

# 会议 / 直播

## uni-app 5.X

---

2024-08-30

# 实时音视频开发指导

# 实时音视频开发指导

更新时间:2024-08-30

欢迎使用融云实时音视频（RTC）。RTC 服务基于房间模型设计，可以支持一对一音视频通信、和多人音视频通信。底层基于融云 IM 信令通讯，可保障在长时间音视频通话及弱网情况下保持正常连通。

本页面简单介绍融云 RTC 服务能力和 SDK 产品。

## 客户端 SDK

融云客户端 SDK 提供丰富的接口，大部分能力支持开箱即用。配合 RTC 服务端 API 接口，可满足丰富的业务特性要求。

### 提示

[前往融云产品文档](#) · [客户端 SDK 体系](#) · [RTCLib](#) · [CallKit](#) · [CallLib](#) · [CallPlus](#) >

## SDK 适用场景

CallPlus、CallLib/Kit、RTCLib 是融云 RTC 服务提供的三款经典的客户端 SDK。其中 CallPlus、CallLib/Kit 用于开发音视频通话（呼叫）业务。RTCLib 是音视频基础能力库，可满足类似会议、直播等业务场景需求，具备较高的扩展与定制属性。

业务分类	适用的 SDK	流程差异	场景描述
通话（呼叫）	CallPlus、CallLib/Kit	SDK 内部呼叫流程自动处理房间号	拨打音视频电话（类比微信音视频通话）
会议	RTCLib	与会者需要约定房间号，参会需进入同一房间	线上会议、小班课、在线视频面试、远程面签等
直播	RTCLib	支持区分主播、观众角色。观众可通过连麦进行发言。	直播社交、大型发布会、语聊房、线上大班课等

## 如何选择 SDK

不同 SDK 适用的业务场景差异较大，请您谨慎选择并决策。

- **CallPlus** 与 **CallLib/Kit** 用于实现通话（呼叫）功能的客户端库。封装了拨打、振铃、接听、挂断等一整套呼叫流程，支持一对一及群组内多人呼叫的通话能力。CallPlus、CallLib/Kit 均依赖 RTCLib，两者区别如下：
  - **【推荐】** CallPlus 是融云新一代针对音视频呼叫场景的 SDK，后续新的产品特性和持续迭代均以 CallPlus 为重点。
  - CallLib/Kit 是老版本的音视频通话 SDK，CallLib 不含任何 UI 界面组件，CallKit 提供了呼叫相关的通用 UI 组件库。
  - CallPlus 与 CallLib/Kit 使用完全不同的后端服务架构实现音视频通话（呼叫）功能，因此与 CallLib/Kit 并不互通。暂不支持从 CallLib/Kit 平滑迁移至 CallPlus。
- **RTCLib** 是融云音视频核心能力库。应用开发者可将 RTCLib 用于支持直播、会议业务场景。

具体选择建议如下：

- 不需要通话（呼叫）功能，可使用 RTCLib，即您仅需要融云为您的 App 提供实时音视频（RTC）核心能力。
- 需要开发支持通话（呼叫）的音视频应用，但不希望自行实现呼叫 UI，可使用 CallKit。直接利用融云提供的呼叫 UI，节省开发时间。
- 需要开发支持通话（呼叫）的音视频应用，不希望 SDK 带任何 UI 组件，可使用 CallPlus、CallLib，推荐您使用 CallPlus。
- 通过融云提供的独立功能插件扩展客户端 SDK 的功能。

在使用融云 SDK 进行开发之前，我们建议使用快速上手教程与示例项目进行评估。

## 高级和扩展功能

RTC 服务支持的高级与扩展功能，包括但不限于以下项目：

- 跨房间连麦：支持多主播跨房间连麦 PK 直播。
- 通话数据统计：按照指定的时间间隔上报通话的详细数据。
- 屏幕共享：通过自定义视频流的方式在房间内发起屏幕共享功能。
- 自定义加密：可选择对媒体流进行加密，从而保障用户的数据安全。
- 插件支持：支持通过插件实现美颜、CDN 播放器等功能。
- 云端录制：在音视频通话（呼叫）、直播、会议时分别录制每个参与者的音视频、或合并后进行录制。
- 内容审核：融云媒体服务器（RTC Server）把收到的音视频流转码后送审，审核结果返回应用服务器。

部分功能需配合 RTC 服务端 API 使用。具体支持的功能与平台相关。具体使用方法请参见客户端 SDK 开发文档或服务端开发文档。

## 平台兼容性

CallKit、CallLib、RTCLib 均支持主流移动操作平台，客户端功能在多端基本保持一致，支持多平台互通。CallPlus 暂仅支持 Android、iOS、Web 平台。

平台/框架	接口语种	支持架构	说明
<b>Android</b>	Java	armeabi-v7a、arm64-v8a、x86、x86-64	系统版本 4.4 及以上
<b>iOS</b>	Objective-C	---	系统版本 9.0 及以上
<b>Windows</b>	C++、Electron	x86、x86-64	Windows 7 及以上
<b>Linux</b>	C、Electron	---	推荐 Ubuntu 16.04 及以上；其他发行版需求请咨询商务
<b>MacOS</b>	Electron	---	系统版本 10.10 及以上
<b>Web</b>	Javascript	---	详见客户端文档「Web 兼容性」
<b>Flutter</b>	dart	---	Flutter 2.0.0 及以上
<b>uni-app</b>	Javascript	---	uni-app 2.8.1 及以上
<b>React Native</b>	Javascript	---	React Native 0.65 及以上
<b>Unity</b>	C#	Android(armeabi-v7a、arm64-v8a) iOS(arm64,armv7)	---

## 版本支持

RTC 服务客户端 SDK 针对各平台/框架提供的最新版本如下（--- 表示暂未支持）：

SDK/平台	Android	iOS	Web	Electron	Flutter	Unity	uni-app	小程序	React Native	Windows - C++	Linux - C
RTCLib	5.6.x	5.6.x	5.6.x	5.6.x	5.2.x	5.2.x	5.2.x	5.0.x	5.2.x	5.1.x	见注 <sup>1</sup>
CallLib	5.6.x	5.6.x	5.0.x	5.1.x	5.1.x	---	5.1.x	3.2.x	5.1.x	---	---
CallKit	5.6.x	5.6.x	---	---	---	---	---	---	---	---	---
CallPlus	2.x	2.x	2.x	---	---	---	---	---	---	---	---

注 1：关于 Linux 平台的支持，请咨询融云的商务。

## SDK 体积对比

### Android 端

以下数据基于 RTC 5.X 版本。

CPU 架构	集成 RTCLib 增量	集成 CallLib 增量	集成 CallKit 增量
armeabi	4.5MB	4.6MB	7.4MB
arm64-v8a	5.1MB	5.1MB	8.0MB
x86	5.4MB	5.4MB	8.3MB
全平台	17.2MB	17.2MB	20.1MB

### iOS 端

以下数据基于 RongCloudRTC 5.X 版本。

CPU 架构	集成 RTCLib 增量	集成 CallLib 增量	集成 CallKit 增量
arm64	4.3M	4.4M	8.9M
arm64 + armv7	8.6M	8.9M	14.8M

## 实时音视频服务端


实时音视频服务端 API 可以协助您构建集成融云音视频能力的 App 后台服务系统。

您可以使用服务端 API 将融云服务集成到您的实时音视频服务体系中。例如，向融云获取用户身份令牌 (Token)，从应用服务端封禁用户、移出房间等。

 提示

[前往融云服务端开发文档 · 集成必读](#) >>

## 控制台

使用[控制台](#) ，您可以对开发者账户和应用进行管理，开通音视频服务，以及其他高级服务，查看应用数据报表，和计费数据。

 提示

音视频服务必须要从控制台开通后方可使用。参见[开通音视频服务](#)。

## 实时音视频数据

您可以前往控制台的[数据统计页面](#)，查询、查看音视频用量、业务健康检查等数据。开通相应服务后，还能获取如业务数据分析等数据。

融云还提供通话质量实时的监控工具，以图表形式展示每一通音视频通话的质量数据，帮助定位通话问题，提高问题解决效率。

融云不会利用客户的数据。同时融云提供完善的数据隐私保护策略。参见 [SDK 隐私政策](#)。

## 导入 SDK

## 导入 SDK

更新时间:2024-08-30

由于 uni-app 的 SDK 是在 uni 原生插件的基础上封装了 Typescript 调用层，导入 SDK 时，必须先引入 uni 原生插件。

### 步骤 1：导入 uni 原生插件

uni 原生插件均已上架 uni-app 插件市场。

- 即时通讯 uni 原生插件 [RCUniIMV2](#)
- 实时音视频 uni 原生插件 [RCUniRtc](#)

请使用 [HBuilder X](#) 将即时通讯 [RCUniIMV2](#) 插件和实时音视频 [RCUniRtc](#) 插件导入应用工程。

1. 前往 uni-app 插件市场，购买（0 元）或 下载融云 uni-app 原生插件 [RCUniIMV2](#) 和 [RCUniRtc](#)。
2. 使用 HBuilder X 导入原生插件，并完成相应配置。请根据项目打包方式，选择合适的步骤。
  - 云打包适用：
    1. 在插件市场操作 [购买（0 元）for 云打包](#) 后，然后在 HBuilder X 中，打开项目的 `manifest.json` 文件。
    2. 点击 **App**原生插件配置 -> 选择云端插件 -> 选中 **RCUniIM/RCUniRtc**。
  - 本地打包适用：
    1. 使用 HBuilder X 在项目根目录下创建 `nativeplugins` 文件夹。
    2. 将下载的插件解压之后放入 `nativeplugins` 文件夹中。
    3. 在 HBuilder X 中，打开项目的 `manifest.json` 文件。
    4. 点击 **App**原生插件配置 -> 选择本地插件 -> 选中 **RCUniIM/RCUniRtc**。

请参照以下 `nativeplugins` 文件目录结构：

```

nativeplugins
├── RongCloud-IM
│   ├── android
│   │   └── RCUniIM.aar
│   ├── ios
│   │   ├── RCUniIM.xcframework
│   │   ├── RongChatRoom.xcframework
│   │   └── RongIMLibCore.xcframework
│   └── package.json
├── RongCloud-RTC
│   ├── android
│   │   ├── RCUniRtc-release.aar
│   │   └── libs
│   │       └── RongRTCWrapper.jar
│   ├── ios
│   │   ├── RCUniRtc.xcframework
│   │   ├── RongIMLibCore.xcframework
│   │   ├── RongRTCLib.xcframework
│   │   └── RongRTCLibWrapper.xcframework
│   └── package.json

```

目录说明：

- `android` 目录：包含融云 uni-app Android 原生插件
- `ios` 目录：包含融云 uni-app iOS 原生插件
- `package.json`：插件的依赖

## 步骤 2：安装 Typescript 依赖项

原生插件配置完成后，还需要安装两个 Typescript 层的依赖项。

uni Typescript 插件均已上架 uni-app 插件市场。

- [即时通讯 Typescript 插件 RongCloud-IMWrapper-V2](#) [↗](#)
- [实时音视频 Typescript 插件 RongCloud-RTCWrapper](#) [↗](#)

## 步骤 3：在代码中导入 SDK

```

// IM
import RCIMIEngine from "@uni_modules/RongCloud-IMWrapper-V2/js_sdk/RCIMEEngine";
// RTC
import RCRTCEngine from '@uni_modules/RongCloud-RTCWrapper/lib/RCRTCEngine';

```

版本依赖说明

依赖SDK	版本
RTCLib	5.1.17



## 初始化

## 初始化

更新时间:2024-08-30

在使用 SDK 其它功能前，必须先进行初始化。本文中 will 详细说明初始化的方法。

### 注意事项

- 必须在应用生命周期内调用初始化方法，只需要调用一次。

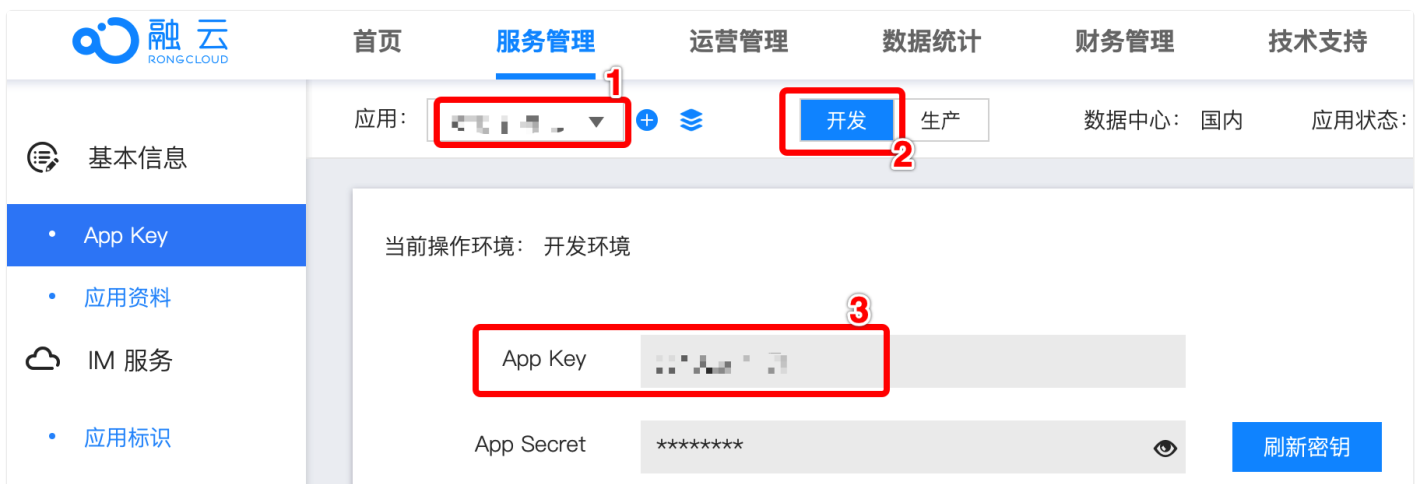
### 准备 App Key

您必须拥有正确的 App Key，才能进行初始化。

您可以[控制台](#)，查看您已创建各个应用的 App Key。

如果您拥有多个应用，请注意选择应用名称（下图中标号 1）。另外，融云的每个应用都提供用于隔离生产和开发环境的两套独立 App Key / Secret。在获取应用的 App Key 时，请注意区分环境（生产 / 开发，下图中标号 2）。

- 如果您并非应用创建者，我们建议在获取 App Key 时确认页面上显示的数据中心是否符合预期。
- 如果您尚未向融云申请应用上线，仅可使用开发环境。



### 初始化之前

部分配置必须在初始化之前完成，否则 SDK 功能无法正常工作。

- 开通音视频服务：音视频服务需要手动开通。请根据应用的具体业务类型，开通对应的音视频服务。详细说明请参见[开通音视频服务](#)。
- 海外数据中心：因为音视频业务依赖即时通讯业务 IMLib 提供信令通道，如果您的应用使用海外数据中心，必须在初始化之前修改 IMLib SDK 默认连接的服务地址为海外数据中心地址。否则 SDK 默认连接中国国内数据中心服务地址。详细说明

请参见[配置海外数据中心服务地址](#)。

## 权限配置

iOS 需要在项目的 manifest.json 中 App 权限配置 - iOS 隐私信息访问的许可描述中加入摄像头和麦克风的使用描述

Android 需要在项目的 manifest.json 中 App 权限配置 - Android 权限配置中勾选相机、麦克风和网络的使用权限

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
```

## 初始化 IMLib

音视频 SDK 是基于即时通信 SDK 作为信令通道的，所以要先初始化 IM SDK。如果不换 AppKey，在整个应用生命周期中，初始化一次即可。

```
// IMLib 初始化
let appKey = '从控制台申请的 AppKey';
let options = {};
let IMEngine = null;
RCIMWEngine.create(appKey, options).then((res) => {
//本地代码保存引擎
IMEngine = res
});
```

## 连接 IM 服务

音视频用户之间的信令传输依赖于融云的即时通信（IM）服务，因此需要先调用 connect 与 IM 服务建立好 TCP 长连接。建议在功能模块的加载位置处调用，之后再行音视频业务。当模块退出后调用 disconnect 或 logout 断开该连接。

```
IMEngine.setOnConnectedListener((res) => {
if (res.code === 0) {
// 连接成功
}
});
let code = await IMEngine.connect(token, timeout)
```

## 初始化 RTC 引擎

在成功连接 IM 服务之后，您可以通过以下代码，初始化 RTC 引擎。

```
let engine = RCRTCEngine.create();
```

或者通过以下方式传递配置项创建引擎。详细说明请参见[引擎配置](#)。

```
let setup = {reconnectable: false, statsReportInterval: 1000}  
let engine = RCRTCEngine.create(setup)
```

## 标准会议流程

## 标准会议流程

更新时间:2024-08-30

在开始之前，请您再次确认您已经完成了[开通音视频服务](#)、[导入 SDK](#)以及[初始化](#)。

### 设置本地事件回调监听

#### 设置加入房间事件回调

```
engine.setOnRoomJoinedListener(({code, message}) => {  
  if (code === 0) {  
    // 加入房间成功  
  } else {  
    // 加入房间失败  
  }  
});
```

#### 设置发布资源事件回调

```
engine.setOnPublishedListener(({type, code, message}) => {  
  // type 发布资源的类型 RCRTCMediaType  
  if (code === 0) {  
    // 发布成功  
  } else {  
    // 发布失败  
  }  
});
```

#### 设置取消发布资源事件回调

```
engine.setOnUnpublishedListener(({type, code, message}) => {  
  // type 取消发布资源的类型 RCRTCMediaType  
  if (code === 0) {  
    // 取消发布成功  
  } else {  
    // 取消发布失败  
  }  
});
```

#### 设置订阅资源事件回调

```
engine.setOnSubscribedListener(({userId, type, code, message}) => {
// userId 远端用户 ID
// type 订阅资源的类型 RCRTCMediaType
if (code === 0) {
// 订阅成功
} else {
// 订阅失败
}
});
```

设置取消订阅资源事件回调

```
engine.setOnUnsubscribedListener(({userId, type, code, message}) => {
// userId 远端用户 ID
// type 取消订阅资源的类型 RCRTCMediaType
if (code === 0) {
// 取消订阅成功
} else {
// 取消订阅失败
}
});
```

## 设置远端事件回调监听

设置远端用户加入房间事件回调

```
engine.setOnUserJoinedListener(({userId, roomId}) => {
// userId 远端用户 ID
// roomId 房间 ID
});
```

设置远端用户离开房间事件回调

```
engine.setOnUserLeftListener(({userId, roomId}) => {
// userId 远端用户 ID
// roomId 房间 ID
});
```

设置远端用户离线事件回调

```
engine.setOnUserOfflineListener(({userId, roomId}) => {
// userId 远端用户 ID
// roomId 房间 ID
});
```

设置远端用户发布资源回调

```
engine.setOnRemotePublishedListener(({userId, roomId, type}) => {
// userId 远端用户 ID
// roomId 房间 ID
// type 远端用户发布的资源类型 RCRTCMediaType
});
```

设置远端用户取消发布资源回调

```
engine.setOnRemoteUnpublishedListener(({userId, roomId, type}) => {
// userId 远端用户 ID
// roomId 房间 ID
// type 远端用户发布的资源类型 RCRTCMediaType
});
```

## 加入房间

以会议成员身份 (MeetingMember) 加入音视频会议

```
let setup = {
type: RCRTCMediaType.AudioVideo
role: RCRTCRole.MeetingMember
};
engine.joinRoom('会议号', setup);
```

## 本地预览

打开前置摄像头采集视频

```
engine.enableCamera(true, RCRTCCamera.Front);
```

导入预览窗口组件

### ⚠ 警告

RCRTCView 内部包装的是原生提供的组件，所以 RCRTCView 只能在 nvue 文件中使用。

```
// 导入 RCRTCView
import RCRTCView from '@uni_modules/RongCloud-RTCWrapper/components/RCRTCView';

// 声明 RCRTCView
export default {
components: {
RCRTCView,
},
}
```

## 添加预览窗口

```
<!-- 增加 RCRTCView 组件, fitType: 视频填充模式, mirror: 视频是否镜像显示 -->
<RCRTCView class="localView" ref="localView" :fitType="RCRTCViewFitType.Center" :mirror="true">
</RCRTCView>
```

## 设置预览窗口

```
// 设置预览窗口
engine.setLocalView(this.$refs.localView.getNativeViewRef(), (code) => {
if (code === 0) {
// 设置成功
} else {
// 设置失败
}
});
```

# 发布资源

## 发布音视频资源

```
engine.publish(RCRTCMediaType.AudioVideo);
```

# 订阅远端用户资源

远端用户发布资源后，订阅远端用户音视频资源

```
engine.subscribe(userId, RCRTCMediaType.AudioVideo);
```

远端用户发布资源后，设置远端用户预览窗口

```
engine.setRemoteView(userId, this.$refs.remoteView.getNativeViewRef(), (code) => {
if (code === 0) {
// 设置成功
} else {
// 设置失败
}
});
```

## 标准低延迟直播流程

## 标准低延迟直播流程

更新时间:2024-08-30

在开始之前，请您再次确认您已经完成了[开通音视频服务](#)、[导入 SDK](#)以及[初始化](#)。

### 提示

多人之间想要发起音视频通话，需要加入同一个音视频房间。对于直播需求来讲，资源类型需选择 `AudioVideo` 或 `Audio`，即音视频直播间或纯音频直播间。加入房间的角色也分为主播 `LiveBroadcaster` 和观众 `LiveAudience`，下面就这两种身份，分别进行说明。

## 主播端

用户加入房间的角色为主播 `LiveBroadcaster`。

### 步骤 1.1：设置监听

#### 设置本地事件监听

设置加入房间事件回调

```
engine.setOnRoomJoinedListener(({code, message}) => {  
  if (code === 0) {  
    // 加入房间成功  
  } else {  
    // 加入房间失败  
  }  
});
```

设置发布资源事件回调

```
engine.setOnPublishedListener(({type, code, message}) => {  
  // type 发布资源的类型 RCRTCMediaType  
  if (code === 0) {  
    // 发布成功  
  } else {  
    // 发布失败  
  }  
});
```

设置取消发布资源事件回调



```
engine.setOnUnpublishedListener(({type, code, message}) => {
// type 取消发布资源的类型 RCRTCMediaType
if (code === 0) {
// 取消发布成功
} else {
// 取消发布失败
}
});
```

#### 设置订阅资源事件回调

```
engine.setOnSubscribedListener(({userId, type, code, message}) => {
// userId 远端用户 ID
// type 订阅资源的类型 RCRTCMediaType
if (code === 0) {
// 订阅成功
} else {
// 订阅失败
}
});
```

#### 设置取消订阅资源事件回调

```
engine.setOnUnsubscribedListener(({userId, type, code, message}) => {
// userId 远端用户 ID
// type 取消订阅资源的类型 RCRTCMediaType
if (code === 0) {
// 取消订阅成功
} else {
// 取消订阅失败
}
});
```

### 设置远端事件监听

#### 设置远端用户加入房间事件回调

```
engine.setOnUserJoinedListener(({userId, roomId}) => {
// userId 远端用户 ID
// roomId 房间 ID
});
```

#### 设置远端用户离开房间事件回调

```
engine.setOnUserLeftListener(({userId, roomId}) => {
// userId 远端用户 ID
// roomId 房间 ID
});
```

#### 设置远端用户发布资源回调

```
engine.setOnRemotePublishedListener(({userId, roomId, type}) => {  
  // userId 远端用户 ID  
  // roomId 房间 ID  
  // type 远端用户发布的资源类型 RCRTCMediaType  
});
```

设置远端用户取消发布资源回调

```
engine.setOnRemoteUnpublishedListener(({userId, roomId, type}) => {  
  // userId 远端用户 ID  
  // roomId 房间 ID  
  // type 远端用户发布的资源类型 RCRTCMediaType  
});
```

## 步骤 1.2：加入房间

1. 构建 RCRTCRoomSetup，指定主播身份和房间类型：

```
let roomSetup = {  
  // 根据实际场景，选择音视频直播：AudioVideo 或纯音频直播：Audio  
  type: RCRTCMediaType.AudioVideo  
  role: RCRTCRole.LiveBroadcaster  
};
```

2. 调用 RCRTCEngine 下的 joinRoom 方法创建并加入一个直播房间：

```
engine.joinRoom('直播间 ID', roomSetup);
```

### ① 提示

客户端通过 joinRoom 传入的"直播间 ID" 来加入不同房间。房间不需要客户端创建或销毁，融云服务若发现该房间不存在时会自动创建，当没有任何主播时（只有观众不算），过一段时间也会自动销毁该房间。

## 步骤 1.3：发布音视频资源

### 发布音视频资源

加入房间后，开始摄像头采集并发布音视频资源。

```
/// 开启摄像头
engine.enableCamera(true);
/// 发布音视频资源
engine.publish(RCRTCMediaType.AudioVideo);
```

## 设置显示视频的组件

设置用于显示视频的 RCRTCView，调用 RCRTCEngine 下的 setLocalView 方法设置本地视频的显示 view。

### ⚠ 警告

RCRTCView 内部包装的是原生提供的组件，所以 RCRTCView 只能在 nvue 文件中使用。

### 导入预览窗口组件

```
// 导入 RCRTCView
import RCRTCView from '@uni_modules/RongCloud-RTCWrapper/components/RCRTCView';

// 声明 RCRTCView
export default {
  components: {
    RCRTCView,
  },
}
```

### 添加预览窗口

```
<!-- 增加 RCRTCView 组件，fitType: 视频填充模式，mirror: 视频是否镜像显示 -->
<RCRTCView class="localView" ref="localView" :fitType="RCRTCViewFitType.Center" :mirror="true">
</RCRTCView>
```

### 设置预览窗口

```
// 设置预览窗口
engine.setLocalView(this.$refs.localView.getNativeViewRef(), (code) => {
  if (code === 0) {
    // 设置成功
  } else {
    // 设置失败
  }
});
```

## 步骤 1.4：主播订阅房间内其他主播的资源

1. 调用 RCRTCEngine 下的 subscribe 方法订阅房间内其他主播的资源，在主播连麦的场景下会用到该方法，当远端主播发布资源时，会通过 setOnRemotePublishedListener 回调通知，需要订阅音视频资源并显示视图。

```
engine.subscribe(userId, RCRTCMediaType.AudioVideo);
```

2. 创建用于显示视频的 RCRTCView，调用 RCRTCEngine 下的 setRemoteView 方法设置远端视频的显示 view。

```
/// 导入组件 和 添加 remoteView 组件，可参考 setLocalView 部分的示例  
/// 设置预览窗口  
engine.setRemoteView(userId, this.$refs.remoteView.getNativeViewRef(), (code) => {  
  if (code === 0) {  
    // 设置成功  
  } else {  
    // 设置失败  
  }  
});
```

## 观众端

用户加入房间的角色为观众 LiveAudience。

### 步骤 2.1：设置监听

设置合流资源发布监听

```
engine.setOnRemoteLiveMixPublishedListener(({type}) => {  
  // type 资源类型 RCRTCMediaType  
});
```

设置合流资源取消发布监听

```
engine.setOnRemoteLiveMixUnpublishedListener(({type}) => {  
  // type 资源类型 RCRTCMediaType  
});
```

### 步骤 2.2：加入房间

1. 构建 RCRTCRoomSetup，指定观众身份和房间类型：

```
let roomSetup = {  
  // 根据实际场景，选择音视频直播：audio_video 或纯音频直播：audio  
  type: RCRTCMediaType.AudioVideo,  
  role: RCRTCRole.LiveAudience,  
};
```

2. 调用 RCRTCEngine 下的 joinRoom 方法创建并加入一个直播房间：

```
engine.joinRoom('直播间 ID', setup);
```

## 步骤 2.3：观众观看直播

1. 在监听到合流资源发布事件之后，调用 RCRTCEngine 下的 subscribeLiveMix 方法订阅直播。

```
engine.subscribeLiveMix(RCRTCMediaType.AudioVideo);
```

2. 创建用于显示视频的 RCRTCVideoView，调用 RCRTCEngine 下的 setLiveMixView 方法设置房间直播资源的显示 view。

```
/// 创建预览窗口  
/// 导入组件 和 添加 remoteView 组件，可参考 setLocalView 部分的示例  
/// 设置预览窗口  
engine.setLiveMixView(this.$refs.remoteView.getNativeViewRef(), (code) => {  
  if (code === 0) {  
    // 设置成功  
  } else {  
    // 设置失败  
  }  
});
```

# 引擎配置

## 引擎配置引擎配置速览

更新时间:2024-08-30

RTC 引擎提供以下配置，可按需修改。

适用平台	配置项	默认值
全平台	断线重连	默认开启
全平台	状态报表数据回调时间间隔	默认1000ms
全平台	音频初始化配置 - 编码类型	默认 Opus
Android	音频初始化配置 - 录音来源	默认来自语音通信
Android	音频初始化配置 - 采样率	默认 16000
Android	音频初始化配置 - 立体声	默认开启
Android	音频初始化配置 - 麦克风采集	默认开启
全平台	视频初始化配置 - 大小流	默认开启
Android	视频初始化配置 - 硬件编码	默认开启
Android	视频初始化配置 - 硬件解码	默认开启
Android	视频初始化配置 - 高压压缩编码	默认关闭
Android	视频初始化配置 - 硬件编码帧率	默认 30 Fps
Android	视频初始化配置 - 采集/解码 到纹理	默认开启

## 断线重连

断线重连功能默认开启，可以在引擎初始化时传入以下配置进行关闭：

```
let setup = {reconnectable: false};
RCRTCEngine.create(setup);
```

## 状态报表数据回调时间间隔

状态数据报表回调默认时间间隔为1000ms，最小时间间隔为100ms。请注意，过小的时间间隔会影响性能。

可以在引擎初始化时传入以下配置进行修改：

```
/// 修改回调时间间隔为 2 秒
let setup = {statsReportInterval: 2000};
RCRTCEngine.create(setup);
```

## 音频初始化配置

## 修改音频编解码类型

目前支持 PCMU 和 OPUS 两种音频编解码方式，默认配置是 OPUS。

可以在引擎初始化时传入以下配置进行修改：

```
/// 修改音频编解码类型为PCMU
let audioSetup = {codec: RCRTCAudioCodecType.PCMU};
let setup = {audioSetup}
RCRTCEngine.create(setup);
```

## 修改音频录音来源

默认录音来源为语音通信。

可以在引擎初始化时传入以下配置进行修改（仅支持 Android）：

```
/// 修改录音来源为麦克风
let audioSetup = {audioSource: RCRTCAudioSource.MIC};
let setup = {audioSetup}
RCRTCEngine.create(setup);
```

RCRTCAudioSource 枚举值对应 Android SDK 中的 MediaRecorder.AudioSource。

## 修改音频采样率

引擎支持的采样率有：8000，16000，32000，44100，48000。默认为 16000。

可以在引擎初始化时传入以下配置进行修改（仅支持 Android）：

```
/// 修改音频采样率为 32000
let audioSetup = {audioSampleRate: RCRTCAudioSampleRate.Type32000};
let setup = {audioSetup}
RCRTCEngine.create(setup);
```

## 开启/关闭立体声

默认开启，可以在引擎初始化时传入以下配置进行关闭（仅支持 Android）：

```
let audioSetup = {enableStereo: false};
let setup = {audioSetup}
RCRTCEngine.create(setup);
```

## 开启/关闭麦克风采集

默认开启，可以在引擎初始化时传入以下配置进行关闭（仅支持 Android）：

```
let audioSetup = {enableMicrophone: false};
let setup = {audioSetup}
RCRTCEngine.create(setup);
```

### 提示

如配置麦克风关闭，则在整个引擎生命周期内都无法再次开启。

## 视频初始化配置

### 开启/关闭大小流

默认开启，可以在引擎初始化时传入以下配置进行关闭：

```
let videoSetup = {enableTinyStream: false};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

### 开启/关闭硬件编码

默认开启，可以在引擎初始化时传入以下配置进行关闭（仅支持 Android）：

```
let videoSetup = {enableHardwareEncoder: false};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

### 开启/关闭硬件解码

默认开启，可以在引擎初始化时传入以下配置进行关闭（仅支持 Android）：

```
let videoSetup = {enableHardwareDecoder: false};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

### 开启/关闭硬件高压压缩编码

默认关闭，可以在引擎初始化时传入以下配置进行开启（仅支持 Android）：

```
let videoSetup = {enableHardwareEncoderHighProfile: true};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

### 提示



开启硬件高压压缩编码可能会引发兼容性问题，请谨慎使用。

## 配置硬件编码帧率

取值范围 0~30 帧，默认 30 帧。

可以在引擎初始化时传入以下配置进行修改（仅支持 Android）：

```
/// 修改硬件编码帧率为 25 帧
let videoSetup = {hardwareEncoderFrameRate: 25};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

## 开启/关闭 采集/解码 到纹理

默认开启，可以在引擎初始化时传入以下配置进行关闭（仅支持 Android）：

```
let videoSetup = {enableTexture: false};
let setup = {videoSetup}
RCRTCEngine.create(setup);
```

### 提示

关闭后，采集/解码将通过 YUV 数据的形式进行，通常建议 Android 5.0 以下的设备关闭此配置以换来更好的兼容性

## 基本操作

## 基本操作 设置加入房间调用监听

更新时间:2024-08-30

调用加入房间接口之前需要先设置方法回调监听，用来判断是否成功加入房间。

- 示例代码：

```
engine.setOnRoomJoinedListener(({code, message}) => {  
  if (code === 0) {  
    // 创建/加入房间成功  
  } else {  
    // 创建/加入房间失败  
  }  
});
```

## 加入房间

调用 RCRTCEngine 下的 joinRoom 方法加入房间，如果该房间之前不存在，则会在调用时自动创建并加入。

- 参数说明：

参数	类型	说明
roomId	字符串	房间唯一 ID <sup>注1</sup>
setup	RCRTCRoomSetup	加入房间时提供的初始化信息 <sup>注2</sup>

注1：roomId 支持大小写英文字母、数字、部分特殊符号 + = - \_ 的组合方式 最长 64 个字符。

注2：setup 内包含用户身份信息和房间的资源类型属性。

- 示例代码：

以会议成员身份加入房间：

```
let setup = {  
  type: RCRTCMediaType.AudioVideo  
  role: RCRTCRole.MeetingMember // 会议成员  
};  
engine.joinRoom(roomId, setup);
```

以主播身份加入房间：

```
let setup = {
  type: RCRTCMediaType.AudioVideo
  role: RCRTCRole.LiveBroadcaster // 主播
};
engine.joinRoom(roomId, setup);
```

以观众身份加入房间：

```
let setup = {
  type: RCRTCMediaType.AudioVideo,
  role: RCRTCRole.LiveAudience, // 观众
};
engine.joinRoom(roomId, setup);
```

## 设置退出房间调用监听

调用退出房间接口之前建议先设置方法回调监听，用来判断是否成功退出房间。

• 示例代码：

```
engine.setOnRoomLeftListener(({code, message}) => {
  if (code === 0) {
    // 退出成功
  } else {
    // 退出失败
  }
});
```

## 退出房间

调用 RCRTCEngine 下的 leaveRoom 接口离开房间，离开时 SDK 内部会自动取消所有已发布和订阅的资源。

• 示例代码：

```
engine.leaveRoom();
```

## 房间事件回调

## 房间事件回调

更新时间:2024-08-30

开发者可通过设置 RCRTCEngine 中的不同函数回调方法来监听房间内远端用户的状态及资源变化。设置监听时参数为空，则表示取消已设置的监听。

### ⚠ 警告

设置监听与 RCRTCEngine.create() 方法需要在同一页面内。如果需要在其他页面设置监听，可以通过 uni.\$emit('事件名', 参数) 将事件转发出去，其他页面通过 uni.\$on('事件名', 回调方法) 监听

## 状态相关

### 1. 远端用户加入通知：

当有远端用户加入时触发。因用户加入房间后才能发布资源，该回调代表这名用户刚刚加入，此时并无任何资源发布，所以此刻也订阅不到该用户的任何媒体流。

- 参数 `userId` 为远端用户 ID

```
setOnUserJoinedListener(callback?: (data: OnUserJoinedData) => void): void;
```

### 2. 远端用户离开通知：

当有远端用户离开房间时触发，此时 SDK 会自动取消订阅该用户已发布的流，无需手动调用 `unsubscribe`。

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID

```
setOnUserLeftListener(callback?: (data: {userId, roomId}) => void): void;
```

### 3. 远端用户掉线通知：

当有远端用户掉线时触发，代表该用户意外与融云服务断连超过 1 分钟。网络不好、App 意外崩溃或用户主动杀进程等情况，都会造成客户端与融云服务断连。如 1 分钟内没有恢复，则会被服务判定掉线，此时 SDK 会自动取消订阅该用户的所有资源，无需手动调用 `unsubscribe`。

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID

```
setOnUserOfflineListener(callback?: (data: {userId, roomId}) => void): void;
```

#### 4. 远端用户资源状态变更通知：

当远端用户调用了 `muteLocalStream` 方法时触发。

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID
- 参数 `type` 为远端用户变更的资源类型
- 参数 `disabled` 为远端用户更新后的值，`true` 代表资源静默，`false` 代表恢复正常。

```
setOnRemoteStateChangedListener(callback?: (data: {userId, roomId, type, disabled}) => void): void;
```

## 资源相关

#### 1. 远端用户资源发布通知：

当远端用户发布资源时触发。

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID
- 参数 `type` 为远端用户发布的资源类型

```
setOnRemotePublishedListener(callback?: (data: {userId, roomId, type}) => void): void;
```

#### 2. 远端用户资源取消发布通知：

当远端用户取消发布资源时触发，接收到后 SDK 会自动取消订阅相应资源。开发者也可以根据资源类型，来给用户做出相应的提示

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID
- 参数 `type` 为远端用户取消发布的资源类型

```
setOnRemoteUnpublishedListener(callback?: (data: {userId, roomId, type}) => void): void;
```

### 3. 合流资源发布通知（仅限直播模式下观众端使用）：

当 MCU 服务器发布资源时触发。

- 参数 `type` 为发布的资源类型

```
setOnRemoteLiveMixPublishedListener(callback?: (data: {type}) => void): void;
```

### 4. 合流资源取消发布通知（仅限直播模式下观众端使用）：

当 MCU 服务器取消发布资源时触发。

- 参数 `type` 为取消发布的资源类型

```
setOnRemoteLiveMixUnpublishedListener(callback?: (data: {type}) => void): void;
```

## 首帧回调

### 1. 远端用户发布的视频资源首帧回调

- 参数 `userId` 为远端用户 ID
- 参数 `roomId` 为远端用户所在的房间 ID
- 参数 `type` 为资源类型

```
setOnRemoteFirstFrameListener(callback?: (data: {userId, roomId, type}) => void): void;
```

### 2. 合流视频资源首帧回调（仅限直播模式下观众端使用）

- 参数 `type` 为资源类型

```
setOnRemoteLiveMixFirstFrameListener(callback?: (data: {type}) => void): void;
```



## 摄像头

## 摄像头 打开/关闭摄像头

更新时间:2024-08-30

调用 RCRTCEngine 下的 enableCamera 打开/关闭摄像头。默认开启前置摄像头，可以通过传入第二个参数指定开启的摄像头。

### 提示

对于 Android 嵌入式设备或使用外接摄像头的情况，建议设置 setOnCameraEnabledListener 回调方法以确保设备开启成功。

### 示例代码：

#### 设置回调

```
engine.setOnCameraEnabledListener(({enable, code, message}) => {  
  if (code === 0) {  
    // 操作成功  
  } else {  
    // 操作失败  
  }  
});
```

#### 开启摄像头

```
engine.enableCamera(true);
```

#### 开启指定摄像头

```
engine.enableCamera(true, RCRTCcamera.Back);
```

#### 关闭摄像头

```
engine.enableCamera(false);
```



## 切换摄像头

调用 RCRTCEngine 下的 switchCamera 切换前后摄像头。通过设置 setOnSwitchCameraListener 回调方法来监听是否成功切换摄像头。

- 示例代码：

设置回调

```
engine.setOnSwitchCameraListener(({camera, code, message}) => {  
  if (code === 0) {  
    // 切换成功  
  } else {  
    // 切换失败  
  }  
});
```

切换摄像头

```
engine.switchCamera();
```

## 设置视频参数

调用 RCRTCEngine 下的 setVideoConfig 设置视频参数。

- 示例代码：

```
let config = {  
  minBitrate: 500,  
  maxBitrate: 2200,  
  fps: RCRTCVideoFps.Fps24,  
  resolution: RCRTCVideoResolution.Resolution_720x1280,  
  mirror: false,  
};  
engine.setVideoConfig(config);
```

## 手动对焦

调用 RCRTCEngine 下的 isCameraFocusSupported 来判断摄像头是否支持区域对焦。

① 提示

对于支持的设备，可调用 `setCameraFocusPositionInPreview` 进行手动对焦，设置对焦的坐标原点为视频区域的左上角。

- 示例代码：

```
let supported = engine.isCameraFocusSupported();
if (supported) {
  engine.setCameraFocusPositionInPreview(100, 100);
}
```

## 区域测光

调用 `RCRTCEngine` 下的 `isCameraExposurePositionSupported` 来判断摄像头是否支持区域测光。

① 提示

对于支持的设备，可调用 `setCameraExposurePositionInPreview` 进行区域测光设置，设置测光的坐标原点为视频区域的左上角。

- 示例代码：

```
bool supported = engine.isCameraExposurePositionSupported();
if (supported) {
  engine.setCameraExposurePositionInPreview(100, 100);
}
```

## 采集方向

调用 `RCRTCEngine` 下的 `setCameraCaptureOrientation` 来设置摄像头采集角度。

- 示例代码：

```
engine.setCameraCaptureOrientation(RCRTCCameraCaptureOrientation.LandscapeRight);
```

## 麦克风

## 麦克风 打开/关闭

更新时间:2024-08-30

可以通过 RCRTCEngine 下的 enableMicrophone 控制麦克风的打开和关闭。

```
enableMicrophone(enable: boolean): number;
```

- 示例代码：

```
engine.enableMicrophone(true);
```

## 设置声音质量和场景

可以通过 RCRTCEngine 下的 setAudioConfig 设置声音质量和场景。具体设置可参见 [音频模式](#)。

```
setAudioConfig(config: RCRTCAudioConfig): number;
```

- 参数说明：

参数	类型	说明
config	RCRTCAudioConfig	音频配置

- 示例代码：

```
let config = {  
  quality: RCRTCAudioQuality.MusicHigh,  
  scenario: RCRTCAudioScenario.MusicClassRoom,  
};  
engine.setAudioConfig(config);
```

## 设置音量

可以通过 RCRTCEngine 下的 adjustLocalVolume 设置音量大小。声音的大小范围为 0-100。

```
adjustLocalVolume(volume: number): number;
```

- 示例代码：

```
engine.adjustLocalVolume(100);
```

## 扬声器

## 扬声器 扬声器/听筒

更新时间:2024-08-30

可以通过 RCRTCEngine 下的 enableSpeaker 切换使用扬声器/听筒播放声音。

```
enableSpeaker(enable: boolean): number;
```

- 参数说明：

参数	类型	说明
enable	boolean	true 使用扬声器；false 使用听筒

- 示例代码：

```
engine.enableSpeaker(true);
```

## 设备检测

## 设备检测

更新时间:2024-08-30

SDK 从 5.2.5 版本开始支持该功能。

在通话/直播前，用户可以通过设备检测接口，查看自己的设备是否都处在可用状态。

### 麦克风 & 扬声器检测

开发者可以通过调用 RCRTCEngine 对象的 startEchoTest 来测试麦克风和扬声器功能。参数 timeInterval 代表录音时长，取值范围为 [2,10] 秒，调用后需让用户对着麦克风说一段话，等录音结束，如果这段声音能被正常播出，则代表麦克风和扬声器都工作正常。startEchoTest 开始后，必须通过调用 stopEchoTest 来结束测试，否则会对其他流程有影响。

- 示例代码：

```
// 开启回声测试。10 表示麦克风会录制 10 秒后，然后通过扬声器播放本次录到的声音，如可以听到录音则表示麦克风和扬声器功能正常
engine.startEchoTest(10);
// 等待并检查是否可以听到自己的声音回放
// 停止测试
engine.stopEchoTest();
```

## 本地用户资源

## 本地用户资源

更新时间:2024-08-30

用户进入音视频房间后，如果想让其他人看见自己的画面、听见自己的声音，需要发布（Publish）本地资源。如果想要看到别人的画面、听见别人的声音，需要订阅（Subscribe）其他人已发布的资源。

### 发布音视频资源

开发者可在 joinRoom 成功后，通过 RCRTCEngine 中的 publish 方法，发布麦克风和摄像头采集的资源。

- 示例代码：

```
// 发布音视频资源，即麦克风、摄像头采集数据  
engine.publish(RCRTCMediaType.AudioVideo);
```

### 取消发布音视频资源

当需要取消发布时，可调用 RCRTCEngine 中的 unpublish 来取消发布麦克风和摄像头采集的资源。

取消发布接口通常跟发布接口配对使用，但如果是用户想要退出房间，则不需要调用取消发布方法，在调用退出房间接口时，SDK 内部会自动进行取消处理。

- 示例代码：

```
// 取消发布音视频资源，即麦克风、摄像头采集数据  
engine.unpublish(RCRTCMediaType.AudioVideo);
```

## 远端用户资源

## 远端用户资源

更新时间:2024-08-30

用户进入音视频房间后，如果想让其他人看见自己的画面、听见自己的声音，需要发布（Publish）本地资源。如果想要看到别人的画面、听见别人的声音，需要订阅（Subscribe）其他人已发布的资源。

### 订阅

用户的订阅需要在收到远端用户发布资源的通知，即 `setOnRemotePublishedListener` 收到回调后订阅。可调用 `RCRTCEngine` 中的 `subscribe` 来订阅某个远端用户的音视频资源，如果远端用户发布的视频资源开启了大小流功能，可以通过 `subscribe` 的 `tiny` 参数，来选择订阅大流或小流（默认）。

• 参数说明：

参数	类型	说明
<code>userId</code>	<code>String</code>	远端用户 ID
<code>type</code>	<code>RCRTCMediaType</code>	资源类型
<code>tiny</code>	<code>bool</code>	是否订阅小流 <i>非必需</i>

• 示例代码：

```
engine.subscribe(userId, RCRTCMediaType.AudioVideo);
```

### 取消订阅

当需要取消订阅时，可调用 `RCRTCEngine` 中的 `unsubscribe` 来取消订阅某个远端用户的音视频资源。取消订阅接口通常跟订阅接口配对使用，但如果是用户想要退出房间，则不需要调用取消订阅方法，在调用退出房间接口时，SDK 内部会自动进行取消处理。

• 参数说明：

参数	类型	说明
<code>userId</code>	<code>String</code>	远端用户 ID
<code>type</code>	<code>RCRTCMediaType</code>	资源类型

• 示例代码：



```
engine.unsubscribe(userId, RCRTCMediaType.AudioVideo);
```

## 发布与取消

## 发布与取消

更新时间:2024-08-30

用户以主播身份加入房间后，需要发布资源，房间内的其他主播和观众才能看见、听见。

### 发布

直播房间发布音视频流，可调用 RCRTCEngine 中的 `publish` 来发布本地音视频资源。

- 示例代码：

```
// 发布音视频资源，即麦克风、摄像头采集数据  
engine.publish(RCRTCMediaType.AudioVideo);
```

### 取消发布

当需要取消发布直播流时，可调用 RCRTCEngine 中的 `publish` 来取消发布本地音视频资源。

- 示例代码：

```
// 取消发布音视频资源，即麦克风、摄像头采集数据  
engine.unpublish(RCRTCMediaType.AudioVideo);
```

## 订阅与取消

## 订阅与取消

更新时间:2024-08-30

对于仅自己发布，不跟其他人连麦<sup>注</sup>的情况，并不需要执行订阅逻辑。

### 提示

注：连麦是指在当前直播间内，主播邀请观众上麦，观众切换为主播后，主播们之间互相发布订阅的情况。

## 订阅资源

直播房间内有多名主播时，主播之间也需要订阅才能相互看见、听见。需调用 RCRTCEngine 中的 subscribe 来订阅主播资源。

### 参数说明：

参数	类型	说明
userId	String	远端主播 ID
type	RCRTCMediaType	资源类型
tiny	bool	是否订阅小流 <i>非必需</i>

### 示例代码：

```
engine.subscribe(userId, RCRTCMediaType.AudioVideo);
```

## 取消订阅

当需要取消订阅时，可调用 RCRTCEngine 中的 unsubscribe 来取消订阅主播资源。取消订阅接口通常跟订阅接口配对使用，但如果是用户想要退出房间，则不需要调用取消订阅方法，在调用退出房间接口时，SDK 内部会自动进行取消处理。

### 参数说明：

参数	类型	说明
userId	String	远端主播 ID

参数	类型	说明
type	RCRTCMediaType	资源类型

- 示例代码：

```
engine.unsubscribe(userId, RCRTCMediaType.AudioVideo);
```

## 合流布局

## 合流布局 画布配置

更新时间:2024-08-30

1. 画布指的是多道视频流新合成后视频背景宽高，不管哪种合流布局方式，都是基于画布宽高范围内渲染，此配置为可选设置。
2. 画布配置的接口需要在发布视频流成功后，才可以调用。
  - 大流画布宽高默认为 360 \* 640 帧率为 25 码率为 800 。
  - 小流画布宽高默认为 180 \* 320 帧率为 15 码率为 200 。
  - 可通过传递 tiny 参数区分设置大小流。

### 示例代码

```
/// 设置大流画布码率
engine.setLiveMixVideoBitrate(2200);
/// 设置大流画布分辨率
engine.setLiveMixVideoResolution(720, 1280);
/// 设置大流画布帧率
engine.setLiveMixVideoFps(RCRTCVideoFps.Fps15);

/// 设置小流画布码率
engine.setLiveMixVideoBitrate(800, true);
/// 设置小流画布分辨率
engine.setLiveMixVideoResolution(360, 640, true);
/// 设置小流画布帧率
engine.setLiveMixVideoFps(RCRTCVideoFps.Fps15, true);
```

2. 填充方式设置，填充方式共分为两种：

- RCRTCLiveMixRenderMode.Crop
- RCRTCLiveMixRenderMode.Whole(默认)

### 示例代码

```
/// 设置填充方式为裁剪
engine.setLiveMixRenderMode(RCRTCLiveMixRenderMode.Crop);
```

## 音频配置

1. 设置音频码率，音频码率默认 200 。
2. 此接口调用时机可以在主播发布资源后的任意时刻。

### 示例代码

```
engine.setLiveMixAudioBitrate(400);
```

## 合流布局

1. 合流布局的功能主要是定义直播间连麦后合流视频的布局方式，可以根据自身业务需求来定义多个连麦者的画面的布局样式。
2. 接口调用时机可以在主播发布资源后的任意时刻。使用自定义布局时需要在房间有人加入和退出时分别调用，以便合流视频中的画面按照 APP 设计规则展示。

直播合流视频布局目前分为三种：1. 悬浮布局（默认）；2. 自适应布局；3. 自定义布局。下面分别介绍布局效果。

### 悬浮布局

背景视频来源默认采用第一个加入房间的主播或调用合流布局相关接口的主播发布的视频，显示区域为整个合流视频，合流视频大小需要调用 `setLiveMixVideoResolution` 接口设置（默认值是  $360 * 640$ ）；当连麦者依次加入时，按照下图显示的序列加载子视图：



当有人离开时，系统会自动按照现有主播加入房间的次序重新布局视图。

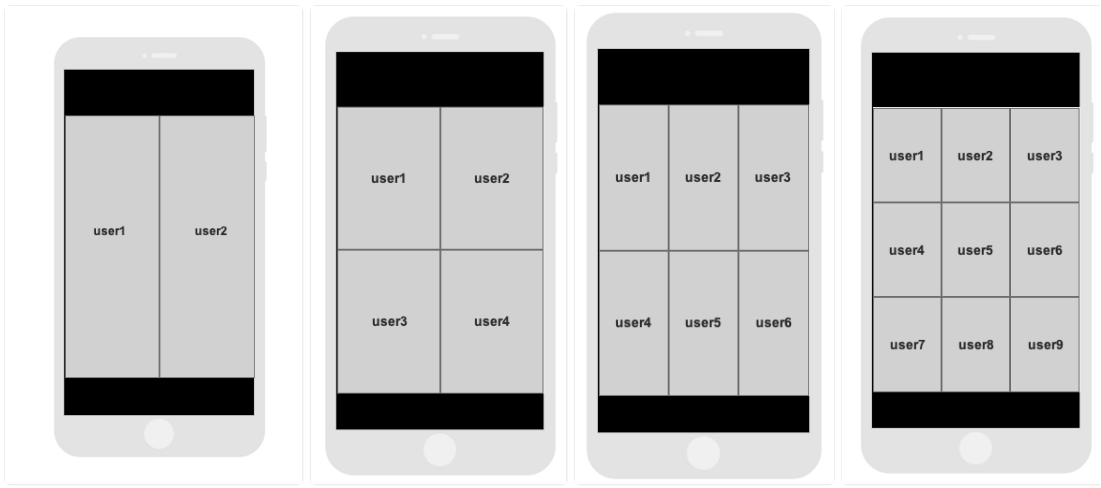
#### 示例代码

```
engine.setLiveMixLayoutMode(RCRTLiveMixLayoutMode.Suspension);
```

### 自适应布局

视频的整体大小为默认值  $360 * 640$  或通过合流接口自定义；当有多人加入房间后，直播系统会按照具体人数平分整体视频区域，使之每个子视图加载区域大小一致；录制系统会按照参会者进入房间的次序依次把参会者图像加载在示意图的相应序号上。

当有人离开时，系统会自动按照现有主播加入房间的次序重新布局视图。



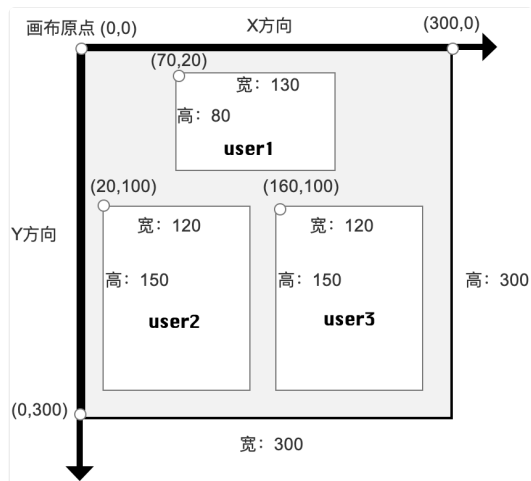
### 示例代码

```
engine.setLiveMixLayoutMode(RCRTCLiveMixLayoutMode.Adaptive);
```

## 自定义布局

通过调用自定义布局接口可以设置合流视频整体尺寸，以及各个连麦者视图位置及大小。

如下图所示，合流布局是以像素方式，定义视频输出尺寸，如图整体视频尺寸 宽\*高 = 300 \* 300 ；以整体作为画布，画布的原点 (0,0) 在左上角, 那么三个连麦主播的窗口相对原点的位置，及其宽度，高度值分别如图所示（相应的设置代码也可在直播布局的接口文档中查看布局实例代码）：



### 示例代码

```
let layouts = [];  
  
let layout1 = {  
  userId: userId1,  
  x: 70,  
  y: 20,  
  width: 130,  
  height: 80,  
};  
layouts.add(layout1);  
  
let layout2 = {  
  userId: userId2,  
  x: 20,  
  y: 100,  
  width: 120,  
  height: 150,  
};  
layouts.add(layout2);  
  
let layout3 = {  
  userId: userId3,  
  x: 160,  
  y: 100,  
  width: 120,  
  height: 150,  
};  
layouts.add(layout3);  
  
engine.setLiveMixCustomLayouts(layouts);
```

\*设置自定义布局后会默认修改 LiveMixLayoutMode 属性为 RCRTCLiveMixLayoutMode.Custom \*

## 音频合流

1. 音频合流布局的原理就是通过控制输入资源列表，设置输出音频配置，达到合并哪些主播音频资源输出到指定配置的音频资源。设置音频资源合流列表后，观众订阅直播资源，就会只听到音频合流列表中对应的主播。
2. 此接口调用时机可以在主播发布资源后的任意时刻。

### 示例代码

```
let userIds = [  
  userId1,  
  userId2,  
  userId3,  
];  
  
engine.setLiveMixCustomAudios(userIds);
```



## 订阅与取消

## 订阅与取消

更新时间:2024-08-30

直播模式中主播发布的音视频资源，会在服务端另行合并生成一道音频合流和一道视频合流。

观众既可以直接订阅原始音视频资源（简称 分流）也可订阅 合流。下面将分别介绍这两种订阅方式的区别和适用场景。

### 提示

观众被定义为只能订阅不能发布，如需发布必须先转为主播身份，再进行资源的发布。

## 订阅分流

分流订阅跟会议模式的订阅一样，适合小众玩法灵活的直播业务场景。比如观众被分成多种角色，需要选择性观看或收听部分主播发布的资源；又或者不同观众的展示布局并不相同，甚至可以随时切换布局的情况，SDK 提供视频展示 View 的创建和与流绑定接口，开发者自行编写展示逻辑。

观众的订阅需要在收到主播刚刚发布资源的通知，即 `setOnRemotePublishedListener` 回调后订阅。可调用 `RCRTCEngine` 中的 `subscribe` 来订阅某个主播的音视频资源，如果主播发布的视频资源开启了大小流功能，可以通过 `subscribe` 的 `tiny` 参数，来选择订阅大流或小流（默认）。

- 示例代码：

```
engine.subscribe(userId, RCRTCMediaType.AudioVideo);
```

## 取消订阅分流

当需要取消订阅分流时，可调用 `RCRTCEngine` 中的 `unsubscribe` 方法来取消订阅指定的主播资源。取消订阅接口通常跟订阅接口配对使用，但如果是用户想要退出房间，则不需要调用取消订阅方法，在调用退出房间接口时，SDK 内部会自动进行取消处理。

- 示例代码：

```
engine.unsubscribe(userId, RCRTCMediaType.AudioVideo);
```

## 订阅合流

大部分直播场景，观众数量比较庞大，且所有观众看到和听到的内容完全一致。当有多个主播时，观众采用分流订阅的方式会造成大量带宽资源的浪费，客户成本也会较高。合流就是用来解决资源浪费问题的，它是由融云服务将所有主播发布的音视频资源，按客户提前指定的人选、布局方式和编码参数进行合并生成的一道音频和一道视频合流（纯音频模式仅有一道音频合流）。观众都去订阅这两道合流，则不管有多少名主播连麦/PK，观众都能用很少的资源得到一致的观看效果，从而为客户节省大量成本。下面介绍订阅合流的步骤：

#### 1. 监听合流发布回调：

- 示例代码：

```
engine.setOnRemoteLiveMixPublishedListener((type) => {  
  // type 媒体类型  
});
```

#### 2. 订阅合流：

监听到合流资源发布事件之后，调用 RCRTCEngine 的 subscribeLiveMix 即可订阅指定的合流资源。合流中的视频流是区分大小流的，可以通过 subscribeLiveMix 方法中的 tiny 参数，来控制订阅大流或小流（默认）。

- 示例代码：

```
engine.subscribeLiveMix(RCRTCMediaType.AudioVideo);
```

## 取消订阅合流

当需要取消订阅合流时，可调用 RCRTCEngine 中的 unsubscribeLiveMix 方法来取消订阅指定的合流资源。取消订阅接口通常跟订阅接口配对使用，但如果是用户想要退出房间，则不需要调用取消订阅方法，在调用退出房间接口时，SDK 内部会自动进行取消处理。

- 示例代码：

```
engine.unsubscribeLiveMix(RCRTCMediaType.AudioVideo);
```

## 资源类型

## 资源类型

更新时间:2024-08-30

### RCRTCMediaType

资源类型	说明
Audio	音频资源
Video	视频资源
AudioVideo	音视频资源

## 同房间连麦

## 同房间连麦 观众上麦

更新时间:2024-08-30

观众上麦本质上是切换身份变成当前房间的主播，然后以主播身份继续直播相关操作。

### 切换为主播

当观众需要上麦，跟房间内的其他主播互动时，调用 `switchLiveRole` 方法，参数传 `RCRTCRole.LiveBroadcaster` 切换身份为主播。

```
// 切换角色的回调
engine.setOnLiveRoleSwitchedListener(({role,code,errMsg}) => {
  if (code == 0) {
    // 切换成功
  } else {
    // 切换失败
  }
});

// 调用切换角色
engine.switchLiveRole(RCRTCRole.LiveBroadcaster);
```

### 观众下麦

主播下麦本质上是切换身份变成当前房间的观众，然后以观众身份继续观看直播等相关操作。

### 切换为观众

当身份为主播时，调用 `switchLiveRole` 方法，参数传 `RCRTCRole.LiveAudience` 切换身份为观众。

#### 提示

如果当前主播用户已通过跨房间连麦加入了其他房间，仍可成功切换为观众身份。此时 SDK 内部会帮其退出所有副房间，但不会结束本次连麦。如果需要结束连麦，请在切换角色前调用 `leaveSubRoom` 方法 `disband` 参数传 `true` 即可。

```
// 切换角色的回调
engine.setOnLiveRoleSwitchedListener(({role,code,errMsg}) => {
if (code == 0) {
// 切换成功
} else {
// 切换失败
}
});
// 调用切换角色
engine.switchLiveRole(RCRTRole.LiveAudience);
```

## 订阅资源

直播用户切换身份后，默认会退订所有资源，请按需重新订阅。

- 观众切换为主播成功后，可以用切换身份之前保存的远端发布资源信息来订阅远端资源。参见[主播订阅资源](#)。
- 主播切换为观众成功后，可以在远端发布合流资源的回调中订阅合流，远端主播发布合流资源的回调会在切换成功后被触发。也可以用之前保存的远端发布资源信息订阅远端主播的分流，详见以下文档：
  - [观众订阅分流](#)
  - [观众订阅合流](#)

## 房间内事件

当房间内的用户使用 `switchLiveRole` 方法上下麦时，同房间内的其他主播可以通过 `setOnRemoteLiveRoleSwitchedListener` 方法监听到远端用户切换身份。

如果用户订阅了一个主播的音视频流，当这个主播切为观众身份时，SDK 内部会主动取消订阅不存在的音视频流，您可以在此回调方法中，更新当前页面视图等操作。

```
// 远端用户切换角色的回调
engine.setOnRemoteLiveRoleSwitchedListener(({roomId,userId,role}) => {

});
```

## 跨房间连麦

## 跨房间连麦

更新时间:2024-08-30

跨房间连麦中需要区分主房间和副房间概念。主副房间是相对概念，定义如下：

- 主房间：本端在最开始加入的房间。
- 副房间：在连麦邀请被接受后，双方均需要加入对方房间。

跨房间连麦前需要双方都已经加入自己的主房间，未加入主房间前无法进行连麦的邀请和被邀请。

### 处理连麦邀请

在建立连麦前，需要由一位主播发出邀请，另一位主播作出同意或拒绝连麦的应答。

### 发起邀请

向指定用户发送跨房间连麦请求，调用 RCRTCEngine 对象的 [requestJoinSubRoom](#) 方法。请求发送成功后，被邀请人会通过 `setOnJoinSubRoomRequestReceivedListener` 回调收到通知。

```
requestJoinSubRoom(roomId: string, userId: string, autoLayout: boolean, extra?: string): number;
```

参数	类型	描述
roomId	String	对方的房间 ID
userId	String	对方用户 ID
autoLayout	bool	是否采用悬浮布局。如为 true，服务端会在加入邀请方房间成功后，把受邀方的流资源合并到邀请方视图上（默认仅悬浮布局合流）。如受邀方未发布资源，则会在受邀方发布资源后进行视图合并。无论 autoLayout 为 true 或 false，双方都可以使用 <code>setLiveMixCustomLayouts</code> 方法主动设置合流布局。一旦主动设置过合流布局，后续音视频直播过程中设置的自动合流参数将失效。默认值：true。
extra	String	附加信息

```
// 邀请其他用户跨房间连麦的回调
engine.setOnJoinSubRoomRequestedListener((res) => {
  if (code == 0) {
    // 邀请其他用户跨房间连麦成功，等待对方处理
  } else {
    // 邀请其他用户跨房间连麦失败
  }
});

// 邀请其他用户跨房间连麦
engine.requestJoinSubRoom(roomId, userId);
```

## 应答邀请

被邀请人端收到邀请后，SDK 会触发 RCRTCEngine 对象的 `setOnJoinSubRoomRequestReceivedListener` 回调方法。接收到邀请通知后，需要进行应答，可以同意连麦或拒绝连麦。

- 如果同意连麦，需调用 `responseJoinSubRoomRequest` 方法并将 `agree` 参数设置为 `true`，发送同意连麦的应答。后续需要调用 `joinSubRoom` 加入副房间。
- 如果拒绝连麦，需调用 `responseJoinSubRoomRequest` 方法并将 `agree` 参数设置为 `false`。

```
responseJoinSubRoomRequest(roomId: string, userId: string, agree: boolean, autoLayout: boolean, extra?: string): number;
```

参数	类型	描述
roomId	String	对方的房间 ID
userId	String	对方用户 ID
autoLayout	bool	是否采用悬浮布局。如为 <code>true</code> ，服务端会在加入邀请方房间成功后，把受邀方的流资源合并到邀请方视图上（默认仅悬浮布局合流）。如受邀方未发布资源，则会在受邀方发布资源后进行视图合并。无论 <code>autoLayout</code> 为 <code>true</code> 或 <code>false</code> ，双方都可以使用 <code>setLiveMixCustomLayouts</code> 方法主动设置合流布局。一旦主动设置过合流布局，后续音视频直播过程中设置的自动合流参数将失效。默认值： <code>true</code> 。
extra	String	附加信息

对连麦邀请作出应答后，本端会通过 `setOnJoinSubRoomRequestRespondedListener` 回调方法收到来自融云服务端的 通知，此时可认为应答成功。同时，邀请方会收到 `setOnJoinSubRoomRequestResponseReceivedListener`，此时表示已接收到对方对连麦邀请作出的回复。

### 提示

- 如果同意连麦，两个房间内的所有非观众用户（被邀请方除外）都会收到 `setOnJoinSubRoomRequestResponseReceivedListener` 应答回调。
- 如果拒绝连麦，只有邀请方会收到应答拒绝的回调 `setOnJoinSubRoomRequestResponseReceivedListener`，其他所有用户都不会收到该应答回调。

```
// 同意其他用户的跨房间连麦邀请后触发的回调
engine.setOnJoinSubRoomRequestRespondedListener(({roomId,userId,agree,code,errMsg}) => {
if (code == 0) {
// 同意响应成功
} else {
// 同意响应失败
}
});
// 收到其他用户发起的跨房间连麦邀请
engine.setOnJoinSubRoomRequestReceivedListener(({roomId,userId,extra}) => {
// 同意其他用户的跨房间连麦邀请
engine.responseJoinSubRoomRequest(roomId, userId, true);
});
```

## 取消邀请

您的 App 用户可能在发起连麦邀请后改变主意，想要取消邀请，此时可调用 RCRTCEngine 对象的 [cancelJoinSubRoomRequest](#) 方法来取消正在进行的跨房间连麦邀请。取消邀请时，被邀请人会收到 `setOnCancelJoinSubRoomRequestReceivedListener` 回调。

```
cancelJoinSubRoomRequest(roomId: string, userId: string, extra?: string): number;
```

参数	类型	描述
roomId	String	对方的房间 ID
userId	String	对方用户 ID

```
// 取消之间发起的跨房间连麦邀请的回调
engine.setOnJoinSubRoomRequestCanceledListener(({roomId,userId,code,errMsg}) => {
if (code == 0) {
// 取消成功
} else {
// 取消失败
}
});

// 取消之间发起的跨房间连麦邀请
engine.cancelJoinSubRoomRequest(roomId, userId);
```

## 加入副房间流程

在连麦邀请被接受后，双方均需要加入对方房间。相对在本端已加入直播房间主播用户来说，对方房间为副房间。加入副房间方法如下：

```
joinSubRoom(roomId: string): number;
```

邀请方与受邀方加入副房间的时机如下：

- 受邀方：在作出同意连麦的应答成功后，调用 `RCRTCEngine` 对象的 [joinSubRoom](#) 方法加入副房间。
- 邀请方：在收到同意连麦的应达后（`setOnJoinSubRoomRequestResponseReceivedListener`），调用 `RCRTCEngine` 对象的 `joinSubRoom` 方法加入副房间。此时会触发 `setOnSubRoomJoinedListener`、`setOnSubRoomBandedListener`、`setOnUserJoinedListener` 回调。



```

// 加入副房间结果的回调
engine.setOnSubRoomJoinedListener(({roomId,code,errMsg}) => {
if (code == 0) {
// 加入副房间成功
} else {
// 加入副房间失败
}
});

// 加入副房间
engine.joinSubRoom(roomId);

```

在超过两位主播连麦时，可能还会有以下两种场景：

- 场景 A：在当前用户加入主房间前，该主房间已有主播用户与其他房间另一用户成功建立跨房间连麦。当前用户可在加入主房间时收到 onSubRoomBanded 回调，用户可在此回调中选择是否加入其他副房间。详细流程如下：

1. 调用 [RCRTCEngine](#) 中 joinRoom 方法加入主房间。
2. setOnSubRoomBandedListener 事件回调方法被触发。

```

/**
 * 主房间中主播已经加入的副房间的回调
 */
setOnSubRoomBandedListener(callback?: (result: OnSubRoomBandedResult) => void): void;

```

3. 调用 RCRTCEngine 对象的 joinSubRoom 方法加入副房间。

- 场景 B：在当前用户加入主房间后，主房间内的另一主播用户发起了跨房间连麦并加入了其他房间，此时当前用户会收到 setOnSubRoomBandedListener 回调，用户可在此回调中选择是否加入其他副房间。

## 订阅资源

在加入副房间成功后，需要订阅副房间主播的资源。有两种方式：

- 加入副房间后，通过 [RCRTCEngine](#) 对象的 subscribe 方法订阅副房间中远端用户已经发布的音视频资源。参考 [订阅资源](#)。

```

engine.setOnSubscribedListener(({id,type,code,message}) => {
// 订阅远端用户的资源的回调
if (code == 0) {
// 订阅成功
} else {
// 订阅失败
}
});

// 订阅远端用户的资源
engine.subscribe(userId, RCRTCMediaType.Video);

```

- 副房间主播发布资源时，setOnRemotePublishedListener 方法会被触发，可通过 roomId 判断是否为副房间。在该回调中调用 subscribe 方法订阅副房间中远端用户刚发布的音视频资源。

```
// 远端用户发布资源的回调
engine.setOnRemotePublishedListener(({roomId,userId,type}) => {
// 订阅远端用户发布的资源
engine.subscribe(userId, type);
});
```

如需取消订阅资源，可调用 unsubscribe 方法取消订阅的资源，可参考 [取消订阅资源](#)。

## 暂时中止连麦

当需要暂时中止连麦时，可离开副房间。调用 RCRTCEngine 对象的 [leaveSubRoom](#) 方法暂时离开副房间，

```
leaveSubRoom(roomId: string, disband: boolean): number;
```

disband 指定是否要结束连麦，在暂时中止连麦的场景下必须设置为 false，仅代表调用方离开副房间，但不结束连麦。成功离开副房间后，调用者会收到 setOnUserLeftListener 回调。其他用户会收到 onUserLeft 回调，对端主播可在该回调中取消订阅离开的主播。

```
// 离开副房间的回调
engine.setOnUserLeftListener(({roomId,code,message}) => {
if (code == 0) {
// 离开成功
} else {
// 离开失败
}
});

// 调用离开副房间，disband:boolean 离开副房间时是否结束连麦。
engine.leaveSubRoom(roomId, disband);
```

暂时中止连麦后，融云服务会自动在合流中移除该主播资源。对端主播房间中订阅合流的观众拉流地址不变，可以看到本房间主播，但无法看到暂时离开的主播。需要恢复连麦时，当前主播可再次加入副房间，无需再执行邀请应答流程。

## 结束连麦

当连麦结束时，需要调用 RCRTCEngine 对象的 [leaveSubRoom](#) 方法离开副房间，同时指定连麦结束。

```
leaveSubRoom(roomId: string, disband: boolean): number;
```

disband 指定是否要结束连麦，在此场景下必须设置为 true。此方法内部会取消订阅已经订阅的副房间所有用户音视频资源，上层无需自行取消订阅。成功结束连麦后，调用者会收到 setOnSubRoomLeftListener 回调，其他用户会收到 setOnSubRoomDisbandListener 回调。

```
// 调用离开副房间的回调
setOnSubRoomLeftListener(callback?: (result: OnSubRoomLeftResult) => void): void;

// 收到结束跨房间连麦的回调
setOnSubRoomDisbandListener(callback?: (result: OnSubRoomDisbandResult) => void): void;
```

暂时中止连麦后，融云服务会自动在合流中移除该主播资源。对端主播房间中订阅合流的观众拉流地址不变，可以看到本房间主播，但无法看到暂时离开的主播。

## 被服务端踢出房间流程

### 提示

通过调用服务端 API 可以将指定房间的指定用户踢出该房间。

- 在踢出某个用户之前，建议先通过服务下发 IM 消息让此客户端执行结束跨房间连麦的流程，然后再从主房间踢出连禁用户。
- 如果将当前用户踢出主房间，该用户会收到 setOnKickedListener 回调，并且 RTC SDK 内部会调用 leaveRoom 方法先退出已经加入的副房间，再退出主房间并销毁资源。UI 层无需再次调用离开主、副房间的方法，只处理上层逻辑即可。内部自动调用 leaveRoom 后，在此房间的其他用户会收到 onUserLeft 回调。
- 如果将当前用户踢出跨房间连麦的副房间，该用户会收到 setOnKickedListener 回调，并且仅退出指定的副房间，并取消订阅该副房间中的音视频资源，UI 层无需再次调用离开副房间的方法，只处理上层逻辑即可。

```
/**
 * 被服务端踢下线，可通过 roomId 判断是否为主房间或副房间
 * 如果用户在房间内，此时收到服务器踢出房间的通知，SDK 会关闭音视频连接，释放资源，
 * 将用户踢出房间，回调通知用户被踢出房间
 *
 */
setOnKickedListener(callback?: (result: OnKickedResult) => void): void;
```

# 融云 CDN

# 融云 CDN

更新时间:2024-08-30

SDK 从 5.2.5 版本开始支持该功能。

当您使用 RTCLib SDK 做直播应用时，可以选用融云 CDN 服务分发直播媒体流。融云 CDN 服务能确保在不同区域、不同场景下加速直播内容的分发，提高资源访问速度。（注：下文中出现的内置 CDN 指的就是 融云 CDN）

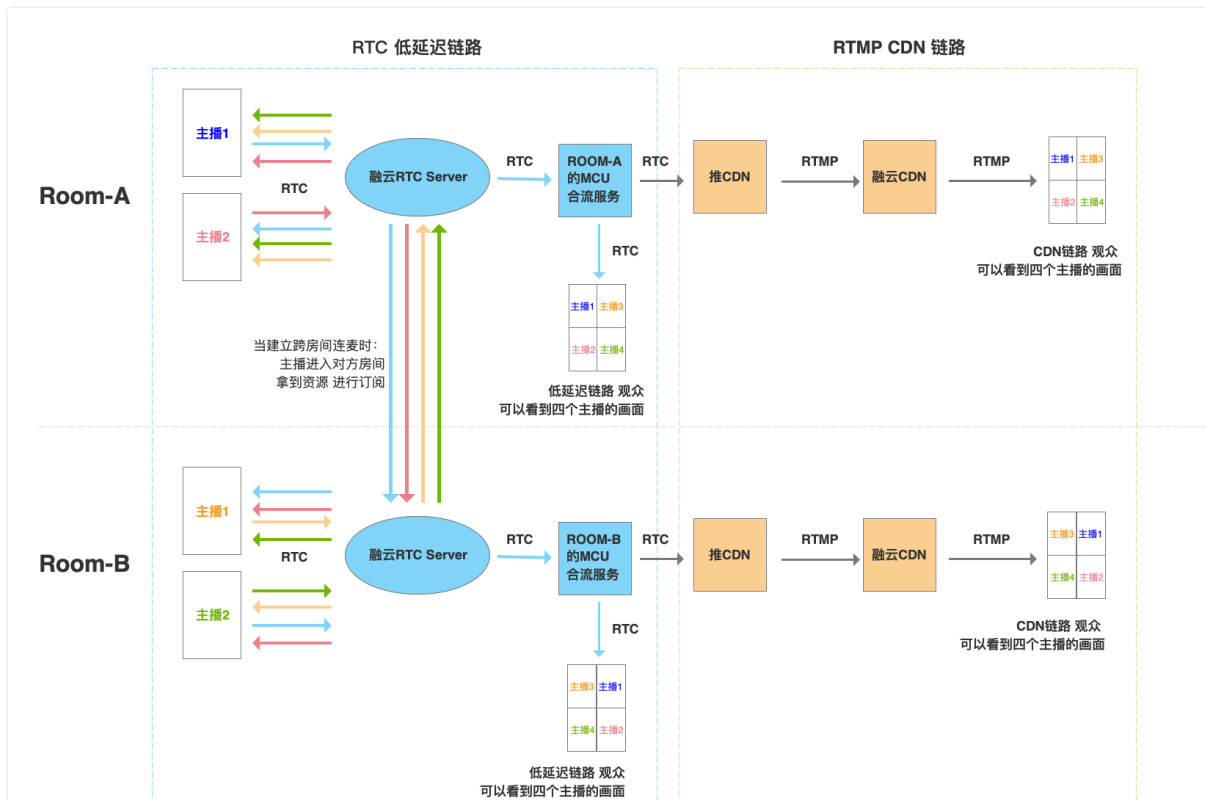
## 融云 CDN 服务架构

主播端用 RTC 协议发布音视频资源，资源由融云服务端接收，并转协议到 RTMP 并推给 CDN。视频编码协议为 h.264。

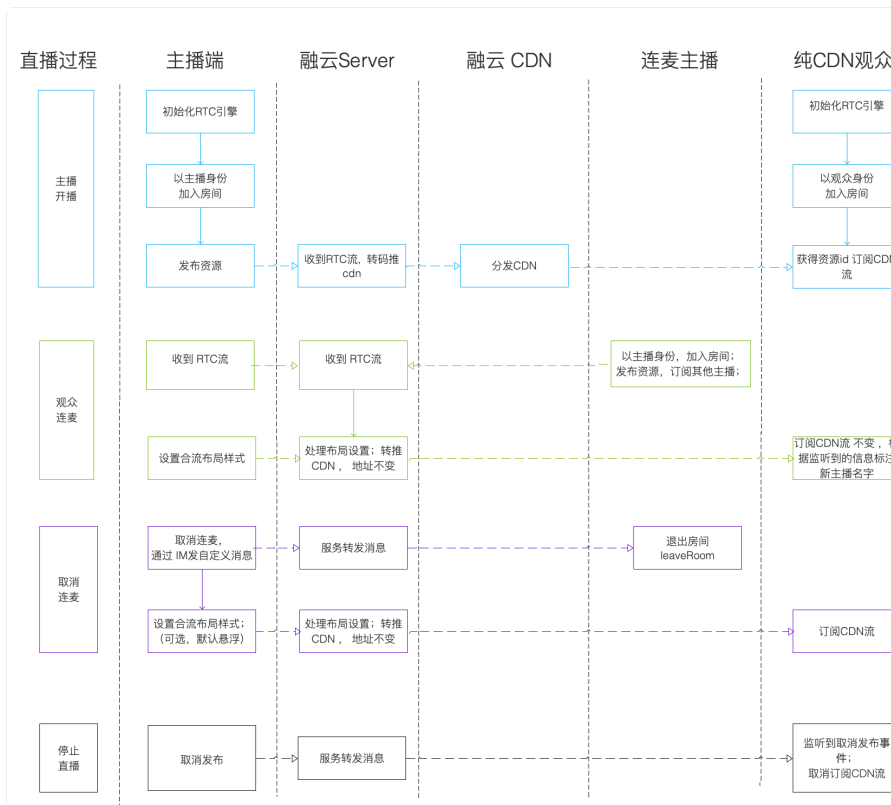
观众端可以感知到主播推流、服务端合流状态变化，无需 App Server 做过多的任务管理。观众端 App 可以调接口来选择订阅 CDN 链路或者 RTC 低延迟链路的直播流。

CDN 拉流时，默认按 CDN 服务输出的视频格式拉流，也可以调接口设置指定的拉流分辨率、码率、帧率（说明：会触发 CDN 转码服务）。

服务架构图：



主要业务流程图：



## 服务开通与配置

融云 CDN 是付费增值服务，且需要开通后才能使用。详见[融云 CDN 服务配置](#)。

## 启用内置 CDN

### 提示

该步骤仅适用于控制台配置的 CDN 推流方式为手动的场景。如果您在控制台配置的 CDN 推流方式为自动，所有直播房间的直播流都会自动推流到 CDN，观众可随时订阅，无需使用此接口启用内置 CDN。

如果控制台配置的 CDN 推流方式为手动，可在直播主播开始推流后，根据产品设计自定决定何时让融云服务开始或停止推流到 CDN。

如需启用内置 CDN，请在主播加入房间成功，并且发布资源成功后，使用 RCRTCEngine 对象的 enableLiveMixInnerCdnStream 方法进行设置。该方法在整个直播过程中都可以操作开关。

```
enableLiveMixInnerCdnStream(enable: boolean): number;
```

参数	类型	说明
enable	bool	是否开启

该方法在整个直播过程中都可以操作开关。开启或关闭内置 CDN 时，观众均会收到对应回调。

- 开启内置 CDN 时：

房间中的观众会收到 `setOnRemoteLiveMixInnerCdnStreamPublishedListener` 回调。

- 关闭内置 CDN 时：

房间中观众会收到 `setOnRemoteLiveMixInnerCdnStreamUnpublishedListener` 回调。

代码示例：

```
engine.setOnLiveMixInnerCdnStreamEnabledListener((res) => {  
  // 启用内置 CDN 的回调  
});  
//启用内置 CDN  
engine.enableLiveMixInnerCdnStream(enable);
```

## 获取房间内 CDN 资源

观众可在加入房间之前设置 `setOnRemoteLiveMixInnerCdnStreamPublishedListener` 回调，用于获取房间内 CDN 资源发布的事件。设置 `setOnRemoteLiveMixInnerCdnStreamUnpublishedListener` 回调，用于获取房间内 CDN 资源取消发布的事件。

```
engine.setOnRemoteLiveMixInnerCdnStreamPublishedListener((res) => {  
  //房间内 CDN 资源发布的回调  
});  
  
engine.setOnRemoteLiveMixInnerCdnStreamUnpublishedListener((res) => {  
  //房间内 CDN 资源取消发布的回调  
});
```

## 订阅 CDN 资源

观众可以调用 `RCRTCEngine` 对象的 `subscribeLiveMixInnerCdnStream` 方法订阅房间中的融云 CDN 资源。默认订阅 CDN 流的原始分辨率、帧率。

```
subscribeLiveMixInnerCdnStream(): number;
```

订阅 CDN 资源的步骤如下：

1. 创建一个 `RCRTCView`，并设置为视频流的预览窗口。

```
// 创建预览窗口  
engine.setLiveMixInnerCdnStreamView(this.$refs.cdnView.getNativeViewRef(), (code) => {  
  if (code === 0) {  
    // 设置成功  
  }  
});
```

2. (可选) 在调用订阅方法前, 可以调用以下方法设置本次订阅的分辨率与帧率。订阅成功后, 也可以使用以下方法切换分辨率与帧率。

- `setLocalLiveMixInnerCdnVideoResolution` : 设置分辨率
- `setLocalLiveMixInnerCdnVideoFps` : 设置帧率

可通过 `setOnLocalLiveMixInnerCdnVideoResolutionSetListener` 和 `setOnLocalLiveMixInnerCdnVideoFpsSetListener` 回调获取分辨率和帧率是否设置成功。

3. 订阅融云 CDN 资源。如未主动设置本地订阅的分辨率或帧率, 默认使用 CDN 流的原始分辨率、帧率。

```
//订阅融云 CDN 资源的回调
engine.setOnLiveMixInnerCdnStreamSubscribedListener((res) => {

});
//订阅融云 CDN 资源
engine.subscribeLiveMixInnerCdnStream();
```

## 取消订阅

观众可以调用 `RCRTCEngine` 对象的 `unsubscribeLiveMixInnerCdnStream` 方法取消订阅融云 CDN 资源。

```
unsubscribeLiveMixInnerCdnStream(): number;
```

示例代码：

```
//取消订阅融云 CDN 的回调
engine.setOnLiveMixInnerCdnStreamUnsubscribedListener((res) => {

});
//取消订阅融云 CDN
engine.unsubscribeLiveMixInnerCdnStream();
```

## 动态切换分辨率和帧率

观众订阅融云 CDN 资源成功后, 调用 `RCRTCEngine` 对象的 `setLocalLiveMixInnerCdnVideoResolution` 方法切换需要观看的分辨率, 调用 `setLocalLiveMixInnerCdnVideoFps` 方法切换需要观看的帧率。

```
/**
 * 下面两个方法有两个不同的调用时机如下：
 * 1. 当没有调用 subscribeLiveMixInnerCdnStream 订阅融云 CDN 流时, 订阅前指定观看的分辨率、帧率。
 * 2. 已经调用 subscribeLiveMixInnerCdnStream 订阅融云 CDN 流后, 切换已经订阅的融云 CDN 视频流的分辨率、帧率。
 */
setLocalLiveMixInnerCdnVideoResolution(width: number, height: number): number;
FsetLocalLiveMixInnerCdnVideoFps(fps: RCRTCVideoFps): number;
```

代码示例：

```
//设置观看 CDN 视频流的分辨率  
engine.setLocalLiveMixInnerCdnVideoResolution(720, 1280);  
  
//设置观看 CDN 视频流的帧率  
engine.setLocalLiveMixInnerCdnVideoFps(RCRTCVideoFps.fps_15);
```

## 设置 CDN 流禁用状态

观众订阅融云 CDN 流成功后，调用 RCRTCEngine 对象的 muteLiveMixInnerCdnStream 方法禁用或启用 CDN 流渲染。

```
muteLiveMixInnerCdnStream(mute: boolean): number;
```

参数	类型	说明
mute	bool	true：停止资源渲染；false：恢复资源渲染

代码示例：

```
engine.muteLiveMixInnerCdnStream(true);
```



## 融云 CDN 服务配置

## 融云 CDN 服务配置

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/appkey/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。

The screenshot shows the 'CDN 服务配置' (CDN Service Configuration) page. At the top, it indicates '融云 CDN 服务状态: 已开通' (Rongyun CDN Service Status: Enabled) with a '关闭服务' (Close Service) button. Below this, there's a '推拉流模式' (Push/Pull Mode) dropdown set to '自行生成推拉流地址' (Generate push/pull stream addresses myself) and a '保存修改' (Save Changes) button. The '域名配置' (Domain Configuration) tab is active, with sub-tabs for '证书管理' (Certificate Management) and '防盗链' (Anti-leech). Under '域名配置', there are two rows for domain settings. The first row is for the '推流域名' (Push Stream Domain), which is '已生效' (Effective), with a 'cname 域名' (CNAME Domain) and a '复制' (Copy) button. The second row is for the '拉流域名' (Pull Stream Domain), also '已生效', with its own 'cname 域名' and '复制' button. A red note states: '注意: CNAME 域名不能直接访问, 您需要在域名服务提供商处完成 CNAME 配置。配置生效后, 即可使用 CDN 链路进行直播拉流' (Note: CNAME domains cannot be accessed directly; you need to complete CNAME configuration at the domain service provider. After configuration is effective, you can use the CDN link for live streaming). Below the note, it says: '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' (Configurations made between 20:00 and 02:00 the next day will not take effect immediately, but will take effect sequentially after 2:00. Configurations made outside this time period will take effect 30 minutes later). There is a '禁用' (Disable) button. The 'HTTPS 设置' (HTTPS Settings) section has a tip: '提示: 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先上传证书' (Tip: Live streams using the FLV/HLS protocol default to HTTP. After configuring an SSL certificate, you can use HTTPS. If you haven't uploaded a certificate to the Rongyun platform, please upload one first). There is a '新增证书' (Add Certificate) button. At the bottom, there are two dropdown menus for '推流' (Push) and '拉流' (Pull), both set to '请选择' (Please select), with '绑定证书' (Bind Certificate) buttons next to them.

## 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。

证书管理 / 新增 返回

提示:

- 1、不支持新增 MD5 加密签名算法证书。由于此类证书安全性较低,且部分主流浏览器已限制使用。
- 2、超过 44 KB 的 crt 文件在 IE11 浏览器上使用存在高风险,请确保您上传的每个 crt 文件大小都小于 44 KB。

\*证书名称:

请输入证书内容

\*证书内容:

请输入私钥内容

\*私钥内容:

请输入备注

备注:

## 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。

域名配置 证书管理 **防盗链**

防盗链配置

推流: KEY:  时长:  秒

拉流: KEY:  时长:  秒

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

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") = 79aead3bd7b5db4adefb93a010298b5`
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=79aead3bd7b5db4adefb93a010298b5&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。

## 第三方 CDN

## 第三方 CDN

更新时间:2024-08-30

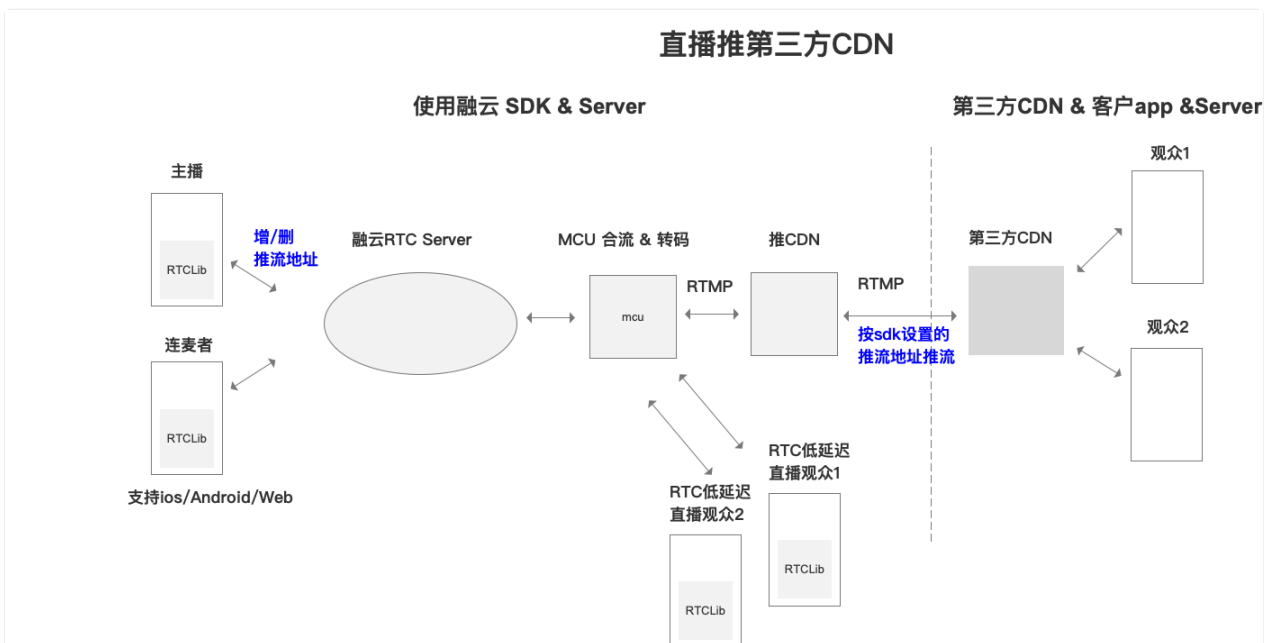
融云支持在低延迟直播中将音视频流旁路出来，转推到任意第三方 CDN 服务。

转推第三方 CDN 服务有两种控制方式，本文仅介绍方案一：

- 方案一：使用客户端接口进行控制，使用 `addLiveCdn` 接口配置 CDN 地址。
- 方案二：使用服务端 API 的 `/rtc/mcu/config` 接口进行控制。本文不做介绍，具体请参见服务端文档[转推第三方 CDN](#)。

## 业务链路

融云直播支持转推第三方 CDN 服务，业务链路如下图所示：



## 配置直播 CDN 地址

开发者首先需要以主播身份加入房间，然后成功发布资源之后，通过 `RCRTCEngine` 中的 `addLiveCdn` 接口来转推第三方 CDN。

设置 CDN 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. 房间模式必须为直播模式。
3. 设置的 CDN 地址不能为空。
4. 最多设置 5 个 CDN 地址，超出会抛出 `50080` cdn 地址配置数量到达上限错误。

5. 如果多次设置相同的地址，会直接返回成功。

• 示例代码：

```
engine.addLiveCdn(url);
```

## 移除配置过的直播 CDN 地址

当主播发布资源成功之后，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. 房间模式必须为直播模式。
3. 移除的 CDN 地址不能为空。
4. 如果移除的地址，之前没有设置过，会直接返回成功。

• 示例代码：

```
engine.removeLiveCdn(url);
```

## CDN 推流回调

融云在低延迟直播旁路推流到融云 CDN 或第三方 CDN 时，支持将推流的状态变化实时通知您的服务器。具体请参见音视频服务端文档：

• [CDN 推流回调](#) 



## 音频模式

## 音频模式

更新时间:2024-08-30

### 场景和音质选择

音频模式与音质合称为音频属性。SDK 针对不同使用场景设计了音频模式，并提供了三个音质选项。音频模式与音质可任意配合使用，达到特殊场景需求。下表列出了几种常见场景下推荐的组合，您也可以直接参考下文中的示例代码。

推荐使用场景	RCRTCAudioScenario 场景枚举	RCRTCAudioQuality 音质枚举	码率
通话，会议场景（默认）	Normal	Speech	人声音质，编码码率最大值为 32Kbps
语聊房，音乐播放场景	MusicChatRoom	Music	标清音乐音质，编码码率最大值为 64Kbps
音乐教学场景	MusicClassRoom	MusicHigh	高清音乐音质，编码码率最大值为 128Kbps

### 设置音频通话质量和模式

设置音频通话质量和音频通话模式 `setAudioConfig` 接口位于 `RCRTCEngine` 类中，可以在加入房间前或者加入房间后，通过 `RCRTCEngine` 实例进行调用设置，详情如下：

```
setAudioConfig(config: RCRTCAudioConfig): number;
```

### 示例代码

以下示例代码为几种常见场景推荐值。

```
/// 普通通话模式(普通音质模式), 满足正常音视频场景, 人声音质, 编码码率最大值为32Kbps
let config = {
quality: RCRTCAudioQuality.Speech,
scenario: RCRTCAudioScenario.Normal,
};
engine.setAudioConfig(config);

/// 音乐教室模式, 提升声音质量, 适用对乐器演奏音质要求较高的场景, 高清音乐音质, 编码码率最大值为128Kbps
let config = {
quality: RCRTCAudioQuality.MusicHigh,
scenario: RCRTCAudioScenario.MusicClassRoom,
};
engine.setAudioConfig(config);

/// 音乐聊天室模式, 提升声音质量, 适用对音乐演唱要求较高的场景, 高清音乐音质, 编码码率最大值为128Kbps
let config = {
quality: RCRTCAudioQuality.MusicHigh,
scenario: RCRTCAudioScenario.MusicChatRoom,
};
engine.setAudioConfig(config);
```

# 混音 混音

更新时间:2024-08-30

混音功能支持将指定的音频文件与本地麦克风采集的音频数据进行混合，支持的用户自定义音频文件格式为：MP3、AAC、M4A、WAV。

Android 平台在调用之前应用必须已经授予 `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />` 权限。  
iOS 平台无特殊需求。

## 提示

如果 Android 10 手机上授予了权限后也出现了混音失败，请参考知识库 [为什么 Android 10 无法使用 startAudioMixing 进行混音？](#)

## 从资源目录下的音频文件进行混音

### 参数说明：

参数	类型	说明
path	String	文件的平台绝对路径，如 <code>../../static/audio/effect_0.mp3</code> 这种路径，需要通过 uni 提供的 <code>plus.io.convertLocalFileSystemURL(path)</code> 方法，将本地 URL 路径转换成平台绝对路径后才可以使用，具体可参考 <a href="#">convertLocalFileSystemURL</a>
mode	RCRTCAudioMixingMode	混音模式：  RCRTCAudioMixingMode.Mix：将音频文件的音频数据与麦克风采集的数据混音发送至对端  RCRTCAudioMixingMode.Replace：将麦克风采集的数据替换为音频文件的音频数据发送至对端  RCRTCAudioMixingMode.None：不做任何操作
playBack	boolean	在调用该方法后是否播放音频文件
loopCount	number	loopCount > 0：循环混音 loopCount 次；loopCount = -1：无限循环；其他取值：混音一次

### 示例代码：

```
engine.startAudioMixingFromAssets(path, RCRTCAudioMixingMode.Mix, true, -1);
```

## 从本地目录下的音频文件进行混音

- 参数说明：

参数	类型	说明
path	String	文件的平台绝对路径
mode	RCRTCAudioMixingMode	混音模式。  RCRTCAudioMixingMode.Mix: 将音频文件的音频数据与麦克风采集的数据混音发送至对端。  RCRTCAudioMixingMode.Replace: 将麦克风采集的数据替换为音频文件的音频数据发送至对端。  RCRTCAudioMixingMode.None: 不做任何操作
playBack	boolean	在调用该方法后是否播放音频文件
loopCount	number	loopCount > 0 : 循环混音 loopCount 次；loopCount = -1 : 无限循环；其他取值：混音一次

- 示例代码：

```
engine.startAudioMixing(path, RCRTCAudioMixingMode.Mix, true, -1);
```

## 调节混音音量

- 参数说明：

参数	类型	说明
volume	number	混音音量，即对端听到的声音音量，取值范围0~100

- 示例代码：

```
engine.adjustAudioMixingVolume(80);
```

## 调节播放音量

- 参数说明：

参数	类型	说明
volume	number	播放音量，即本端听到的声音音量，取值范围0~100

- 示例代码：

```
engine.adjustAudioMixingPlaybackVolume(80);
```

## 获取混音文件时长

- 返回值说明：

类型	说明
number	当前指定的混音文件的总时长，单位毫秒，<0 的情况为接口调用失败。

- 示例代码：

```
let duration = engine.getAudioMixingDuration();
```

## 获取混音进度

- 返回值说明：

类型	说明
double	当前的混音进度，取值范围0~1.0，<0 的情况为接口调用失败。

- 示例代码：

```
let position = engine.getAudioMixingPosition();
```

## 设置混音进度

- 参数说明：

参数	类型	说明
position	double	混音进度，取值范围 0~1.0

- 示例代码：

```
/// 设置从当前文件的 20% 进度处开始混音  
engine.setAudioMixingPosition(0.2);
```

## 暂停混音

- 示例代码：

```
engine.pauseAudioMixing();
```

## 继续混音

- 示例代码：

```
engine.resumeAudioMixing();
```

## 停止混音

- 示例代码：

```
engine.stopAudioMixing();
```

## 设置混音状态监听

开始混音、暂停混音、停止混音和混音文件已自动混流完成的状态监听。

- 示例代码：

```
/// 设置开始混音监听
engine.setOnAudioMixingStartedListener(() => {
// 开始混音
});

/// 设置暂停混音监听
engine.setOnAudioMixingPausedListener(() => {
// 暂停混音
});

/// 设置停止混音监听
engine.setOnAudioMixingStoppedListener(() => {
// 停止混音
});

/// 设置混音文件已自动混流完成监听
engine.setOnAudioMixingFinishedListener(() => {
// 混音文件已自动混流完成
});
```

## 大小流

## 大小流

更新时间:2024-08-30

大小流模式是指在发布资源时上传一大一小两道视频流。

SDK 默认打开发布大小流功能，即每个用户在发布视频资源时自动发布大小两个视频流。小流的分辨率默认跟随大流。

### 提示

在多人音视频通话过程中，大小流模式可有效减少下行带宽占用。订阅方可按需订阅小流。

小视频流与大视频流的分辨率对应关系如下:

大流分辨率	小流分辨率	比例
176X144	176X144	11:9
180X180	180X180	1:1
256X144	256X144	16:9
240X180	240X180	4:3
320X180	256X144	16:9
240X240	180X180	1:1
320X240	240X180	4:3
360X360	180X180	1:1
480X360	240X180	4:3
640X360	256X144	16:9
480X480	180X180	1:1
640X480	240X180	4:3
720X480	240X180	3:2
848X480	256X144	9:5
960X720	240X180	4:3
1280X720	256X144	16:9
1920X1080	256X144	16:9

## 发布方开关大小流

需要在引擎创建时传入视频初始化配置指定开启或关闭大小流，开启功能后，发布资源时会发布大小两道流。



```
/// 创建引擎时开启大小流功能
let videoSetup = { enableTinyStream: true };
let engineSetup = { videoSetup: videoSetup };
RCRTCEngine.create(engineSetup);

/// 创建引擎时关闭大小流功能
let videoSetup = { enableTinyStream: false };
let engineSetup = { videoSetup: videoSetup };
RCRTCEngine.create(engineSetup);
```

## 设置小流属性

开启了大小流功能后，小流会自动使用默认属性。您也可以通过调用 `setVideoConfig` 方法，将 `tiny` 参数传入 `true`，来单独设置小流属性。

```
let config = {
  minBitrate: 200,
  maxBitrate: 900,
  fps: RCRTCVideoFps.Fps15,
  resolution: RCRTCVideoResolution.Resolution_480x640,
};
engine.setVideoConfig(config, true);
```

## 订阅方切换大小流

如果远端用户在加入房间前开启了大小流功能，本地订阅远端视频流时可以通过调用 `subscribe` 方法时传入 `tiny` 参数订阅大流或小流：

```
/// 订阅大流
engine.subscribe(userId, RCRTCMediaType.AudioVideo, false);

/// 订阅小流
engine.subscribe(userId, RCRTCMediaType.AudioVideo, true);
```

## 发布自定义资源

## 发布自定义资源

更新时间:2024-08-30

### 提示

在 Android 平台至少需要调用 `publish(RCRTCMediaType.AudioVideo)` 或 `publish(RCRTCMediaType.Audio)` 方法发布成功音频资源后，再发布自定义视频流才有效。tag 不能包含 `_` 和 `RongCloudRTC` 字符。

## 创建自定义视频资源

- 示例代码：

```
/// 从本地目录下的资源文件创建自定义视频资源，路径需要是平台绝对路径  
engine.createCustomStreamFromFile(path, tag);
```

## 设置自定义视频属性

- 示例代码：

```
let config = {  
  minBitrate: 500,  
  maxBitrate: 2200,  
  fps: RCRTCVideoFps.Fps24,  
  resolution: RCRTCVideoResolution.Resolution_720x1280,  
};  
engine.setCustomStreamVideoConfig(tag, config);
```

## 设置自定义视频资源预览窗口

- 示例代码：

导入预览窗口组件

```
// 导入 RCRTCView
import RCRTCView from '@uni_modules/RongCloud-RTCWrapper/components/RCRTCView';

// 声明 RCRTCView
export default {
  components: {
    RCRTCView,
  },
}
```

## 添加预览窗口

```
<!-- 增加 RCRTCView 组件, fitType: 视频填充模式, mirror: 视频是否镜像显示 -->
<RCRTCView class="customStreamView" ref="customStreamView" :fitType="RCRTCViewFitType.Center"
:mirror="true">
</RCRTCView>
```

## 设置预览窗口

```
// 设置预览窗口
engine.setLocalCustomStreamView(tag, this.$refs.customStreamView.getNativeViewRef(), (code) => {
  if (code === 0) {
    // 设置成功
  } else {
    // 设置失败
  }
});
```

## 发布自定义视频资源

- 示例代码：

```
engine.publishCustomStream(tag);
```

## 设置远端自定义视频资源发布监听

- 示例代码：

```
engine.setOnRemoteCustomStreamPublishedListener(({roomId, userId, tag, type}) => {  
  // roomId 房间 ID  
  // userId 远端用户 ID  
  // tag 远端自定义视频资源 tag  
  // type 资源类型  
});
```

## 订阅远端自定义视频资源

- 示例代码：

```
engine.subscribeCustomStream(userId, tag);
```

## 设置远端自定义视频资源预览窗口

- 示例代码：

```
// 设置预览窗口  
engine.setRemoteCustomStreamView(userId, tag, this.$refs.remoteCustomStreamView.getNativeViewRef(),  
(code) => {  
  if (code === 0) {  
    // 设置成功  
  } else {  
    // 设置失败  
  }  
});
```

## 水印处理

## 水印处理

更新时间:2024-08-30

SDK 从 5.2.5 版本开始支持该功能。

融云支持在直播中在视频流上添加水印。添加水印有两种控制方式，本文仅介绍方案一：

- 方案一：使用客户端 SDK 提供的 `setWatermark` 方法添加图片水印。客户端发布的视频流即带有图片水印，因此订阅分流或合流的直播观众均会看到带水印的视频流。
- 方案二：使用服务端 API 的 `/rtc/mcu/config` 接口，在服务端处理，添加时间戳水印、文字水印或图片水印。这种方式支持为单人视频流或合流视频添加水印，但只有订阅合流的观众可看到带水印的视频流。本文不介绍服务端的处理方案，如有需要，请参见服务端文档[直播合流](#)。

### 提示

方案一适用于实现 App 客户端用户自主添加个性化水印；方案二更适用于由 App 添加统一风格的水印。客户端与服务端添加的水印相互独立。如果同时使用，则订阅合流的观众可能会看到水印叠加。

## 设置水印

RCRTCEngine 内置了设置水印的接口，通过调用 `setWatermark` 方法即可实现为相机、自定义文件、共享桌面采集到的视频流添加水印的功能。每一道视频流水印设置是独立的。

## 方法

```
/**
 * 设置水印
 * @param path 水印图片的本地路径
 * @param positionX 水印的位置,x坐标,参数取值范围 0 ~ 1,浮点型
 * @param positionY 水印的位置,y坐标,参数取值范围 0 ~ 1,浮点型
 * @param zoom 图片缩放系数,参数取值范围 0 < zoom <= 1
 * SDK 内部会根据视频分辨率计算水印实际的像素位置和尺寸。
 * @return 接口调用状态码 0: 成功, 非0: 失败
 */
setWatermark(path: string, positionX: number, positionY: number, zoom: number): number;
```

参数	类型	必填	说明
path	String	是	水印图片的本地路径
positionX	number	是	水印的位置，x坐标，浮点型。注意：参数取值范围 0 ~ 1，SDK 内部会根据视频分辨率计算水印实际的像素位置。

参数	类型	必填	说明
positionY	number	是	水印的位置，y坐标，浮点型。注意：参数取值范围 0 ~ 1，SDK 内部会根据视频分辨率计算水印实际的像素位置。
zoom	number	是	水印的缩放比，注意：参数取值范围 $0 < zoom \leq 1$ ，SDK 内部会根据视频分辨率计算水印实际的大小

- zoom：水印的缩放比，取值范围  $0 < zoom \leq 1$ 。例如，水印图片的宽度  $480 \times 0.2 = 96$ ，高度根据图片原始宽高比等比例缩放。

例如，当前视频的编码分辨率是 480（宽） $\times$  640（高），且 position 参数被您设置为 (0.1, 0.1)、zoom 设置为 0.2。那么水印的左上坐标点就是  $(480 \times 0.1, 640 \times 0.1)$ ，即 (48, 64)，水印的宽度是  $480 \times 0.2 = 96$ ，水印的高度会根据水印图片的宽高比由 SDK 自动算出。

## 清除水印

通过调用 RCRTCEngine 对象的 removeWatermark 方法移除客户端设置的水印。

## 方法

```
// 清除客户端设置的水印
/**
 * 移除水印
 * @return 接口调用状态码 0：成功，非0：失败
 */
removeWatermark(): number;
```

## 示例代码

```
//添加水印的回调
engine.setOnWatermarkSetListener(({code,message}) => {
if (code == 0){
// 水印设置成功
}
});

//添加水印
let path = '***'
engine.setWatermark(path, 0.2, 0.2, 0.2)

//清除水印的回调
engine.setOnWatermarkRemovedListener(({code,message}) => {
if (code == 0) {
//水印清除成功
}
});
//清除水印
engine.removeWatermark();
```

## 注意事项

- 添加水印图片不宜过大。建议图片宽高均不超过 200px，否则会影响图片在安卓原生层转 Bitmap 的生成。
- 禁止程序循环反复调用添加/清除水印接口。

- 安卓端水印功能需要在 manifest.json 的 App 权限配置 中勾选下方两项外部存储读写权限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

## 通话数据统计

## 通话数据统计

更新时间:2024-08-30

音视频通话过程中，底层音视频库会按照 RCRTCEngineSetup 中的 statsReportInterval 指定的时间间隔（默认1秒）上报通话的详细数据，上层依据此数据可进行提示，状态判断等处理。

可通过 RCRTCEngine 中的 setOnNetworkStatsListener 等方法注册监听。

### 详细说明

- 回调参数说明：

setOnNetworkStatsListener 说明：

回调参数	回调类型	说明
stats	RCRTCNetworkStats	网络状态信息

setOnLocalAudioStatsListener 说明：

回调参数	回调类型	说明
stats	RCRTCLocalAudioStats	音频发送状态信息

setOnLocalVideoStatsListener 说明：

回调参数	回调类型	说明
stats	RCRTCLocalVideoStats	视频发送状态信息

setOnLocalCustomVideoStatsListener 说明：

回调参数	回调类型	说明
tag	string	自定义视频资源 TAG
stats	RCRTCLocalVideoStats	自定义视频资源发送状态信息

setOnRemoteAudioStatsListener 说明：

在直播场景下，主播角色的用户可通过该回调获取房间内正在说法的其他主播及其音量。在会议场景下，与会用户可通过该回调获取房间内正在说法的其他用户及其音量。



回调参数	回调类型	说明
roomId	string	远端房间 ID
userId	string	远端用户 ID
stats	RCRTCRemoteAudioStats	音频接收质量状态信息

setOnRemoteVideoStatsListener 说明：

回调参数	回调类型	说明
roomId	string	远端房间 ID
userId	string	远端用户 ID
stats	RCRTCRemoteVideoStats	视频接收质量状态信息

setOnLiveMixAudioStatsListener 说明：

回调参数	回调类型	说明
callback	RCRTCRemoteAudioStats	合流音频接收质量状态信息

setOnLiveMixMemberAudioStatsListener 说明：

在直播场景下，观众可以通过该回调获取正在说话的主播的音量。

回调参数	回调类型	说明
callback	OnLiveMixMemberAudioStatsResult	上报远端分流音频统计信息。其中 userId (string) 为远端用户 ID，volume (number) 为音量。

setOnLiveMixVideoStatsListener 说明：

回调参数	回调类型	说明
stats	RCRTCRemoteVideoStats	合流视频接收质量状态信息

setOnRemoteCustomVideoStatsListener 说明：

回调参数	回调类型	说明
roomId	string	远端房间 ID
userId	string	远端用户 ID
tag	string	远端自定义视频资源 TAG
stats	RCRTCRemoteVideoStats	自定义视频资源接收质量状态信息

- RCRTCNetworkStats

属性	类型	说明
type	RCRTCNetworkType	网络类型，未知/WIFI/移动网络
ip	string	IP 地址
sendBitrate	number	发送码率，单位：kbps
receiveBitrate	number	接收码率，单位：kbps
rtt	number	往返延时，单位：ms

- RCRTCLocalAudioStats

属性	类型	说明
codec	RCRTCAudioCodecType	编码类型，PCMU/OPUS
bitrate	number	码率，单位：kbps
volume	number	音量
packageLostRate	number	丢包率：取值范围是 0-100
rtt	number	往返延时，单位：ms

- RCRTCLocalVideoStats

属性	类型	说明
tiny	bool	小流标记，true 小流，false 大流
codec	RCRTCVideoCodecType	编码类型，H264
bitrate	number	码率，单位：kbps
fps	number	视频帧率
width	number	视频宽度
height	number	视频高度
packageLostRate	number	丢包率：取值范围是 0-100
rtt	number	往返延时，单位：ms

- RCRTCRemoteAudioStats

属性	类型	说明
codec	RCRTCVideoCodecType	编码类型，H264

属性	类型	说明
bitrate	number	码率，单位：kbps
volume	number	音量
packageLostRate	number	丢包率：取值范围是 0-100
rtt	number	往返延时，单位：ms

- RCRTCRemoteVideoStats

属性	类型	说明
codec	RCRTCVideoCodecType	编码类型，H264
bitrate	number	码率，单位：kbps
fps	number	视频帧率
width	number	视频宽度
height	number	视频高度
packageLostRate	number	丢包率：取值范围是 0-100
rtt	number	往返延时，单位：ms

## 示例代码

```
/// 设置状态监听回调

engine.setOnNetworkStatsListener((stats) => {
// 网络状态信息
});
engine.setOnLocalAudioStatsListener((stats) => {
// 音频发送状态信息
});
engine.setOnLocalVideoStatsListener((stats) => {
// 视频发送状态信息
});
engine.setOnRemoteAudioStatsListener(({roomId, userId, stats}) => {
// 音频接收状态信息
});
engine.setOnRemoteVideoStatsListener(({roomId, userId, stats}) => {
// 视频接收状态信息
});
engine.setOnLiveMixAudioStatsListener((stats) => {
// 合流音频接收状态信息
});
engine.setOnLiveMixVideoStatsListener((stats) => {
// 合流视频接收状态信息
});
engine.setOnLocalCustomVideoStatsListener(({tag, stats}) => {
// 自定义视频发送状态信息
});
engine.setOnRemoteCustomVideoStatsListener(({roomId, userId, tag, stats}) => {
// 自定义视频接收状态信息
});

/// 调用以上接口参数为空，即为取消状态监听回调。如下示例
/// 取消网络状态监听
engine.setOnNetworkStatsListener();
```

## 网络探测

## 网络探测

更新时间:2024-08-30

SDK 从 5.2.5 版本开始支持该功能。

`startNetworkProbe` 方法支持用户在加入房间前进行网络质量探测，通过相应的回调将当前网络往返时延、上下行丢包率、网络质量数据返回给上层应用。

**提示**

请在加入房间前完成网络质量探测。SDK 不支持在音视频通话过程中进行网络质量探测。

## 开启 RTC 网络探测

加入房间前，通过调用 `RCRTCEngine` 对象的 `startNetworkProbe` 方法开启 RTC 网络质量探测，开启探测成功后，会回调注册的回调来反馈探测结果。每隔约 2 秒回调一次上下行网络的带宽、丢包、网络抖动和往返时延等数据。通过 `setOnNetworkProbeUpLinkStatsListener`、`setOnNetworkProbeDownLinkStatsListener`、`setOnNetworkProbeFinishedListener` 方法接收相关的事件。

探测总时长约为 30 秒，30 秒后会自动停止探测并触发 `setOnNetworkProbeFinishedListener` 回调方法通知上层应用探测结束，此时回调中的 `code` 值为 0。

**提示**

在探测自动结束（`setOnNetworkProbeFinishedListener code == 0`）前，如调用 `joinRoom` 或 `destroy` 等 API 方法会打断当前正在进行的 RTC 网络探测，此时回调中的 `code` 值不为 0。

```

// 开启网络探测的回调
engine.setOnNetworkProbeStartedListener(({code, errMsg}) => {
if (res.code == 0) {
// 开启网络探测成功
}
});

// 开启网络探测
engine.startNetworkProbe();

engine.setOnNetworkProbeUpLinkStatsListener(({qualityLevel,packetLostRate,rtt}) => {
//汇报网络探测上行数据
});

engine.setOnNetworkProbeDownLinkStatsListener(({qualityLevel,packetLostRate,rtt}) => {
//汇报网络探测下行数据
});

engine.setOnNetworkProbeFinishedListener(({code, errMsg}) => {
// 网络探测完成 code 为 0 时是正常结束，非 0 为探测中断
});

```

## 停止 RTC 网络探测

开启探测成功约 30 秒后，会自动停止探测，如果不想等待自动结束，可调用 `stopNetworkProbe` 方法强制停止探测。

```

// 停止网络探测的回调
engine.setOnNetworkProbeStoppedListener(({code,errMsg}) => {
if (code == 0) {
// 停止网络探测成功
}
});

// 停止网络探测
engine.stopNetworkProbe();

```

## 更新日志

## 更新日志 [5.2.5] - 2022-10-17

更新时间:2024-08-30

---

- 修改引擎生命周期
- 新增视频水印功能
- 新增融云内置 CDN 功能（适用于直播场景）
- 新增网络检测和麦克风检测功能
- 新增主播观众角色切换功能（适用于直播场景）
- 新增 sei 功能

### [5.1.1] - 2022-02-14

- 适配 Rongcloud RTC 5.1.17 SDK

## 状态码

## 状态码

更新时间:2024-08-30

状态码	说明
-1	未知错误
-10	引擎已销毁
-11	引擎未初始化
-12	切换摄像头失败
-13	本地用户未加入房间
-14	远端用户未加入房间
-15	启动混音失败
-16	资源不存在
-17	参数错误
-18	创建自定义资源失败
-19	接口暂不支持
-20	打开文件失败
0	成功
1	IM 错误
2	HTTP 错误
40001	操作的用户已经不在该房间
40002	server 内部错误
40003	房间不存在错误
40004	非法的用户 id
40005	重复加入已经存在的 RTC room
40016	未开通视频直播服务，参见控制台文档 <a href="#">如何开通音视频服务</a>
40017	未开通音频直播服务，参见控制台文档 <a href="#">如何开通音视频服务</a>
40018	生成 token 失败
40021	用户被封禁，请确认用户当前的状态。
40022	连麦邀请的房间不存在
40023	连麦邀请的人不在房间内
40024	连麦的人正在邀请中
40025	连麦取消邀请，但是邀请信息不存在
40026	应答邀请时，但邀请信息不存在
40027	应答邀请时，sessionID 不一致
40029	房间已经存在



状态码	说明
40030	房间类型不支持
40031	直播身份切换失败
40032	已经加入了房间
40033	(跨App通信场景) 允许互通App或网关地址未配置
50000	初始化失败, IM Server 未连接
50001	调用方法时输入参数错误
50002	加入相同房间错误, 表示用户在客户端重复加入相同的房间
50003	调用房间内接口时, 不在房间中
50004	未开通音视频服务, 参见控制台文档 <a href="#">如何开通音视频服务</a>
50006	RTC token为空, 请查看是否还在房间内或者房间是否销毁
50007	状态异常,
50008	观众加聊天室成功, kv 回调超时(30s)
50009	观众加聊天室调用 api 失败
50010	HTTP 请求超时
50011	HTTP 响应错误 (含 500, 404, 405 等错误)
50012	网络不可用
50020	重复发布资源错误
50021	初步会话协商错误, 没有消息的音视频参数
50022	会话协商错误, 协商数据不匹配或者其他问题
50023	发布的资源个数已经到达上限
50024	取消发布不存在的资源
50025	创建本地 Offer 失败
50026	创建本地 Answer 失败
50030	订阅不存在的音视频资源
50031	资源重复订阅
50032	取消订阅不存在的音视频资源
50065	RTC connection is null
50066	发布的 Stream 为空
50067	设置远端 SDP Error
50068	设置本地 SDP Error
50069	解析 JSON 错误
50070	服务端返回的 LiveInfo 为空
50071	PeerConnection 添加 Stream 失败
50073	RTC Token 无效
50074	调用相关接口, 没有加入主房间错误
50075	操作的副房间号码和主房间号码一致错误
50076	取消的跨房间连麦请求不存在

状态码	说明
50077	响应的跨房间连麦请求不存在
50079	服务端返回的信息异常
50080	CDN 地址配置数量到达上限（最大为10个）
50081	帧时间戳非法
50082	解码视频帧失败
50090	音效文件数量已经到达最大数量
50091	处理非法的 soundId，如停止播放没有播放过的音效文件 id，此音效 ID 没有预设或者播放过
50100	自动重连出现异常
51000	视频硬编码初始化失败，SDK内部默认切换到软编码
51001	视频硬编码过程中失败，SDK内部默认切换到软编码
51002	视频硬解码初始化失败，SDK内部默认切换到软解码
51003	视频硬解码过程中失败，SDK内部默认切换到软解码
51004	无效的 Camera Id
51005	未找到 Camera Device
51006	打开 Camera 失败
51007	初始化 RCRTCEngine 超时
51100	创建 SDP 中的 Answer 失败
51200	CameraManager 已被释放
51201	操作被取消
51202	AudioManager 已被释放
54001	订阅 CDN 流时未找到 CDN Player 模块
54002	CDN 服务宕机，视频中断，一般是视频源异常或者不支持的视频类型
54008	CDN 数据连接中断或音视频源格式不支持
54009	初始化 CDN Player 模块异常
54010	获取屏幕共享权限失败
54011	已开启录屏功能，不能再次启用录屏