实时音视频
通话 CallLib/Kit
uni-app 5.X

2024-08-30

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。

> 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
Android	Java	armeabi-v7a、arm64-v8a、x86、x86-64	系统版本 4.4 及以上
iOS	Objective-C	---	系统版本 9.0 及以上
Windows	C++、Electron	x86、x86-64	Windows 7 及以上
Linux	C、Electron	---	推荐 Ubuntu 16.04 及以上；其他发行版需求请咨询商务
MacOS	Electron	---	系统版本 10.10 及以上
Web	Javascript	---	详见客户端文档「Web 兼容性」
Flutter	dart	---	Flutter 2.0.0 及以上
uni-app	Javascript	---	uni-app 2.8.1 及以上
React Native	Javascript	---	React Native 0.65 及以上
Unity	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1]
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。

> 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。

> 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。

运行示例项目（Demo）

更新时间:2024-08-30

融云 uni-app CallLib 音视频产品提供一个 QuickDemo 示例项目（GitHub ），集中演示了融云音视频通话在 Android 和 iOS 端的功能，以便开发者体验产品，快速集成，实现单群聊、音视频通话等场景需求。

环境要求

- **HBuilder X**： 建议最新版本
- **Android**：4.4 及以上
- **iOS**：9.0 及以上

融云开发者账户

- 注册开发者账号 。注册成功后，控制台会默认自动创建您的首个应用，默认生成开发环境下的 App Key，使用国内数据中心。
- 获取开发环境的应用 App Key 。如不使用默认应用，请参考 如何创建应用，并获取对应环境 App Key 和 App Secret 。

> 提示
>
> 每个应用具有两个不同的 App Key，分别对应开发环境与生产环境，两个环境之间数据隔离。在您的应用正式上线前，可切换到使用生产环境的 App Key，以便上线前进行测试和最终发布。

- 如果仅为体验 QuickDemo 创建应用，建议选择国内数据中心。如果选择海外数据中心，则需要额外在 QuickDemo 中修改 SDK 连接的服务地址。配置方法可参见 数据中心 。

开通音视频服务

开发环境下的每个应用均可享有 10000 分钟免费体验时长。如果在开发环境下开通音视频服务，可直接按照以下步骤开通音视频服务。服务开通后即可开始免费体验和测试。免费体验时长用完即止。

如果在生产环境下开通音视频服务，则需要先预存费用，才可开通。详情请参考开通音视频服务 。

获取用户 Token

用户 Token 是与用户 ID 对应的身份验证令牌，是应用程序的用户在融云的唯一身份标识。应用客户端必须与融云建立 IM 连接，连接时必须传入 Token。

在实际业务运行过程中，应用客户端需要通过应用的服务端调用 IM Server API 申请取得 Token。详见 Server API 文档 注册用户 。

在本教程中，为了快速体验和测试 SDK，我们将使用控制台「北极星」开发者工具箱，从 API 调试页面调用 获取 Token 接口，获取到 **userId** 为 1 的用户的 Token。提交后，可在返回正文中取得 Token 字符串。

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"code":200,"userId":"1","token":"gxld6GHx3t1eDxof1qtxxYrQcjkbhl1V@sgyu.cn.example.com;sgyu.cn.example.c
```

重复上一步，获取 **userId** 为 2 的用户的 Token。记录下该 Token，用于体验时使用。

运行 QuickDemo

1. 在运行 QuickDemo 前请确保已完成上述步骤。以下是检查清单:

   - 已注册融云开发者账户
   - 已准备好 App Key
   - 已开通音视频服务免费体验，且已等待 30 分钟
   - 已获取用于体验的两个 Token
   - 下载并安装 HBuilder X

2. 克隆 QuickDemo（Github） 代码。

   克隆下载示例代码

   ```
   git clone https://github.com/rongcloud/uni-calllib.git
   ```

   注意，QuickDemo 示例代码在 `uni-calllib/example` 目录下。

3. 在 HBuilderX 中，打开 `uni-calllib/example`。

4. 前往 DCloud 插件市场，购买下列融云 uni-app 插件，或将插件下载到本地：

   - 融云即时通讯 SDK uni 原生插件 `RCUniIMV2`：
     https://ext.dcloud.net.cn/plugin?id=9227
   - 融云实时音视频 SDK uni 原生插件 `RCUniCall`：
     https://ext.dcloud.net.cn/plugin?id=6372
   - 融云即时通讯 SDK uni TS 插件 `RongCloud-IMWrapper-V2`：
     https://ext.dcloud.net.cn/plugin?id=9225
   - 融云实时音视频 SDK uni TS 插件 `RongCloud-CallWrapper`：
     https://ext.dcloud.net.cn/plugin?id=7136

5. 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 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 **RCUniIMV2/RCUniCall**。

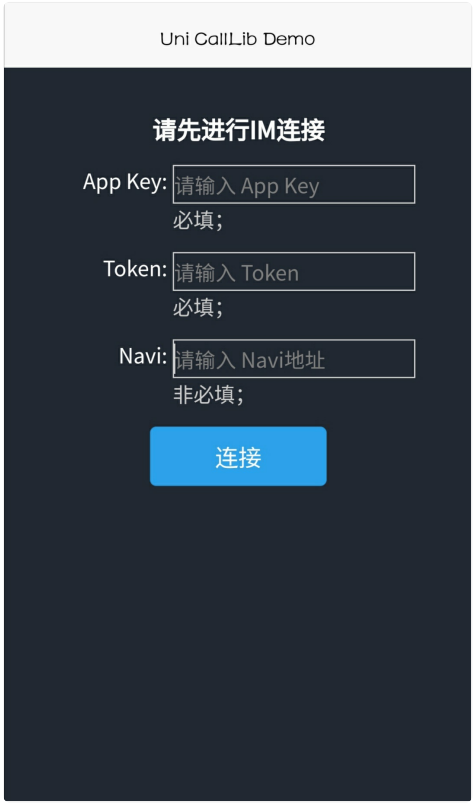
- 本地打包适用：

  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 **RCUniIMV2/RCUniCall**。

6. 在 HBuilder X 中，点击 运行 -> 点击 运行到手机或模拟器 -> 点击 制作自定义调试基座

7. 制作基座完成后：

- 在 HBuilder X 中，点击 运行 -> 点击运行到手机或模拟器 -> 点击 运行基座选择 -> 选择 自定义调试基座。
- 连接 Android 或 iOS 手机，HBuilder X 点击 运行 -> 点击 运行到手机或模拟器 -> 点击 已连接的手机。



导入CallLib SDK

更新时间:2024-08-30

导入 SDK 时，必须先引入即时通讯（IM）和实时音视频（RTC）的 uni 原生插件，再安装 Typescript 层的依赖项。

步骤 1：导入 uni 原生插件

由于 uni-app CallLib 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。

请使用 HBuilder X 将即时通讯（IM）插件和音视频通话（RTC）插件导入应用工程。

> 提示
>
> 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：https://ext.dcloud.net.cn/plugin?id=9227
> - 实时音视频 uni 原生插件 RCUniCall：https://ext.dcloud.net.cn/plugin?id=6372

1. 前往 uni-app 插件市场，购买或下载融云 uni-app 原生插件 RCUniIMV2 和 RCUniCall。
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 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 **RCUniIMV2/RCUniCall**。
   - 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 **RCUniIMV2/RCUniCall**。

        请参照以下 `nativeplugins` 文件目录结构：

```
nativeplugins
└──── RongCloud-IM-V2
    ├──── android
    ├──── ios
    └──── package.json
└──── RongCloud-Call
    ├──── android
    ├──── ios
    └──── 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。

即时通讯依赖项

请从 uni-app 插件市场安装 `RongCloud-IMWrapper-V2`：

https://ext.dcloud.net.cn/plugin?id=9225

> 如果您曾使用 NPM 安装过即时通讯依赖项 `@rongcloud/imlib-uni` 或使用的是 `RCUniIM`，请在升级时替换为 `RCUniIMV2`，可参考 uni-app IM 的文档修改对应的代码。

音视频通话依赖项

请从 uni-app 插件市场安装 `RongCloud-CallWrapper`：

https://ext.dcloud.net.cn/plugin?id=7136

> 如果您曾使用 NPM 安装过音视频通话依赖项 `@rongcloud/calllib-uni`，请在升级时替换为从插件市场安装的方式，并注意修改初始化代码。

步骤三：在项目中引用

引用 IM SDK：

```
import RCIMIWEngine from "@/uni_modules/RongCloud-IMWrapper-V2/js_sdk/RCIMEngine"
```

引用 Calllib SDK：

```
import * as CallLib from "@/uni_modules/RongCloud-CallWrapper/lib/index"

CallLib.init({});
```

实现音视频通话

更新时间:2024-08-30

CallLib 是在 RTCLib 基础上，额外封装了一套音视频呼叫功能 SDK，包含了单人、多人音视频呼叫的各种场景和功能。

前置条件

- 注册开发者账号。注册成功后，控制台会默认自动创建您的首个应用，默认生成开发环境下的 App Key，使用国内数据中心。
- 获取开发环境的应用 App Key。如不使用默认应用，请参考 如何创建应用，并获取对应环境 App Key 和 App Secret。

> 提示
>
> 每个应用具有两个不同的 App Key，分别对应开发环境与生产环境，两个环境之间数据隔离。在您的应用正式上线前，可切换到使用生产环境的 App Key，以便上线前进行测试和最终发布。

- 完成开通音视频服务。请开通音视频通话服务。
- 完成导入 CallLib SDK。

平台兼容性

平台	说明
Android	系统版本 4.4 及以上
iOS	系统版本 9.0 及以上

Demo 项目

融云提供了 uni-app 通话 Demo 项目。详见运行 Demo 项目。

步骤 1：配置权限

在项目 manifest.json 源码视图中 Android 打包配置 `permission` 节点下新增 CallLib 所需的设备权限。如果已经有以下的权限，则可以不用再添加。

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

步骤 2：使用 App Key 初始化

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，所以要先初始化 IM SDK（RongCloud-IMWrapper-V2 ）。IM SDK 的初始化方法接受一个 App Key 参数，和一个可选的 IM 引擎配置（RCIMIWEngineOptions ）。IM 引擎相关配置详见 IMLib 文档引擎配置。

```
import RCIMIWEngine from "@/uni_modules/RongCloud-IMWrapper-V2/js_sdk/RCIMEngine"

let appKey = 'xxx';
let options = {};
let yourEngine = null;
RCIMIWEngine.create(appKey,options).then((res) => {
//本地代码保存引擎
yourEngine = res
});
```

IM SDK 初始化完成之后，初始化 CallLib SDK。

```
import * as CallLib from "@/uni_modules/RongCloud-CallWrapper/lib/index"

CallLib.init({});
```

步骤 3：监听通话事件

CallLib 提供以下呼叫相关的事件。

1. 监听通话呼入，通过回调 `CallLib.onCallReceived` 监听。

```
CallLib.onCallReceived( (res)=> {
console.log("Engine:OnCallReceived=>"+"监听通话呼入, 目标id=>", res.data.targetId);
});
```

2. 已建立通话，通过回调 `CallLib.onCallConnected` 监听。如果是一对一通话，可在该方法触发后展示远端画面。

```
CallLib.onCallConnected((res)=>{
console.log("Engine:OnCallConnected=>"+"通话接通时，通过回调 onCallConnected 通知当前 call 的详细信息", res)
});
```

3. 对端用户加入了通话，通过回调 `CallLib.onRemoteUserJoined` 监听。如果是多人通话，可在该方法触发后展示远端用户的画面。

```
CallLib.onRemoteUserJoined((res)=>{
console.log("Engine:OnRemoteUserJoined=>"+"主叫端拨出电话，被叫端收到请求后，加入通话，被叫端Id为=>",
res.data.userId);
})
```

4. 通话中的某一个参与者，邀请好友加入通话，通过回调 CallLib.OnRemoteUserInvited 监听。

```
CallLib.onRemoteUserInvited((res)=>{
console.log("Engine:OnRemoteUserInvited=>"+"通话中的某一个参与者，邀请好友加入通话，发出邀请请求后,远端Id为=>",
res.data.userId)
})
```

5. 通话结束，通过回调 CallLib.OnCallDisconnected 监听。reason 表示挂断原因。具体请参见挂断通话原因。

```
CallLib.onCallDisconnected((res)=>{
console.log("Engine:OnCallDisconnected=>"+"挂断成功, 挂断原因=>", res.data.reason)
})
```

6. 通话中的远端参与者离开，通过回调 CallLib.OnRemoteUserLeft 监听。reason 表示挂断原因。具体请参见挂断通话原因。

```
CallLib.onRemoteUserLeft((res)=>{
console.log("Engine:OnRemoteUserLeft=>"+"远端用户挂断，远端Id为=>", res.data.reason)
})
```

步骤 4：获取用户 Token

用户 Token 是与用户 ID 对应的身份验证令牌，是应用程序的用户在融云的唯一身份标识。应用客户端在使用融云即时通讯功能前必须与融云建立 IM 连接，连接时必须传入 Token。

在实际业务运行过程中，应用客户端需要通过应用的服务端调用 IM Server API 申请取得 Token。详见 Server API 文档 注册用户 。

在本教程中，为了快速体验和测试 SDK，您可以控制台「北极星」开发者工具箱 IM Server API 调试 页面调用接口获取 Token 用于测试。

步骤 5：建立 IM 连接

1. 使用 setOnConnectionStatusChangedListener 监听 IM 连接状态的变化，连接状态发生变化时返回

RCIMIWConnectionStatus 。详见连接状态监听。

```
yourEngine.setOnConnectionStatusChangedListener((res) => {
//...
});
```

2. 使用上方获取的 Token，模拟 userId 为 1 的用户连接到融云服务器。调用结果会直接通过 connect 方法中传入的 RCIMIWConnectCallback 返回。

```
let callback = {
onDatabaseOpened:(res) => {
//...
},
onConnected:(res) => {
//...
}};
let code = await yourEngine.connect(token, timeout, callback);
```

SDK 已实现自动重连机制，请参见 IMLib 文档 连接。

步骤 6：发起呼叫

主动呼叫分为发起单人通话和发起多人通话，可根据实际需求调用。多人通话的场景必须在一个群组内。

发起单人通话

```
CallLib.startSingleCall(targetId, type, extra);
```

参数	类型	必填	说明
targetId	String	是	对方的userId
type	Number	是	呼叫类型（0:音频 1:音视频）
extra	String	否	附加信息默认传''

发起多人通话

```
CallLib.startGroupCall(groupId,userIds,observerUserIds,type,extra);
```

参数	类型	必填	说明
groupId	String	是	群组 Id
userIds	Array	是	被呼叫的群内成员 Id

参数	类型	必填	说明
observerUserIds	Array	是	观察者ID默认为空数组
type	Number	是	呼叫类型（0:音频 1:音视频）
extra	String	否	附加信息默认传"

步骤 7: 展示通话视图

您需要使用 SDK 提供的视频组件展示本地预览视图和远端用户的视频视图。

1. 发起单人或多人通话后，可使用 CallLib.getCurrentCallSession 获取当前通话中的成员信息。

```
let currentCallSession = CallLib.getCurrentCallSession()
let users = currentCallSession.users
```

2. 直接在 .nvue 中使用 <RongCloud-Call-RCUniCallView ref="localVideoView"></RongCloud-Call-RCUniCallView> 视频组件，该组件不需要额外引入。

```
<RongCloud-Call-RCUniCallView ref="localVideoView" ></RongCloud-Call-RCUniCallView>
```

3. 通过 CallLib.setVideoView 设置视频预览窗口。在调用 CallLib.setVideoView 前一定要确保 UI 界面已经渲染完成，所以要延迟调用。

```
setTimeout(()=>{
CallLib.setVideoView(userId, this.$refs.localVideoView.ref, type,isZOrderOnTop);
},200);
```

参数	类型	必填	说明
userId	String	是	用户id
ref	String	是	对视频容器的引用
type	Number	是	视频显示模式 0 铺满 1 自适应
isZOrderOnTop	Boolean	否	是否置顶 （仅Android 需要设置）

步骤 8：呼叫接听

调用 CallLib.accept 方法接听通话。

```
CallLib.accept()
```

步骤 9：呼叫挂断

调用 CallLib.hangup 方法挂断通话

```
CallLib.hangup()
```

步骤 10：离线推送通知

如果需要在 App 已经被系统回收的情况下也收到呼叫推送通知，则必须集成远程推送。请参见 IM 客户端文档 推送概述 。

主叫方

更新时间:2024-08-30

主动呼叫分为发起单人通话和发起多人通话，可根据实际需求调用。多人通话的场景必须在一个群组内。

发起单人通话

参数说明

参数	类型	必填	说明
targetId	String	是	对方的userId
type	Number	是	呼叫类型（0:音频 1:音视频）
extra	String	否	附加信息默认传''

代码示例

```
CallLib.startSingleCall(targetId, type, extra);
```

发起多人通话

参数说明

参数	类型	必填	说明
groupId	String	是	群组 Id
userIds	Array	是	被呼叫的群内成员 Id
observerUserIds	Array	是	观察者ID默认为空数组
type	Number	是	呼叫类型（0:音频 1:音视频）
extra	String	否	附加信息默认传''

代码示例

```
CallLib.startGroupCall(groupId,userIds,observerUserIds,type,extra);
```

监听呼叫

已建立通话，通过回调 `call.onCallConnected` 监听。

代码示例

```
CallLib.onCallConnected((res)=>{
console.log("Engine:OnCallConnected=>"+"通话接通时，通过回调 onCallConnected 通知当前 call 的详细信息", res)
});
```

对端用户加入了通话，通过回调 CallLib.onRemoteUserJoined 监听。

代码示例

```
CallLib.onRemoteUserJoined((res)=>{
console.log("Engine:OnRemoteUserJoined=>"+"主叫端拨出电话，被叫端收到请求后，加入通话，被叫端Id为=>",
res.data.userId);
})
```

挂断通话

调用 CallLib.hangup 方法挂断通话

代码示例

```
CallLib.hangup()
```

通话结束，通过回调 CallLib.OnCallDisconnected 监听。

代码示例

```
CallLib.onCallDisconnected((res)=>{
console.log("Engine:OnCallDisconnected=>"+"挂断成功, 挂断原因=>", res.data.reason)
})
```

通话中的远端参与者挂断，通过回调 call.OnRemoteUserLeft 监听。

代码示例

```
CallLib.onRemoteUserLeft((res)=>{
console.log("Engine:OnRemoteUserLeft=>"+"远端用户挂断，远端Id为=>", res.data.reason)
})
```

邀请通话

调用 CallLib.inviteUsers 方法邀请用户加入当前通话（仅限群组）。

参数说明

参数	类型	必填	说明
userIds	Array	是	邀请的用户 ID 列表
observerUserIds	Array	是	被邀请观察者id列表 (只能听或看，不能推流的用户)默认传空数组[]

代码示例

```
CallLib.inviteUsers(userIds,observerUserIds);
```

被叫方 监听来电

更新时间:2024-08-30

监听通话呼入，通过回调 CallLib.onReceivedCall 监听。

代码示例

```
CallLib.onCallReceived( (res)=> {
console.log("Engine:OnCallReceived=>"+"监听通话呼入, 目标id=>", res.data.targetId);
});
```

接听通话

调用 CallLib.accept 方法接听通话。

代码示例

```
CallLib.accept()
```

拒绝/挂断通话

调用 CallLib.hangup 方法拒绝/挂断通话

代码示例

```
CallLib.hangup()
```

通话结束，通过回调 CallLib.OnCallDisconnected 监听。

代码示例

```
CallLib.onCallDisconnected((res)=>{
console.log("Engine:OnCallDisconnected=>"+"挂断成功, 挂断原因=>", res.data.reason)
})
```

通话中的远端参与者挂断，通过回调 CallLib.OnRemoteUserLeft 监听。

代码示例

```
CallLib.onRemoteUserLeft((res)=>{
console.log("Engine:OnRemoteUserLeft=>"+"远端用户挂断，远端Id为=>", res.data.reason)
})
```

邀请通话

调用 CallLib.inviteUsers 方法邀请用户加入当前通话（仅限群组）。

参数说明

参数	类型	必填	说明
userIds	Array	是	邀请的用户 ID 列表
observerUserIds	Array	是	被邀请观察者id列表 (只能听或看，不能推流的用户)默认传空数组[]

代码示例

```
CallLib.inviteUsers(userIds,observerUserIds);
```

通话监听

更新时间:2024-08-30

融云 calllib-uni 库提供了如下监听函数，用于处理呼叫相关的业务逻辑上报。

添加监听

1. 监听通话呼入，通过回调 CallLib.onCallReceived 监听。

   代码示例

```
CallLib.onCallReceived( (res)=> {
console.log("Engine:OnCallReceived=>"+"监听通话呼入, 目标id=>", res.data.targetId);
});
```

2. 开始呼叫通话,电话已拨出，通过回调 CallLib.onCallOutgoing 监听。

   代码示例

```
CallLib.onCallOutgoing((res)=>{
console.log("主叫端拨出电话后，通过回调 onCallOutgoing，通知当前 call 的详细信息",res)
})
```

3. 已建立通话，通过回调 CallLib.onCallConnected 监听。

   代码示例

```
CallLib.onCallConnected((res)=>{
console.log("Engine:OnCallConnected=>"+"通话接通时，通过回调 onCallConnected 通知当前 call 的详细信息", res)
});
```

4. 通话结束，通过回调 CallLib.OnCallDisconnected 监听。reason 表示挂断原因。具体请参见挂断通话原因。

   代码示例

```
CallLib.onCallDisconnected((res)=>{
console.log("Engine:OnCallDisconnected=>"+"挂断成功, 挂断原因=>", res.data.reason)
})
```

5. 被叫端正在振铃，通过回调 CallLib.onRemoteUserRinging 监听。

   代码示例

```
CallLib.onRemoteUserRinging((res)=>{
console.log("主叫端拨出电话，被叫端收到请求，发出振铃响应时触发，对端Id为=>", res.data.userId)
})
```

6. 对端用户加入了通话，通过回调 CallLib.onRemoteUserJoined 监听。

   代码示例

```
CallLib.onRemoteUserJoined((res)=>{
console.log("Engine:OnRemoteUserJoined=>"+"主叫端拨出电话，被叫端收到请求后，加入通话，被叫端Id为=>",
res.data.userId);
})
```

7. 通话中的某一个参与者，邀请好友加入通话，通过回调 CallLib.OnRemoteUserInvited 监听。

   代码示例

```
CallLib.onRemoteUserInvited((res)=>{
console.log("Engine:OnRemoteUserInvited=>"+"通话中的某一个参与者，邀请好友加入通话，发出邀请请求后,远端Id为=>",
res.data.userId)
})
```

8. 通话中的远端参与者离开，通过回调 CallLib.OnRemoteUserLeft 监听。reason 表示挂断原因。具体请参见挂断通话原因。

   代码示例

```
CallLib.onRemoteUserLeft((res)=>{
console.log("Engine:OnRemoteUserLeft=>"+"远端用户挂断，远端Id为=>", res.data.reason)
})
```

9. 通话中某一个参与者切换通话类型，通过回调 CallLib.onRemoteUserMediaTypeChanged 监听。

代码示例

```
CallLib.onRemoteUserMediaTypeChanged((res)=>{
console.log("当通话中的某一个参与者切换通话类型，例如由 audio 切换至 video，回调 onRemoteUserMediaTypeChanged,
切换媒体类型的Id为=>",res.data.user.userId);
})
```

10. 通话过程中发生异常，通过回调 CallLib.onError 监听。reason原因详情

代码示例

```
CallLib.onError((res)=>{
console.log("通话过程中，发生异常，异常原因=>",res.data.reason)
})
```

移除监听

1. 移除监听-接收到通话呼入。

代码示例

```
CallLib.removeCallReceivedListener()
```

2. 移除监听-开始呼叫通话的回调。

代码示例

```
CallLib.removeCallOutgoingListener()
```

3. 移除监听-通话已接通。

代码示例

```
CallLib.removeCallReceivedListener()
```

4. 移除监听-通话已结束。

   代码示例

```
CallLib.removeCallDisconnectedListener()
```

5. 移除监听-对端用户正在振铃。

   代码示例

```
CallLib.removeRemoteUserRingingListener()
```

6. 移除监听-对端用户加入了通话。

   代码示例

```
CallLib.removeRemoteUserJoinedListener()
```

7. 移除监听-有用户被邀请加入通话。

   代码示例

```
CallLib.removeRemoteUserInvited()
```

8. 移除监听-对端用户挂断。

   代码示例

```
CallLib.removeRemoteUserLeftListener()
```

9. **移除监听-对端用户切换了媒体类型。**

   代码示例

```
CallLib.removeRemoteUserMediaTypeChangedListener()
```

10. **移除监听-通话出现错误的回调。**

    代码示例

```
CallLib.removeErrorListener()
```

挂断通话原因

更新时间:2024-08-30

状态码	平台	说明
0	Android/iOS	己方取消已发出的通话请求
1	Android/iOS	己方拒绝收到的通话请求
2	Android/iOS	己方挂断
3	Android/iOS	己方忙碌
4	Android/iOS	己方未接听
5	Android/iOS	己方不支持当前引擎
6	Android/iOS	己方网络出错
7	iOS	己方获取媒体资源失败
8	iOS	己方发布资源失败
9	iOS	己方订阅资源失败
10	Android/iOS	对方取消已发出的通话请求
11	Android/iOS	对方拒绝收到的通话请求
12	Android/iOS	通话过程对方挂断
13	Android/iOS	对方忙碌
14	Android/iOS	对方未接听

状态码	平台	说明
15	Android/iOS	对方不支持当前引擎
16	Android/iOS	对方网络错误
17	iOS	对方获取媒体资源失败
18	iOS	对方发布资源失败
19	iOS	对方订阅资源失败
20	iOS	己方其他端已加入新通话
21	iOS	己方其他端已在通话中
22	Android/iOS	己方已被禁止通话
23	iOS	对方其他端已加入新通话
24	iOS	对方其他端已在通话中
25	iOS	对方已被禁止通话
26	iOS	己方其他端已接听
27	iOS	己方其他端已挂断
28	Android/iOS	己方被对方加入黑名单
29	Android/iOS	音视频服务已关闭
30	iOS	己方被降级为观察者
31	Android	己方摄像头初始化错误，可能是没有打开使用摄像头权限
32	Android	其他端已经接听

状态码	平台	说明
33	Android	im ipc服务已断开

通话出现错误原因

更新时间:2024-08-30

状态码	平台	说明
0	Android/iOS	成功
1	Android	开通的音视频服务没有及时生效或音视频服务已关闭
2	iOS	网络不可用
3	iOS	已经处于通话中了
4	iOS	无效操作
5	iOS	参数错误
6	iOS	网络不稳定
7	iOS	媒体服务请求失败
8	iOS	媒体服务初始化失败
9	iOS	媒体服务未初始化
10	iOS	媒体服务请求超时
11	iOS	未知的媒体服务错误
12	iOS	已被禁止通话
13	iOS	音视频服务已关闭
14	iOS	音视频发布资源失败

状态码	平台	说明
15	iOS	音视频订阅资源失败
16	iOS	其他端已在通话中错误

通话信息 获取当前呼叫成员信息

更新时间:2024-08-30

单聊/群聊通过 `CallLib.getCurrentCallSession` 获取当前呼叫 session 里的成员信息。

代码示例

```
let currentCallSession = CallLib.getCurrentCallSession()
let users = currentCallSession.users
```

摄像头设置　　开关摄像头

更新时间:2024-08-30

在通话建立之后打开/关闭摄像头。

```
CallLib.enableCamera(isOpen,RCCallIWCamera)
```

参数	类型	必填	说明
isOpen	Boolean	是	是否开启摄像头
RCCallIWCamera	Number	是	0 未指定 1 前置相机 2 后置相机

切换前后置摄像头

在通话建立之后切换前后置摄像头，该方法适用于通过 SDK 打开 **默认摄像头** 的场景，视频默认打开前置摄像头。

```
CallLib.switchCamera();
```

设置本地摄像头预览视图是否镜像显示

在通话前，使用 CallLib 中的 `setVideoConfig` 方法可以设置本地摄像头视频预览视图是否镜像显示。注意，该方法不影响远端的显示。

```
// isPreviewMirror true：镜像 false: 非镜像
CallLib.setVideoConfig({isPreviewMirror: true});
```

美颜处理

更新时间:2024-08-30

本文描述如何在融云音视频 SDK 基础上实现美颜功能。

官方美颜插件

您可以使用融云官方提供的基础美颜插件。

步骤 1：集成插件

1. 从 uni-app 插件市场安装 RCUniBeauty 原生插件到项目中。
2. 原生插件配置完成后，还需要从 uni-app 插件市场安装 `RongCloud-BeautyWrapper` 到项目中，这个插件封装了提供给 js 层的所有接口。
3. 导入插件：

```
// RCBeautyEngine
import RCBeautyEngine from "@/uni_modules/RongCloud-BeautyWrapper/lib/RCBeautyEngine"
```

步骤 2：使用插件

打开/关闭美颜，并设置美颜参数

```
let options = {
whitenessLevel: 0,
ruddyLevel: 0,
smoothLevel: 0,
brightLevel: 5
}
RCBeautyEngine.setBeautyOptions(true, options);
```

参数	类型	说明
enable	Boolean	true :打开，false :关闭。默认 false
option	Object	设置美颜参数（美白，磨皮，红润，亮度）{whitenessLevel:0-9,smoothLevel:0-9,ruddyLevel:0-9,brightLevel:0-9}

获取当前设置的美颜参数

```
let beautyOptions = RCBeautyEngine.getCurrentBeautyOptions();
```

设置美颜滤镜

```
RCBeautyEngine.setBeautyFilter(0);
```

参数	类型	说明
filter	Number	0 无美颜滤镜 1 唯美 2 清新 3 浪漫

获取当前设置的滤镜

```
let filter = RCBeautyEngine.getCurrentBeautyFilter();
```

重置美颜参数

```
RCBeautyEngine.resetBeauty();
```

相芯美颜插件

> 提示
>
> 使用相芯美颜需要购买相关授权，详情请咨询融云商务。插件使用方法详见 RCUniFUBeauty 。

音频管理 麦克风设置

更新时间:2024-08-30

关闭/打开麦克风，默认处于打开状态。

```
CallLib.enableMicrophone(enabled)
```

参数	类型	说明
enabled	Boolean	true :打开 false :关闭。默认 true

扬声器设置

设置是否打开扬声器。

```
CallLib.enableSpeaker(enable)
```

参数	类型	说明
enabled	Boolean	true :打开，false :关闭。默认 false

更新日志 5.1.15

更新时间:2024-08-30

> **发布日期：**2022/01/10

1. **封装融云** CallLib 5.1.15 **版本。**
2. **集成** SDK **时，音视频通话** Typescript **层依赖项从** NPM **改为从** uni-app **插件市场安装。**

5.1.12

> **发布日期：**2021/11/15

1. **封装融云** CallLib 5.1.12 **版本。**

状态码

更新时间:2024-08-30

uni-app 平台 SDK 是对 Android、iOS 平台 SDK 的二次封装。请参考下方列出的 Android、iOS 平台的状态码。

即时通讯（IM）

- Android 状态码
- iOS 状态码

即时通讯（RTC）

- Android 状态码
- iOS 状态码