场景融合

视频直播

iOS 2.X

2024-05-17

视频直播概述

更新时间:2024-05-17

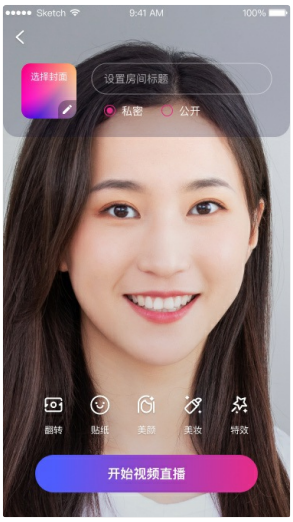
RCLiveVideoLib 是融云针对视频直播场景设计的 SDK，API 设计参考了市面主流视频直播 App 的功能和场景特点。SDK 融合 **RongIMLib** 和 **RongRTCLib** 实现视频直播场景，将复杂逻辑（订阅和发布流、上下麦、连麦布局等）进行了封装，降低开发者的接入难度。相较于开发者自己接入低延迟直播和即时通讯，SDK 具备以下优势：

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	1.0.0
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	1.0.0
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	1.0.0
取消连麦申请	观众取消连麦申请，房主接收回调	1.0.0

功能	描述	支持版本
同意连麦申请	房主同意连麦申请，并与观众开始连麦	1.0.0
拒绝连麦申请	房主拒绝连麦申请，观众接收回调	1.0.0
查询连麦申请	查询当前房间上麦申请信息	1.0.0
邀请上麦	房主邀请观众进行视频连麦	1.0.0
取消上麦邀请	房主取消连麦邀请，观众接收到回调	1.0.0
同意上麦邀请	观众同意连麦邀请，并与房主开始连麦	1.0.0
拒绝上麦邀请	观众拒绝连麦邀请，房主接收到回调	1.0.0
抱用户下麦	将指定用户抱下麦位	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
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	1.0.0

开通融云 CDN 服务

更新时间:2024-05-17

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:2024-05-17

- **Xcode**：确保与苹果官方同步更新
- **CocoaPods**：1.10.0 及以上
- **iOS**：11.0 及以上
- **Objective-C**：2.0

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。

使用 pod 集成视频直播 SDK

> 注意
>
> - 请确保 CocoaPods 版本为 1.10.0 或更高版本，您可以通过运行 `pod --version` 命令进行查看当前版本。
> - 如果您使用 融云 CDN，请在 Podfile 中添加 RongRTCPlayer 库。

1. 在项目的 **Podfile** 中添加

```
pod 'RCLiveVideoLib'
# 融云 CDN 插件（可选）
pod 'RongCloudRTC/RongRTCPlayer'
```

2. 在终端中运行以下命令：

```
pod install
```

3. **Pod** 安装完成后，`CocoaPods` 会在您的工程根目录下生成一个 .xcworkspace 文件。您需要通过此文件打开您的工程，而不是之前的 .xcodeproj

SDK 的包大小

- 视频直播 SDK 支持真机和模拟器；
- 视频直播 SDK 打包增量大概 6 M（包含依赖库 IM 和 RTC）。

版本依赖说明

`RCLiveVideoLib` 依赖融云 `IMLib` 与 `RTCLib`，依赖版本如下。使用 CocoaPods 安装时会自动处理依赖关系。

依赖组件	版本
IMLib	5.1.7 及以上
RTCLib	5.1.16.1 及以上
RongRTCPlayer	必须和 RTCLib 保持一致

初始化

更新时间:2024-05-17

本节介绍如何快速初始化视频直播 SDK。

在使用 RCLiveVideoLib 之前，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 开通音视频直播服务。

RCLiveVideoLib 初始化

RCLiveVideoLib 依赖融云即时通讯 SDK 提供消息通讯能力。因此，需要使用即时通讯 SDK 的初始化方法。

- 如果您的项目未集成任何融云 SDK，请使用 IMLib 的初始化方法。请在 application:didFinishLaunchingWithOptions: 中执行下列方法初始化：

```
/// 使用 **IMLib** 初始化
/// appkey 即您申请的 appkey，需要开通音视频直播服务
/// token一般是您在登录自己的业务服务器之后，业务服务器返回给您的，可存在本地。
[[RCIMClient sharedRCIMClient] initWithAppKey:appkey];
[[RCIMClient sharedRCIMClient] connectWithToken:token timeLimit:5
dbOpened:^(RCDBErrorCode code) {
//消息数据库打开，可以进入到主页面
} success:^(NSString *userId) {
//连接成功
} error:^(RCConnectErrorCode status) {
if (status == RC_CONN_TOKEN_INCORRECT) {
//Token 错误，可检查客户端 SDK 初始化与 App 服务端获取 Token 时所使用的 App Key 是否一致
} else if(status == RC_CONNECT_TIMEOUT) {
//连接超时，弹出提示，可以引导用户等待网络正常的时候再次点击进行连接
} else {
//无法连接 IM 服务器，请根据相应的错误码作出对应处理
}
}];
```

- 如果您的项目已集成融云 IMLib 或 IMKit SDK，您无需做任何更改。

开启和结束视频直播

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 开启和结束视频直播。

开启和结束视频直播接口

- 准备视频直播：调用 `prepare` 接口初始化直播配置，并接收到本地视频流回调 `didOutputFrame:` ；
- 开启视频直播：调用 `begin` 接口开启直播，开始将本地音视频流推到远端；
- 结束视频直播：调用 Server 接口关闭直播后，离开直播间。

> 注意：SDK 没有关闭直播房间的接口，请在业务服务器实现，详情请参考 DemoServer 。

准备视频直播

在准备阶段可以预览视频，配置流信息（分辨率、码率、帧率等），处理视频流（美颜、水印等）。相关接口一般在 viewDidLoad 方法里调用，开发者也可根据业务情况，选择合适的时机。

设置视频信息

开发者需要根据业务场景设置视频信息：分辨率、码率和帧率等，一般直播场景推荐采用分辨率 640x480，其它分辨率请参考融云 RTC 分辨率 。默认情况下，SDK 根据当前分辨率进行匹配，自动适用对应的默认最小和最大码率设置。在通话过程中，实际视频码率在最小码率和最大码率之间根据网络情况浮动。您也可以自定义本端的最小和最大码率。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>

RCRTCVideoStreamConfig *config = [[RCRTCVideoStreamConfig alloc] init];
config.videoSizePreset = RCRTCVideoSizePreset640x480;
config.videoFps = RCRTCVideoFPS15;
config.minBitrate = 200;
config.maxBitrate = 900;
RCRTCEngine.sharedInstance.defaultVideoStream.videoConfig = config;
```

设置代理

RCLiveVideoEngine 根据业务划分有三类事件代理：

- 通用代理：`RCLiveVideoDelegate` 监听房间、麦位、视频流、数据统计等事件；
- 布局代理：`RCLiveVideoMixDelegate` 和 `RCLiveVideoMixDataSource` 监听布局相关事件；
- PK 代理：`RCLiveVideoPKDelegate` 监听 PK 相关事件；

开发者设置代理后，可以在 RCLiveVideoDelegate 事件的 didOutputFrame: 回调里处理视频流，设置美颜、水印等。

```
RCLiveVideoEngine.shared.delegate = self;
RCLiveVideoEngine.shared.pkDelegate = self;
RCLiveVideoEngine.shared.mixDelegate = self;
RCLiveVideoEngine.shared.mixDataSource = self;
```

添加视频预览

将视频直播预览 prevewView 添加到要显示的视图上，并调用准备接口 prepare。

```
UIView *previewView = [RCLiveVideoEngine.shared previewView];
previewView.frame = self.view.bounds;
[self.view addSubview:previewView];

[RCLiveVideoEngine.shared prepare];
```

处理视频流输出回调

在视频流输出 didOutputFrame: 回调中，可以通过视频帧数据 RCRTCVideoFrame 实现流二次处理，例如：美颜 。

```
/// 美颜处理后，再返回处理后的 frame
- (RCRTCVideoFrame *)didOutputFrame:(RCRTCVideoFrame *)frame {
return frame;
}
```

开启视频直播

通常在直播场景有房间列表，该列表由业务服务器维护，客户端调用服务器创建房间接口后，再调用开启直播 `begin:completion:` 接口。

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。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>

/// 调用服务器创建房间接口
[network createRoom:^(NSString *roomId) {
}];
```

开始直播

根据直播推拉流方式分为：MCU（低延迟）、CDN(内置或三方)，MCU 在任何情况下都能推拉流，CDN 需要配置 CDN 服务。
客户端在获取房间 ID 之后，如果采用 MCU 或内置 CDN，直接调用 `begin:completion:` 方法开启直播；如果采用三方 CDN，需调用 `begin:publishURL:completion:` 方法开启直播并设置推流地址。

```
/// 开启直播，需要传入roomId，roomId为业务服务器返回
[RCLiveVideoEngine.shared begin:roomId completion:^(RCLiveVideoErrorCode code) {
if (code == RCLiveVideoSuccess) {
/// TODO success
} else {
/// TODO failure
}
}];
```

结束视频直播

结束直播需要调用业务服务器接口，关闭成功后，客户端离开直播间。

结束直播

开发者需要调用业务服务器接口关闭直播房间，关闭成功后，房间内用户会收到房间关闭的回调。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>
/// 调用服务器关闭房间接口
[network closeRoom:^(BOOL success) {
if (success) {
[RCLiveVideoEngine.shared leave:^(RCLiveVideoErrorCode code) {
/// TODO dismiss/pop View Controller
}];
} else {
/// TODO network error
}
}];

/// 房间已关闭
- (void)roomDidClosed {
/// TODO dismiss/pop View Controller
}
```

加入和离开视频直播

更新时间:2024-05-17

本节介绍观众端如何使用 RCLiveVideoLib 加入和离开视频直播。

加入和离开视频直播的接口

- 加入视频直播：调用 `joinRoom:completion:` 方法加入视频直播房间；
- 离开视频直播：调用 `leaveRoom` 接口离开视频直播房间；

加入视频直播

通常客户端会从服务端拉取房间列表后，根据房间 ID 调用 joinRoom:completion: 接口加入指定的房间。观众端根据业务需求，订阅低延迟或 CDN 视频流。

> 注意：观众加入的房间必须已创建。

设置代理

用户在加入房间前，需要设置事件监听，根据事件回调处理业务逻辑。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>

RCLiveVideoEngine.shared.delegate = self;
RCLiveVideoEngine.shared.pkDelegate = self;
RCLiveVideoEngine.shared.mixDelegate = self;
```

添加视频预览

用户在加入房间前，需要将视频直播画面 previewView 添加到要显示的视图上。

```
UIView *previewView = [RCLiveVideoEngine.shared previewView];
previewView.frame = self.view.bounds;
[self.view addSubview:previewView];
```

加入房间并订阅低延迟视频流

客户端调用 joinRoom:completion: 接口加入房间，默认订阅低延迟音视频流。

```
/// 以订阅低延迟视频流的方式加入房间
- (void)joinRoom:(NSString *)roomId {
[RCLiveVideoEngine.shared joinRoom:roomId completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

订阅默认 CDN 视频流

在加入视频直播房间前，可设置订阅流的方式为默认 CDN，以及订阅流的分辨率和帧率，开发者需要提前做好准备工作。

> 准备工作:
>
> - 开通 融云 CDN 服务；
> - 在 Podfile 中添加 `RongCloudRTC/RongRTCPlayer` 依赖；
> - 使用自定义的分辨率和帧率，需要提交工单，客服会为您开通转码服务。

设置 CDN 配置代理

客户端在加入房间前，设置 RCSCDNDataSource 代理，并在 liveType 的实现里返回 RCSLiveTypeInnerCDN。

```objc
#import <RCLiveVideoLib/RCLiveVideoLib.h>
/// 设置 CDN 配置信息
[RCLiveVideoEngine.shared setCDNDataSource:self];

/// 观众订阅的流的类型
- (RCSLiveType)liveType {
return RCSLiveTypeInnerCDN;
}
```

设置订阅流的分辨率和帧率

加入房间，订阅 CDN 拉流时，可以设置 CDN 订阅流的分辨率 `innerCDNPreset` 和帧率 `innerCDNFPS`。注意，这种情况，需要提交工单，客服会为您开通转码服务。

```objc

/// 自定义内置 CDN 帧率
- (RCRTCVideoFPS)innerCDNFPS {
return RCRTCVideoFPS15;
}

/// 自定义内置 CDN 分辨率
- (RCRTCVideoSizePreset)innerCDNPreset {
return RCRTCVideoSizePreset1280x720;
}
```

订阅三方 CDN 视频流

除了默认 CDN 的方式，也可设置三方 CDN 的方式，开发者需要提前做好准备工作。

> 准备工作:
>
> - 开发者自行维护三方 CDN 的推拉流地址，主播设置推流地址，观众播放拉流地址；
> - 根据三方 CDN 推荐选择视频直播播放器，并实现 `RCSLivePlayer` 接口启动三方播放器。

设置 CDN 配置代理

客户端在加入房间前，设置 RCSCDNDataSource 代理，并在 liveType 的实现里返回 RCSLiveTypeThirdCDN。

```objc
#import <RCLiveVideoLib/RCLiveVideoLib.h>
/// 设置 CDN 配置信息
[RCLiveVideoEngine.shared setCDNDataSource:self];

/// 观众订阅的流的类型
- (RCSLiveType)liveType {
return RCSLiveTypeInnerCDN;
}
```

实现播放器接口

客户端在加入房间订阅流前，会通过 RCSCDNDataSource 代理获取播放器 thirdCDNPlayer，开发者需要实现播放器的开始和结束方法。

```objc

/// 自定义内置 CDN 帧率
- (RCRTCVideoFPS)innerCDNFPS {
return RCRTCVideoFPS15;
}

/// 自定义内置 CDN 分辨率
- (RCRTCVideoSizePreset)innerCDNPreset {
return RCRTCVideoSizePreset1280x720;
}
```

设置 CDN 配置代理

客户端在加入房间前，设置 RCSCDNDataSource 代理，并在 liveType 的实现里返回 RCSLiveTypeThirdCDN。

```objc
#import <RCLiveVideoLib/RCLiveVideoLib.h>
/// 设置 CDN 配置信息
[RCLiveVideoEngine.shared setCDNDataSource:self];

/// 观众订阅的流的类型
- (RCSLiveType)liveType {
return RCSLiveTypeThirdCDN;
}
```

实现播放器接口

客户端在加入房间订阅流前，会通过 RCSCDNDataSource 代理获取播放器 thirdCDNPlayer，开发者需要实现播放器的开始和结束方法。

```
/// 三方 CDN 配置
- (UIView<RCSLivePlayer> *)thirdCDNPlayer {
return RCSThirdPlayerView.shared;
}

/// 以 七牛云 播放器为例，实现三方 CDN 配置
/// 自定义 RCSThirdPlayerView 实现 RCSLivePlayer 协议
@interface RCSThirdPlayerView : UIView <RCSLivePlayer>

@property (nonatomic, strong) PLPlayerOption *option;
@property (nonatomic, strong) PLPlayer *player;

+ (instancetype)shared;

@end

@implementation RCSThirdPlayerView

+ (instancetype)shared {
static RCSThirdPlayerView *_shared = nil;
static dispatch_once_t onceToken;
dispatch_once(&onceToken, ^{
_shared = [RCSThirdPlayerView new];
});
return _shared;
}

#pragma mark - RCSLivePlayer delegate
/// 开始
- (void)start:(nonnull NSString *)roomId {
[self.player.playerView removeFromSuperview];
[self stop];

NSString *pullPath = @"三方 CDN 流地址";
NSURL *pullURL = [NSURL URLWithString:pullPath];
if (!pullURL) { return;}

PLPlayer *player = [[PLPlayer alloc] initWithURL:pullURL option:self.option];
if (!player) { return;}
if (!player.playerView) { return;}

[self addSubview:player.playerView];
[player.playerView mas_makeConstraints:^(MASConstraintMaker *make) {
make.edges.offset(0);
}];

[player play];

self.player = player;
}

/// 结束
- (void)stop {
[self.player stop];
self.player = nil;
}

#pragma mark - lazy load
- (PLPlayerOption *)option {
if (!_option) {
_option = [PLPlayerOption defaultOption];
[_option setOptionValue:@(15) forKey:PLPlayerOptionKeyTimeoutIntervalForMediaPackets];
[_option setOptionValue:@(2000) forKey:PLPlayerOptionKeyMaxL1BufferDuration];
[_option setOptionValue:@(1000) forKey:PLPlayerOptionKeyMaxL2BufferDuration];
[_option setOptionValue:@(NO) forKey:PLPlayerOptionKeyVideoToolbox];
}
return _option;
}

@end
```

离开视频直播

观众离开视频直播房间需调用 leaveRoom: 接口。

调用leaveRoom

在成功回调里对自己的业务Controller popViewController 即可。
失败回调里根据自己的业务需求自行处理。

```objc
#import <RCLiveVideoLib/RCLiveVideoLib.h>

[RCLiveVideoEngine.shared leaveRoom:^(RCLiveVideoCode code) {
if (code == RCLiveVideoSuccess) {
dispatch_async(dispatch_get_main_queue(), ^{
[self.navigationController popViewControllerAnimated:YES];
});
} else {
[SVProgressHUD showErrorWithStatus:[NSString stringWithFormat:LVSLocalizedString(@"live_finish_live_fail"),code]];
}
}];
```

观众连麦

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 上麦和下麦。

> 注意：
> 在视频直播 SDK 中，并没有权限的概念，房间中任何人都可以加入和离开视频直播连麦，也可以抱其他用户下麦，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、管理员可以抱其他用户下麦

上麦

房间内的观众都可以加入视频直播连麦，连麦后该观众的角色切换为主播，并会自动发布本地音视频流和订阅房间内其他主播的音视频流，同时，其他主播也会订阅该用户的音视频流。流的发布和订阅在 RCLiveVideoLib 中已实现，开发者不需要处理。

调用上麦方法

房间内观众调用 joinLiveVideoAtIndex:completion: 接口加入连麦。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>

/// 加入连麦
/// @param index 麦位序号，当传入 -1 时，自动查找第一个空麦位
[RCLiveVideoEngine.shared joinLiveVideoAtIndex:index completion:^(RCLiveVideoCode code) {

}];
```

处理上麦回调

成功调用自由上麦方法后，观众端会触发 liveVideoDidBegin 回调。
主播端和观众端都会触发 liveVideoUserDidUpdate 回调。

```
/// 上麦成功，可添加成功提示语等业务逻辑
- (void)liveVideoDidBegin:(RCLiveVideoCode)code;

/// 主播端和观众端都会触发，可添加更新用户列表等业务逻辑
- (void)liveVideoUserDidUpdate:(NSArray<NSString *> *)userIds;
```

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，离开麦位。离开麦位后，用户角色切换为观众，并取消发布本地音视频流，取消订阅其他主播的流，重新订阅合流。其他主播取消订阅该用户的音视频流。流的发布和订阅在 RCLiveVideoLib 中已实现，开发者不需要处理。

调用接口

房间连麦主播调用 leaveLiveVideo: 接口结束连麦。

```
#import <RCLiveVideoLib/RCLiveVideoLib.h>
/// 结束连麦
[RCLiveVideoEngine.shared leaveLiveVideo:^(RCLiveVideoCode code) {

}];
```

离开回调

下麦后，客户端收到 liveVideoDidFinish 回调，并携带离开原因 RCLiveVideoFinishReasonLeave。

```
/// 视频直播连麦结束
- (void)liveVideoDidFinish:(RCLivevideoFinishReason)reason {
/// TODO reason
}
```

强制下麦

强制下麦是指房主或其他有权限的用户将正在进行视频直播连麦的用户抱下麦。该操作直接由 KV 管控，被下麦的用户的信息会删除，角色也将变成观众，流的变更和下麦一致。

房主/管理员将某个用户下麦

调用 kickLiveVideo:completion: 接口让某个用户强制离开麦位。

```
///抱下麦
[RCLiveVideoEngine.shared kickLiveVideo:userId completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
```

下麦回调

被踢下麦后，用户收到 liveVideoDidFinish 回调，并携带离开原因 RCLiveVideoFinishReasonKick。

```
/// 视频直播连麦结束，kicked
- (void)liveVideoDidFinish:(RCLivevideoFinishReason)reason {
/// TODO reason
}
```

申请连麦

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 观众申请视频直播连麦。

功能介绍

房间内观众都可以发起连麦申请，观众发起的申请会被存储到申请记录中，主持人/管理员在收到申请通知后，可以选择同意或拒绝申请，在申请未被处理前，发起者也可以取消申请。申请被同意后，观众会切换角色，并发布本地音视频流和订阅远端用户流，同时，其他主播也会订阅该用户的音视频流。右侧时序图是通过申请方式连麦的过程，其中主要属性有：Client 客户端，有两种角色 host 和 audience；RCLiveVideoLib，视频直播 SDK。

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 `getRequests:` 接口查询上麦申请。

发起申请

观众调用 `requestLiveVideo:completion:` 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
- (void)requstLiveVideo:(NSInteger)index {
[RCLiveVideoEngine.shared requestLiveVideo:index completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

收到申请

房间内所有用户都会触发回调 `liveVideoRequestDidChange`，开发者可以根据业务需求，决定哪些用户能够接收和处理该事件。

```
/// 连麦申请变化回调
- (void)liveVideoRequestDidChange {
/// 首先获取申请列表
__weak typeof(self) weakSelf = self;
[RCLiveVideoEngine.shared getRequests:^(RCLiveVideoErrorCode code, NSArray<NSString *> *users) {
if (code == RCLiveVideoSuccess) {
weakSelf.requstUser = users;
[weakSelf.tableView reloadData];
} else {
/// TODO get requests error
}
}];
}
```

同意上麦

主持人（房主/管理员等具有权限的用户）调用接口 `acceptRequest:completion:` 同意连麦申请。

```
/// 同意申请
- (void)acceptRequst:(NSString *)userId {
[RCLiveVideoEngine.shared acceptRequest:userId completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

同意连麦回调

当连麦申请被同意后，观众端触发回调 `liveVideoRequestDidAccept`，同时 SDK 会切换用户角色，发布本地音视频流，并订阅其他主播音视频流，执行成功后，该用户会触发 `liveVideoDidBegin:` 回调。

```
/// 同意申请回调
- (void)liveVideoRequestDidAccept {
/// TODO
}
- (void)liveVideoDidBegin:(RCLiveVideoCode)code {
/// TODO
}
```

房主端拒绝上麦

主持人可以调用接口 `rejectRequest:completion:` 拒绝连麦申请。

```
/// 拒绝申请
- (void)rejectRequest:(NSString *)userId {
[RCLiveVideoEngine.shared rejectRequest:userId completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

观众端收到拒绝回调

当连麦申请被拒绝后，观众端触发回调 `liveVideoRequestDidReject`。

```
/// 拒绝申请回调
- (void)liveVideoRequestDidReject {
/// TODO
}
```

观众撤销申请

在连麦申请未被处理之前，观众调用接口 `cancelRequest:` 可以取消自己发起的连麦申请。当连麦申请被取消后，房间内用户触发回调 `liveVideoRequestDidChange` 回调。

```
/// 取消申请
- (void)cancelRequst {
[RCLiveVideoEngine.shared cancelRequest:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

邀请连麦

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 邀请观众视频直播连麦。

功能简介

邀请上麦由主持人（管理员）发起，当被邀请的观众收到邀请时，可以选择同意或拒绝；主持人在等待期间也可以取消邀请。右边为邀请连麦时序图，图中包含以下属性：Client 客户端，有两种角色 host 和 audience；RCLiveVideoLib，视频直播 SDK。

> 注意：
>
> 1. 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；
> 2. 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；

主播（管理员）发起邀请连麦

房主调用 inviteLiveVideo:atIndex:completion: 邀请用户上麦。您可以使用 atIndex 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
[RCLiveVideoEngine.shared inviteLiveVideo:userId atIndex:index completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
```

被邀请观众收到邀请通知

观众触发回调 liveVideoInvitationDidReceive:：

```
/// 收到邀请回调
- (void)liveVideoInvitationDidReceive:(NSString *)inviter atIndex:(NSInteger)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 acceptInvitationOfUser:atIndex:completion:，主播端触发回调 liveVideoInvitationDidAccept:。

```
/// 观众同意邀请
[RCLiveVideoEngine.shared acceptInvitationOfUser:inviter atIndex:index completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
/// 主播（管理员）收到同意邀请回调
- (void)liveVideoInvitationDidAccept:(NSString *)invitee {
/// TODO
}
```

观众拒绝上麦,主播收到观众拒绝邀请

观众处理邀请调用 rejectInvitationOfUser:completion:，主播端触发回调 liveVideoInvitationDidReject。

```
/// 拒绝邀请
[RCLiveVideoEngine.shared rejectInvitationOfUser:inviter completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
/// 主播（管理员）收到拒绝邀请回调
- (void)liveVideoInvitationDidReject:(NSString *)invitee {
/// TODO
}
```

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:completion: 取消上麦邀请，观众端触发回调 liveVideoInvitationDidCancel。

```
///主播（管理员）取消邀请回调
[RCLiveVideoEngine.shared cancelInvitation:userId completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
///观众收到 取消邀请回调
- (void)liveVideoInvitationDidCancel {
/// TODO user action
}
```

连麦布局

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 设置视频直播连麦布局。

功能简介

视频直播 SDK 封装了视频渲染和视频布局等复杂逻辑。根据主流连麦场景，内置了 7 种常见的视频合流布局模式，也支持自定义模式，每种布局支持添加视图和事件。布局类型为枚举类型，开发者可以设置布局类型无缝切换。

连麦布局类型

```
typedef NS_ENUM(NSInteger, RCLiveVideoMixType) {
/// 默认模式
RCLiveVideoMixTypeDefault = 0,
/// 房主模式
RCLiveVideoMixTypeOneToOne,
RCLiveVideoMixTypeOneToSix,
/// 格子模式
RCLiveVideoMixTypeGridTwo,
RCLiveVideoMixTypeGridThree,
RCLiveVideoMixTypeGridFour,
RCLiveVideoMixTypeGridSeven,
RCLiveVideoMixTypeGridNine,
/// 自定义模式
RCLiveVideoMixTypeCustom,
};
```

布局类型视图

类型	简介	示图
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。对应的枚举类型为：RCLiveVideoMixTypeDefault、RCLiveVideoMixTypeOneToOne。	
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。对应的枚举类型为：RCLiveVideoMixTypeOneToSix。	
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。对应的枚举类型为：RCLiveVideoMixTypeGridTwo，RCLiveVideoMixTypeGridThree，RCLiveVideoMixTypeGridFour，RCLiveVideoMixTypeGridSeven，RCLiveVideoMixTypeGridNine。	

设置连麦布局

房间直播视频布局方式由主持人控制，主持人通过更改不同的布局类型，SDK 内部触发合流布局更新，麦位对应的渲染区域也会改变。右边为更新布局的时序图，图中包含以下属性：Client 客户端，有两种角色 host 和 audience；RCLiveVideoLib，视频直播 SDK。

> 注意：连麦布局发生改变后，所有连麦的主播会被强制下麦；

添加预览视图

视频直播 SDK 封装了不同角色的视频渲染和视频布局等复杂逻辑，开发者只需用 previewView 接口展示视图即可。

```
/// 在viewDidLoad里一般会这样设置
- (void)viewDidLoad {
[super viewDidLoad];
/// 添加视频预览
UIView *previewView = [RCLiveVideoEngine.shared previewView];
previewView.frame = self.view.bounds;
[self.view addSubview:previewView];
```

设置事件代理

调用 mixDelegate 设置代理，并实现代理事件回调方法。

```
RCLiveVideoEngine.shared.mixDelegate = self;
}

///当麦位信息类型发生变化的时候，回调会被触发
- (void)roomMixTypeDidChange:(RCLiveVideoMixType)mixType {
/// TODO setup UI
}
```

更改布局类型

调用 setMixType:completion: 接口设置直播连麦类型。设置成功后，房间内用户会收到 roomMixTypeDidChange 回调，开发者可以在此方法更新 UI 布局。

```
- (void)setupMixType:(RCLiveVideoMixType)type {
[RCLiveVideoEngine.shared setMixType:type completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

连麦布局更新

当连麦布局发生变化后，布局对应的麦位信息更新接口 liveVideoDidLayout:withFrame: 也会回调，开发者可以在此方法自定义麦位 UI。

```
- (void)liveVideoDidLayout:(RCLiveVideoSeat *)seat withFrame:(CGRect)frame {
/// Custom Seat View
}
```

自定义连麦布局

当内置布局不满足时，可以自定义布局，客户端需要实现代理 RCLiveVideoMixDataSource，返回麦位位置 liveVideoFrames 和 画布大小 liveVideoPreviewSize。

```
/// 设置自定义模式
- (void)setupCustomMixType {
RCLiveVideoEngine.shared.mixDataSource = self;
[RCLiveVideoEngine.shared setMixType:RCLiveVideoMixTypeCustom completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}

#pragma mark - RCLiveVideoMixDataSource -
/// 画布尺寸
- (CGSize)liveVideoPreviewSize {
return CGSizeMake(720, 1280);
}
/// 麦位位置，CGRectMake 参数取值范围均为 [0,1]
- (NSArray<NSValue *> *)liveVideoFrames {
return @[
@(CGRectMake(0.0000, 0.0000, 0.5000, 1.0000)),
@(CGRectMake(0.5000, 0.0000, 0.5000, 1.0000)),
];
}
```

麦位管理

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 管理和同步麦位信息。

功能介绍

直播间的主持人或麦位上的用户，可以通过麦位管理控制连麦：锁麦、禁麦、开关视频、开关麦克风、大小流等，麦位状态会实时同步房间内的所有用户。右侧是麦位控制时序图，开发者可以在 RCLiveVideoSeat 中查看接口。

麦位锁定

调用 setLock:completion: 接口更新麦位锁定状态。房间内其他用户收到 seat:didLock: 回调，如果麦位锁定，观众不能在该麦位上麦。

```
///锁定麦位
- (void)switchLock:(RCLiveVideoSeat *)seat {
BOOL lock = !seat.lock;
[seat setLock:lock completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
/// 房间内用户收到回调
- (void)seat:(RCLiveVideoSeat *)seat didLock:(BOOL)isLocked {
/// refresh UI
}
```

麦位静音

调用 setMute:completion: 接口更新麦位静音状态。房间内其他用户收到 seat:didMute: 回调，麦位静音状态变化不影响观众上下麦。

```
- (void)switchMute:(RCLiveVideoSeat *)seat {
BOOL mute = !seat.mute;
[seat setMute:mute completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
/// 房间内用户收到回调
- (void)seat:(RCLiveVideoSeat *)seat didMute:(BOOL)isMuted {
/// refresh UI
}
```

开关视频

麦位用户调用 setUserEnableVideo:completion: 接口更新是否开启视频。房间内其他用户收到 seat:didUserEnableVideo: 回调。

```
- (void)switchEnableVideo:(RCLiveVideoSeat *)seat {
BOOL enable = !seat.userEnableVideo;
[seat setUserEnableVideo:enable completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
/// 房间内用户收到回调
- (void)seat:(RCLiveVideoSeat *)seat didUserEnableVideo:(BOOL)enable {
/// refresh UI
}
```

用户开关音频

调用 setUserEnableAudio:completion: 接口更新麦位用户是否开启音频。房间内其他用户收到 seat:didUserEnableAudio: 回调。

```
- (void)switchEnableAudio:(RCLiveVideoSeat *)seat {
BOOL enable = !seat.userEnableAudio;
[seat setUserEnableAudio:enable completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
/// 房间内用户收到回调
- (void)seat:(RCLiveVideoSeat *)seat didUserEnableAudio:(BOOL)enable {
/// refresh UI
}
```

麦位音频回调

麦上用户讲话，房间内用户触发 seat:didSpeak: 回调。

```
/// 房间内用户收到回调
- (void)seat:(RCLiveVideoSeat *)seat didSpeak:(NSInteger)audioLevel {
/// refresh UI
}
```

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 YES，除非特殊场景，不建议您改变该属性。

```
/// 控制本端订阅分流时是否采用小流，默认为 YES
@property (nonatomic, assign) BOOL enableTiny;
```

跨房间 PK

更新时间:2024-05-17

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。

发起 PK 邀请

主播 A 调用发起 PK 邀请接口邀请其他主播连麦 PK，被邀请的主播 B 会收到 PK 邀请回调。

```
/// 主播 A 发起 PK 邀请
- (void)sendPKInvitation:(NSString *)roomId userId:(NSString *)userId {
[RCLiveVideoEngine.shared sendPKInvitation:roomId invitee:userId completion:^(RCLiveVideoCode code) {
/// TODO code
}];
}
/// 主播 B 接收到 PK 邀请
- (void)didReceivePKInvitationFromRoom:(NSString *)inviterRoomId byUser:(NSString *)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:inviter:completion:` 接口
/// 3、如果拒绝，调用 `rejectPKInvitation:inviter:reason:completion:` 接口
/// 如果有其他原因无法：超时、忙碌等，调用 `rejectPKInvitation:inviter:reason:completion:` 接口
}
```

撤销 PK 邀请

发起邀请的主播 A 调用撤销 PK 邀请接口取消已经发送的 PK 邀请，被邀请的主播 B 会收到撤销 PK 邀请回调。

```
/// 取消 PK 邀请
- (void)cancelPKInvitation:(NSString *)roomId userId:(NSString *)userId {
[RCLiveVideoEngine.shared cancelPKInvitation:roomId invitee:userId completion:^(RCLiveVideoCode code) {
/// TODO code
}];
}
/// 接收到 PK 邀请取消
- (void)didCancelPKInvitationFromRoom:(NSString *)inviterRoomId byUser:(NSString *)inviterUserId {
/// 一般可以这样操作
/// 取消弹窗或更新 UI
}
```

同意 PK 邀请

被邀请的主播 B 调用同意 PK 邀请接口同意 PK 邀请，发起邀请的主播 A 接收到同意 PK 邀请回调，两个房间的所有用户收到 PK 开始的回调。

```
/// 主播 B 同意并开启 PK
- (void)acceptPKInvitation:(NSString *)roomId userId:(NSString *)userId {
[RCLiveVideoEngine.shared acceptPKInvitation:roomId inviter:userId completion:^(RCLiveVideoCode code) {
/// TODO code
}];
}
- (void)didAcceptPKInvitationFromRoom:(NSString *)inviteeRoomId
byUser:(NSString *)inviteeUserId {
/// TODO PK invitation is accepted & PK will begin
}
- (void)didBeginPK:(RCLiveVideoCode)code {
/// TODO PK did begin
}
```

拒绝 PK 邀请

被邀请的主播 B 调用拒绝 PK 邀请接口同意 PK 邀请，发起邀请的主播 A 接收到拒绝 PK 邀请的回调。

```
/// 拒绝 PK 邀请
- (void)rejectPKInvitation:(NSString *)roomId userId:(NSString *)userId {
[RCLiveVideoEngine.shared rejectPKInvitation:roomId inviter:userId reason:@"拒绝 PK 邀请原因" completion:^(RCLiveVideoCode code) {
/// TODO code
}];
}
- (void)didRejectPKInvitationFromRoom:(NSString *)inviteeRoomId
byUser:(NSString *)inviteeUserId
reason:(NSString *)reason {
/// TODO reason
}
```

声音控制

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

```
/// 控制 PK 声音
- (void)mutePKUser:(BOOL)isMute {
[RCLiveVideoEngine.shared mutePKUser:isMute completion:^(RCLiveVideoCode code) {
/// 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 didBeginPK: 的回调。

```
/// 恢复 PK
- (void)resumePK:(RCLiveVideoPK *)PK {
[RCLiveVideoEngine.shared resumePK:PK completion:^(RCLiveVideoCode code) {
/// TODO code
}];
}
- (void)didBeginPK:(RCLiveVideoCode)code {
/// TODO code
}
```

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 didFinishPK: 回调

```
/// 结束 PK
- (void)quitPK {
[RCLiveVideoEngine.shared quitPK:^(RCLiveVideoCode code) {
/// TODO code
}];
}
- (void)didFinishPK:(RCLiveVideoCode)code {
/// TODO code
}
```

房间属性

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 自定义视频直播房间属性。

功能介绍

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。右边为时序图。

> 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。

设置房间属性

房间内任何用户都可以调用 setRoomInfo:completion: 方法，设置房间自定义属性。

```
/// 设置房间名称
- (void)updateRoomName:(NSString *)roomName {
[RCLiveVideoEngine.shared setRoomInfo:@{@"roomName": @"room name"} completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
/// 设置房间公告
- (void)updateRoomName:(NSString *)roomName {
[RCLiveVideoEngine.shared setRoomInfo:@{@"roomNotice": @"room notice"} completion:^(RCLiveVideoErrorCode code) {
/// TODO code
}];
}
```

房间属性更新

当用户刚加入房间或自定义属性变化时，会触发 roomInfoDidUpdate:value: 回调。

```
//收到房间回调
- (void)roomInfoDidUpdate:(NSString *)key value:(NSString *)value {
if ([key isEqualToString:@"roomName"]) {
/// TODO room name update
} else if ([key isEqualToString:@"roomNotice"]) {
/// TODO room notice update
}
}
```

用户管理

更新时间:2024-05-17

本节介绍如何使用 RCLiveVideoLib 管理用户。

功能介绍

主持人可以将用户踢出房间，时序图参考右边。

> 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。

踢出房间

调用 kickOutRoom:completion: 接口，被踢出用户接收到 userDidKickOut:byOperator: 回调。

/// 房主将用户踢出房间

- (void)kickOut:(NSString *)userId {
  [RCLiveVideoEngine.shared kickOutRoom:userId completion:^(RCLiveVideoErrorCode code) {
  /// TODO code
  }];
  }

/// 被踢出房间的用户接收到回调，离开房间

- (void)userDidKickOut:(NSString *)userId byOperator:(NSString *)operatorId {
  if (![userId isEqualToString:currentUserId]) return;
  ######## (step2)
  [RCLiveVideoEngine.shared leaveRoom:^(RCLiveVideoErrorCode code) {
  if (code == RCLiveVideoSuccess) {
  /// TODO pop 或者 dismiss
  } else {
  /// TODO error
  }
  }];
  ######## (step2)
  }

发送消息

更新时间:2024-05-17

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 completion 结果回调
- (void)sendMessage:(RCMessageContent *)content completion:(nonnull RCLVResult)completion;

//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
- (nullable RCMessage *)sendMessage:(RCConversationType)conversationType
targetId:(NSString *)targetId
content:(RCMessageContent *)content
pushContent:(nullable NSString *)pushContent
pushData:(nullable NSString *)pushData
success:(nullable void (^)(long messageId))successBlock
error:(nullable void (^)(RCErrorCode nErrorCode, long messageId))errorBlock;
```

更新日志 2.1.1.2

更新时间:2024-05-17

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

2.1.0.5

功能优化：

1. 灵活调度音视频流

2.1.0.3

功能优化：

1. 开放 SDK 依赖版本，不再限制 IM 和 RTC 版本

2.1.0.4

新增功能：

1. 添加合流参数将要更新回调

2.1.0.2

修复问题：

1. 修复语音过滤问题：房主没声音
2. 将 PK 模型类添加到 RCLiveVideoLib 文件中

2.1.0.1

修复问题：

1. 修复视频镜像
2. 快速切换麦位加锁

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

接口变更：

1. 用户踢出房间回调 `didKickOutRoom:byOperator:`，变更为 `userDidKickOut:byOperator:`

修复问题：

1. 修复角色切换时重复订阅合流的问题

2.0.0

新增功能：

1. 添加 7 种内置连麦模式和自定义连麦模式
2. 添加麦位管理功能 `RCLiveVideoSeat`
3. 添加自定义布局代理 `RCLiveVideoMixDataSource` 和 `RCLiveVideoMixDelegate`
4. 添加房间连麦模式更新回调 `roomMixTypeDidChange:`

接口变更：

1. 麦位布局更新回调 `didLiveVideoLayout:` ，变更为 `liveVideoDidLayout:withFrame:`
2. 移除 `didClickUser:` 回调，请在 `liveVideoDidLayout:withFrame:` 回调中，自定义 UI 和处理事件
3. 直播结束回调 `liveVideoDidFinish:` 添加结束原因 `RCLivevideoFinishReason`
4. 邀请连麦接口 `inviteLiveVideo:atIndex:completion:` 添加指定麦序
5. 邀请连麦回调 `liveVideoInvitationDidReceive:atIndex:` 添加邀请者和麦序

常见问题　　创建视频直播房间返回失败

更新时间:2024-05-17

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，在视频连麦开始时，房主会重新更新布局，房间内其他用户会收到布局更新的回调 `didLiveVideoLayout`。开发者可以根据回调方法携带的布局信息，自定义 UI 接收点击事件。另外，如果 previewView 没有遮挡，可以通过回调方法 `didClickUser` 接收事件。

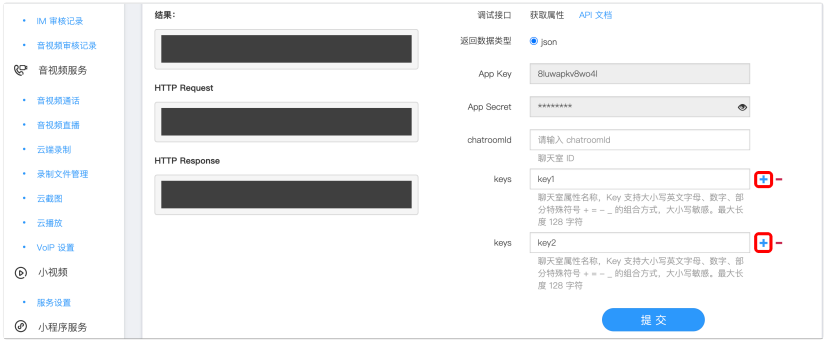
如何利用控制台查询房间属性（KV）

遇到设置房间信息相关的问题，可以通过追踪后台房间 KV 数据去定位问题。

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:2024-05-17

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。

iOS 相关资源

- **RC RTC 开源项目**

  GitHub · Gitee

- **QuickDemo**

  GitHub · Gitee

状态码

更新时间:2024-05-17

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
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
80401	连麦失败
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	决绝连麦申请失败
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
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

状态码	说明
80703	取消 PK 邀请失败
80704	响应 PK 邀请失败
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