int code, RCLiveError error) {

}
});
```

房间属性

更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 自定义视频直播房间属性。

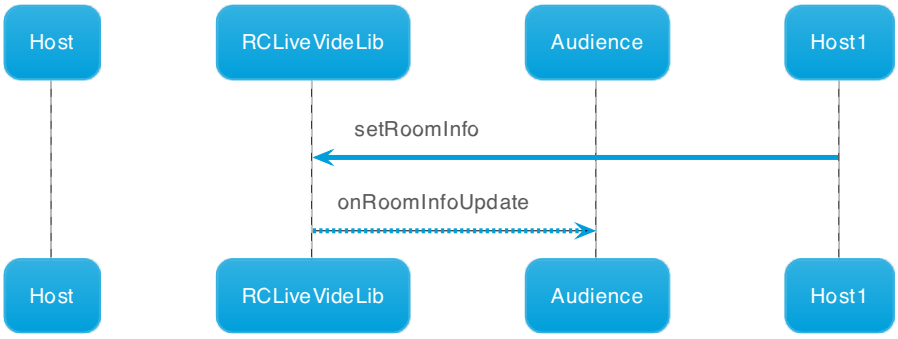
注意：

- 视频直播间支持自定义房间属性：比如公告，房间名等需要及时同步的信息
- 房间自定义属性数量不能大于 50 个，多出属性可能更新失败；
- 设置房间信息接口一次性最多设置 10 个属性。

视频直播房间属性接口

更新房间属性 时序图参考右边:

- Host1 主播
- Audience 观众
- RCLiveVideLib SDK



设置房间属性

视频直播 SDK 提供了自定义属性接口 setRoomInfo，针对不同的直播场景（教育、游戏、社交等）开发者可以自定义属性。

```
/**
* 更新房间自定义属性,建议属性数量<50
*
* @param kvInfo 房间自定义属性<key, value>
* @param callback 结果回调
*/
Map<String, String> kv = new HashMap<>();
kv.put(key, vaule);
RCLiveEngine.getInstance().setRoomInfo(kv, new RCLiveCallback() {
@Override
public void onSuccess() {
//TOOD code
}

@Override
public void onError(int code, RCLiveError error) {
//TOOD code
}
});
```

其他用户触发 onRoomInfoUpdate 回调

其他用户触发 onRoomInfoUpdate 回调。步骤如下：

```
//收到房间回调
@Override
public void onRoomInfoUpdate(String key, String value) {
switch (key) {

}
}
```

用户管理

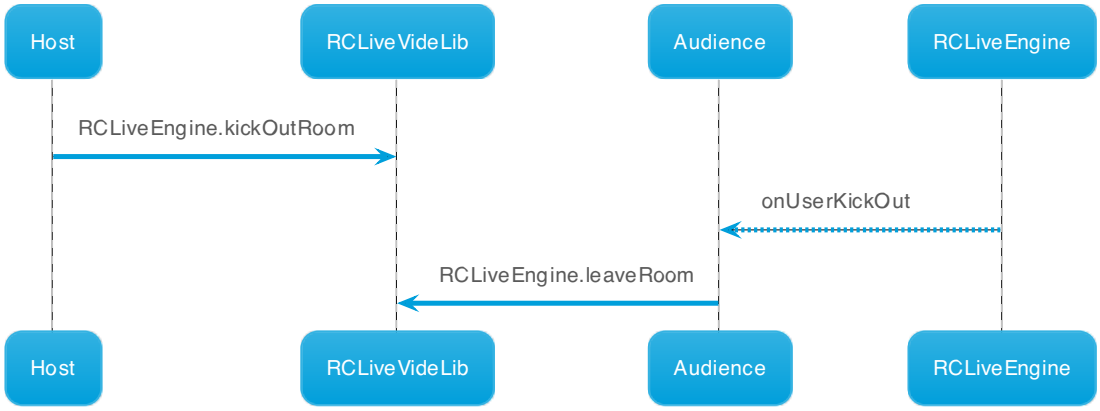
更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 管理用户。

> 注意：在视频直播 SDK 中，并没有权限的概念，也就是说，房间的任何人都可以调用踢出房间的接口，您需要根据自己业务的需求，确定哪些人可以调用，比如：房主、管理员等。

视频直播用户管理接口

> 将用户踢出房间 时序图参考右边:
>
> - Host1 主播
> - Audience 观众
> - RCLiveVideLib SDK



踢出房间

调用 kickOutRoom 接口

```
/**
* 将用户踢出房间
*
* @param userId 被踢出用户的id
* @param callback 结果回调
*/
RCLiveEngine.getInstance().kickOutRoom(userId, new RCLiveCallback() {
@Override
public void onSuccess() {

}

@Override
public void onError(int code, RCLiveError error) {

}
});
```

被踢用户触发 onUserKickOut 回调

观众接收到 onUserKickOut 回调

```
  //被踢出房间，调用离开房间接口和反注册
  RCLiveEngine.getInstance().leaveRoom(new RCLiveCallback() {
    @Override
    public void onSuccess() {
      //销毁资源
    }
    @Override
    public void onError(int code, RCLiveError error) {
    }
  });

}
```

已复制

复制以上代码

发送消息

更新时间:2024-08-30

本节介绍如何使用 RCLiveVideoLib 发送直播间消息。

注意：

- 发送直播间消息接口在 IMLib 中。集成了 IMLib 或者 IMKit 后可调用该 API。
- 视频直播房 SDK 依赖 IMLib（或 IMKit），因此您可以使用即时通讯客户端 SDK 的全部能力。详见 即时通讯客户端 SDK 文档。

视频直播房间发送消息属性接口

发送直播间消息

视频直播 SDK 提供了属性接口 sendMessage，可直接发送当前所在直播间的消息，也可以使用 IMLib提供的发送消息接口。

```
/**
* 发送直播房间消息
*
* @param content 消息内容
* @param callback 结果回调
*/
RCLiveEngine.getInstance().sendMessage(textMessage, new RCLiveCallback() {
@Override
public void onSuccess() {

}

@Override
public void onError(int code, RCLiveError error) {

}
});
//也可以使用IMLib提供的发送消息方法,详细用法可参照 IM 开发文档
RongCoreClient.getInstance().sendMessage(Conversation.ConversationType.CHATROOM,
roomId,
messageContent,
pushContent,
pushData,
new IRongCoreCallback.ISendMessageCallback() {
@Override
public void onAttached(Message message) {

}

@Override
public void onSuccess(Message message) {


}

@Override
public void onError(Message message, IRongCoreEnum.CoreErrorCode coreErrorCode) {

}
});
```

更新日志 2.1.2

更新时间:2024-08-30

新增功能:

1. 新增主播开播添加三方 CDN 地址
2. 新增观众订阅三方 CDN 方法

2.1.0.8

修复问题:

1.解决镜像问题
2.去掉切换主播角色重复推流的问题

2.1.0.7

新增功能:

1.增加音视频首帧回调接口 `onReportFirstFrame`

2.1.0.6

新增功能:

1.取消默认的视频流参数

2.1.0.5

修复问题:

1. 修复多端抢同一个麦位状态错误问题

2.1.0.4

修复问题:

1. 新增了合流布局参数接口 `onInitMixConfig` ，用户可自定义修改合流参数
2. 合流布局裁剪根据麦位上是否有视频流来做判断

2.1.0.3

修复问题：

1. 修改切换麦位以后，原麦位 view 镜像异常的问题

2.1.0.2

修复问题：

1. 修复获取房间属性 Key 值异常的问题

2.1.0.1

修复问题：

1. 修复声道切换时，加入房间失败的问题

2.1.0

新增功能：

1. 添加 PK 功能相关接口
   - 发起 PK 邀请
   - 取消 PK 邀请
   - 同意 PK 邀请
   - 拒绝 PK 邀请
   - 关闭和开启对方声音
   - 结束 PK
   - 恢复 PK
2. 新增配置第三方CDN地址方法
   - 添加第三方CDN推流地址 `addCDNPublishStream`
   - 移除第三方CDN推流地址 `removeCDNPublishStream`
3. - 新增 `onInitRCRTCConfig` 接口，可自定义RCRTC初始化参数

Bug修复：

1. 解决调用 `enableTiny` 方法切换大小流失败的问题
2. 解决魅族机型开播以后抱用户下麦，用户视图不消失的问题
3. 提供了解决 surfaceView 叠加不显示的方法
4. 添加音频路由，解决部分耳返异常问题

2.0.0

> 发布日期：2022/01/06

1. 支持 7 种内置连麦模式和自定义连麦模式
2. 支持上下麦，切换麦位，锁麦，关闭麦克风等麦位相关操作
3. 支持设置视频分辨率、帧率
4. 支持切换视频大小流
5. 对外暴露本地视频流，可接入第三方美颜等

常见问题　　创建视频直播房间返回失败

更新时间:2024-08-30

这通常是由于您没有开通 App Key 的音视频直播服务，或者是免费时长已用完时。您可以通过控制台查看是否已经开通音视频服务。

申请连麦时，谁有权限通过或拒绝申请？

在视频直播 SDK 中，并没有权限的概念，也就是说，当房间某个用户申请上麦时，任何人都可以接收申请麦位变化的回调。您需要根据自己业务的需求，确定哪些人可以处理申请。

视频连麦期间，点击连麦用户窗口，事件回调怎么处理？

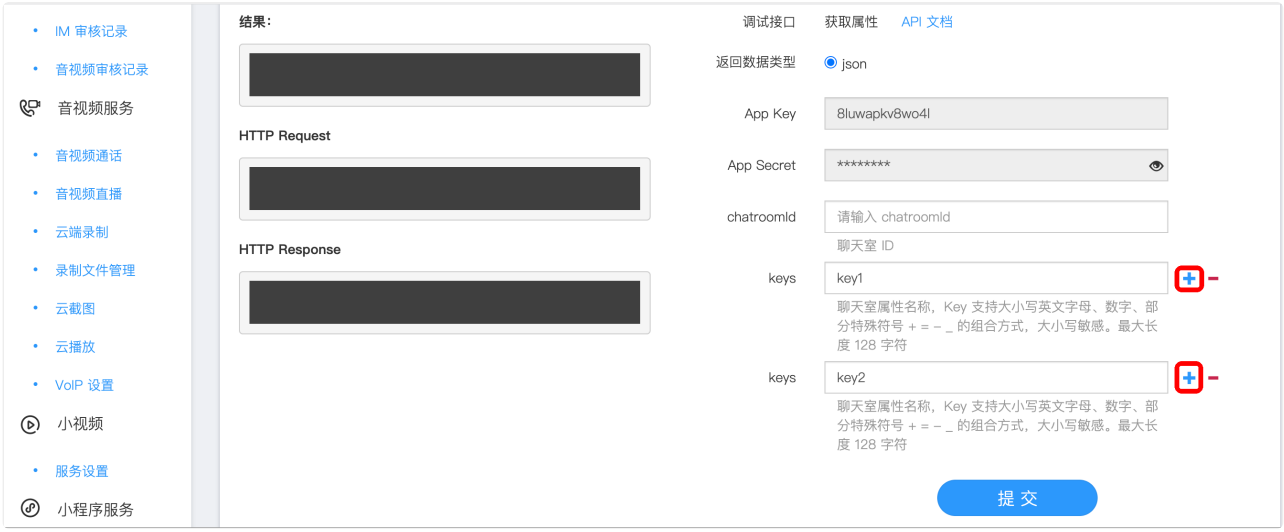
在视频直播 SDK中，通过给RCLiveEngine设置一个SeatViewProvider对象，当麦位信息发生了变化或者有用户上下麦位的时候，房间的所有用户会收到布局更新的回调 `provideSeatView`。用户可以根据回调方法中携带的布局信息来自定义布局 UI，并且给布局设置点击事件。

如何利用控制台查询房间属性（KV）

1. 访问控制台北极星开发者工具箱的 IM Server API 调试 页面。
2. 依次选择聊天室服务 > 获取属性，如下图所示：

   请注意检查页面顶部的应用与生产/开发环境是否正确。

3. 按照页面提示输入 `chatroomId` 和所需要查询的 Key 名称。支持一次查询多个 Key。



除了可以查询聊天室 KV 属性信息，还可以通过该页面查询聊天室信息、聊天室的用户、用户是否在聊天室，另外还可以直接查找、设置、删除聊天室的一些属性。

其他资源

更新时间:2024-08-30

融云为开发者提供了丰富的文档和示例代码资源，包含一个拥有完整功能模块化的开源项目 RCRTC。

Android 相关资源

- **RC RTC 开源项目**

  GitHub · Gitee

- **QuickDemo**

  GitHub · Gitee

状态码

更新时间:2024-08-30

状态码	说明
80000	操作成功
80101	加入房间失败
80102	离开房间失败
80103	没有加入房间
80104	房间不存在或已关闭
80201	发布直播流失败
80202	订阅分流失败
80203	订阅 CDN 流失败
80204	设置 CDN 流参数失败
80301	获取房间信息失败
80302	更新房间信息失败
80303	房间KV更新失败
80304	房间KV获取失败
80401	连麦失败

状态码	说明
80402	下麦失败
80403	连麦状态错误
80411	申请连麦失败
80412	连麦申请已满
80413	正在连麦中，不能发起连麦申请
80414	同意连麦申请失败
80415	拒绝连麦申请失败
80501	麦位序号无效
80502	麦位已满
80503	麦位已上锁
80504	麦位上存在用户
80505	麦位状态异常
80601	设置了相同的连麦类型
80700	当前正在 PK 中
80701	当前不在 PK 中，或者 PK 数据错误
80702	发送 PK 邀请失败
80703	取消 PK 邀请失败
80704	响应 PK 邀请失败

状态码	说明
80705	开始 PK 失败
80706	结束 PK 失败
80707	静音对方主播声音失败
81001	消息发送失败
82001	权限错误
82101	参数检查异常