ALIBABA CLOUD

# 阿里云

## 音视频通信

## 音视频通信公共云合集

文档版本：20220511

阿里云

# 法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

# 通用约定

| 格式 | 说明 | 样例 |
| --- | --- | --- |
| 危险 | 该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 危险<br>重置操作将丢失用户配置数据。 |
| 警告 | 该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 警告<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
| 注意 | 用于警示信息、补充说明等，是用户必须了解的内容。 | 注意<br>权重设置为0，该服务器不会再接受新请求。 |
| 说明 | 用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。 | 说明<br>您也可以通过按Ctrl+A选中全部文件。 |
| > | 多级菜单递进。 | 单击**设置**> **网络**> **设置网络类型**。 |
| **粗体** | 表示按键、菜单、页面名称等UI元素。 | 在**结果确认**页面，单击**确定**。 |
| Courier字体 | 命令或代码。 | 执行 `cd /d C:/window` 命令，进入Windows系统文件夹。 |
| *斜体* | 表示参数、变量。 | `bae log list --instanceid`<br>*Instance_ID* |
| [] 或者 [a\|b] | 表示可选项，至多选择一个。 | `ipconfig [-all\|-t]` |
| {} 或者 {a\|b} | 表示必选项，至多选择一个。 | `switch {active\|stand}` |

# 目录

# 1.场景组件体验

通过阅读本文，您可以了解各场景Demo的下载地址及效果展示，根据实际场景快速体验Demo。

## Demo体验

您可以通过钉钉扫描以下二维码或单击对应的下载地址，下载App进行体验。

| iOS | Android | Mac | Windows | Web |
| --- | --- | --- | --- | --- |
|  | | 互动大班课 | 互动大班课 | 音视频通话<br>超级小班课 |

> **说明**
>
> - 对于Web体验，Windows端目前仅支持Chrome浏览器，Mac端支持Chrome和Safari浏览器。
> - 如果提示**RTC不支持**，请查看是否插入音频设备（麦克风和摄像头）以及浏览器和系统是否禁用音视频。

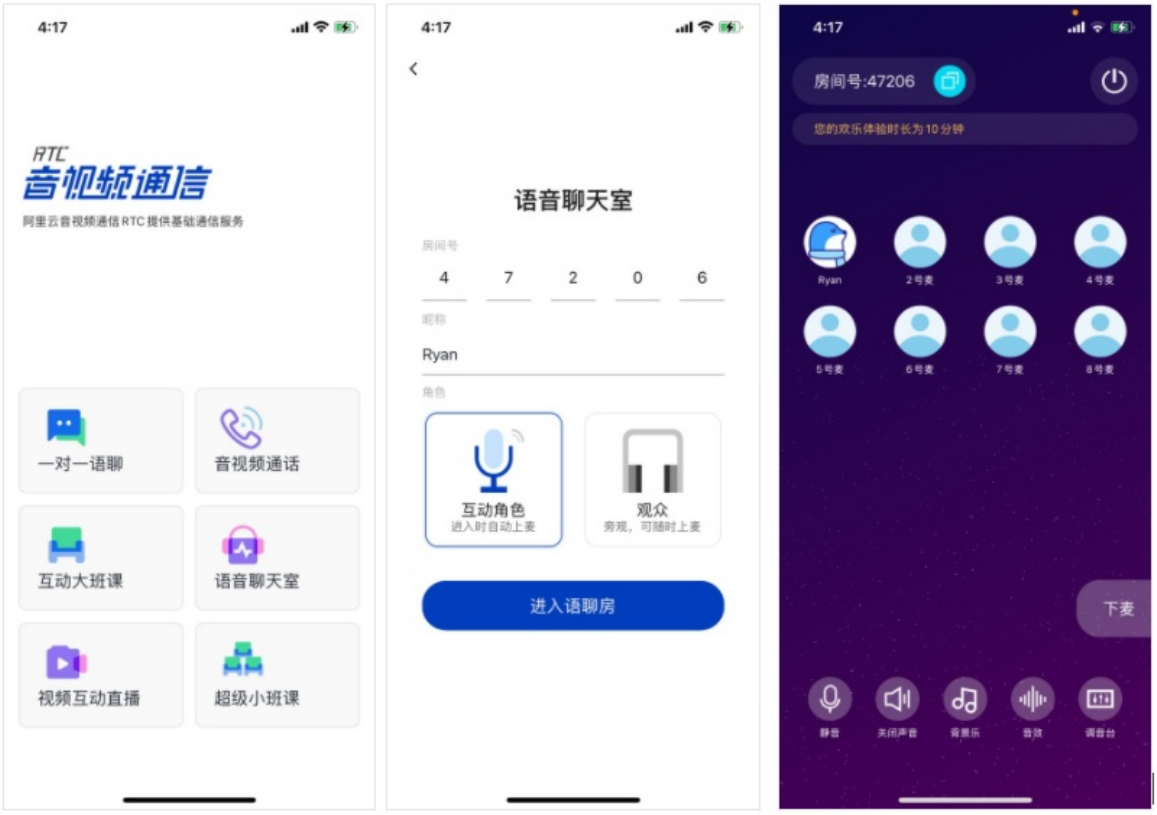
## Demo源码下载

| 场景 | 平台 | 场景简介 | 示例代码 |
| --- | --- | --- | --- |
| 1对1语音聊天 | iOS、Android | 1对1语音聊天简介 | 源码下载 |
| 音视频通话 | iOS、Android、Web | 音视频通话简介 | 源码下载 |
| 互动大班课 | • 学生端：Electron、iOS、Android<br>• 教师端：Electron | 互动大班课简介 | 源码下载 |
| 语音聊天室 | iOS、Android | 语音聊天室简介 | 源码下载 |
| 视频互动直播 | iOS、Android | 视频互动直播简介 | 源码下载 |
| 超级小班课 | • 学生端：iOS、Android<br>• 教师端：Web | 超级小班课简介 | 源码下载 |

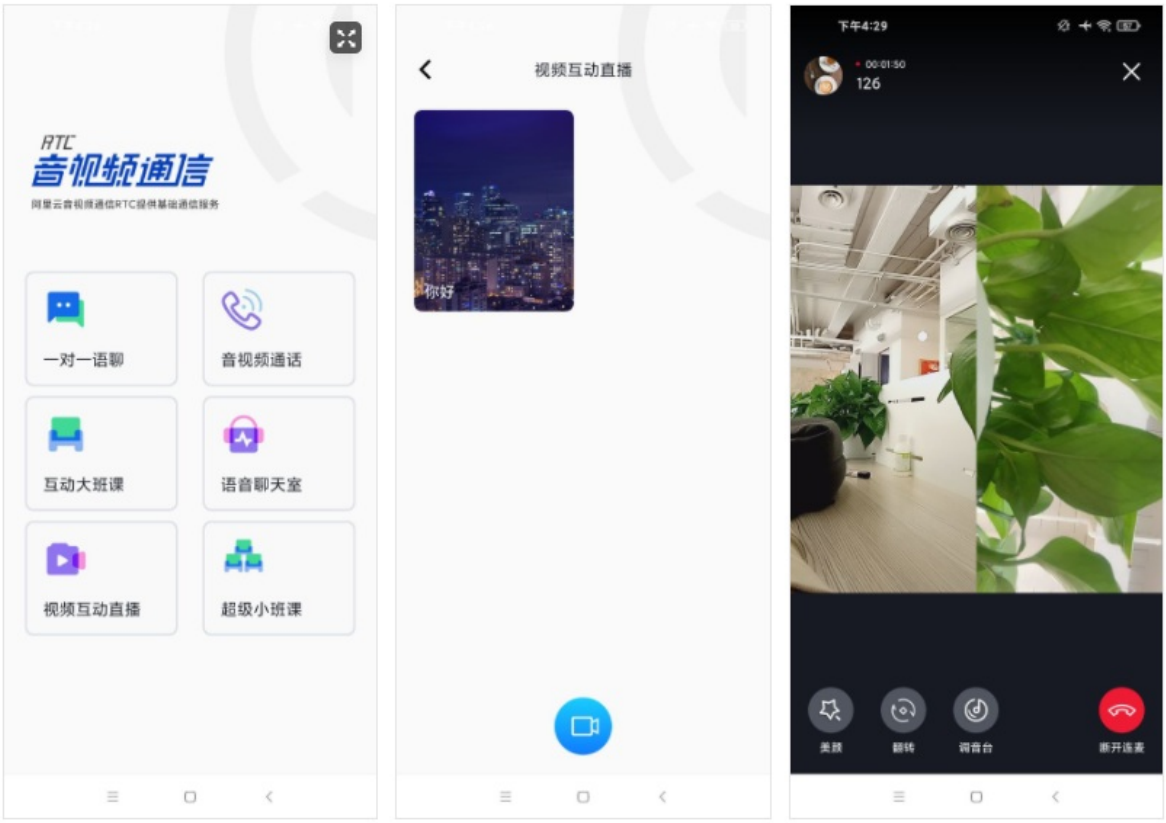
## 界面效果展示

RTC产品体验Demo的效果图如下所示：

- iOS端——语音聊天室

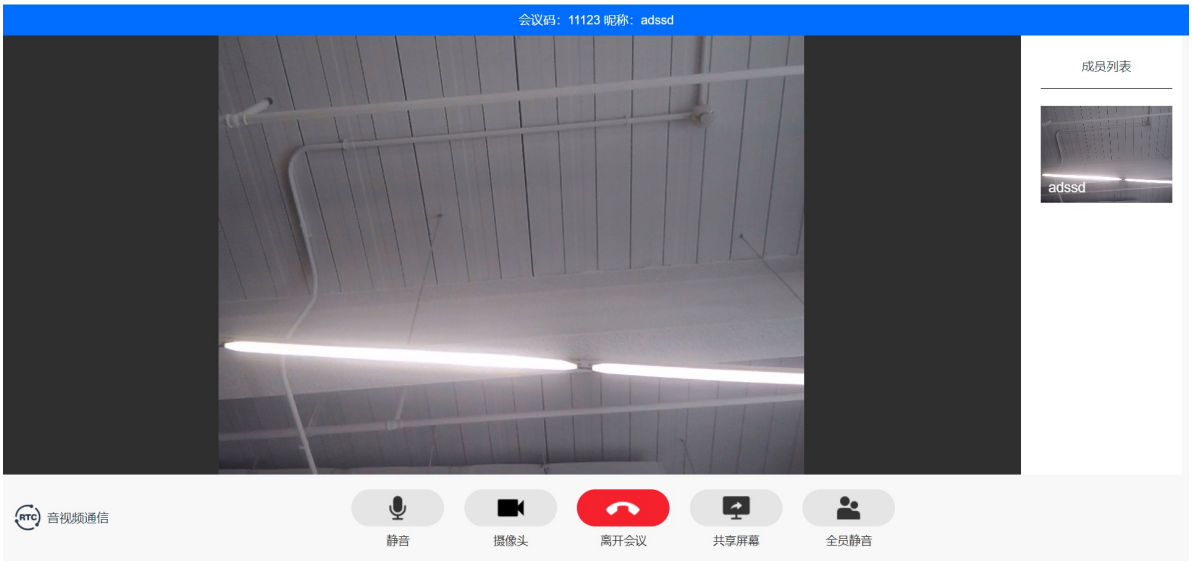



- Android端——视频互动直播




- Web端


音视频通信
234512
请输入会议码
加入
创建会议


会议码：11123 昵称：adssd
成员列表
adssd
音视频通信
静音
摄像头
离开会议
共享屏幕
全员静音


音视频通信
10000人正在观看
互动大班课
教师
学生
您的姓名
教室码
9 6 9 6 2 8
开始上课

# 2.解决方案

## 2.1. 1对1语音聊天

### 2.1.1. 简介

通过阅读本文，您可以快速了解1对1语音聊天基本信息及实现方法。

#### 使用场景

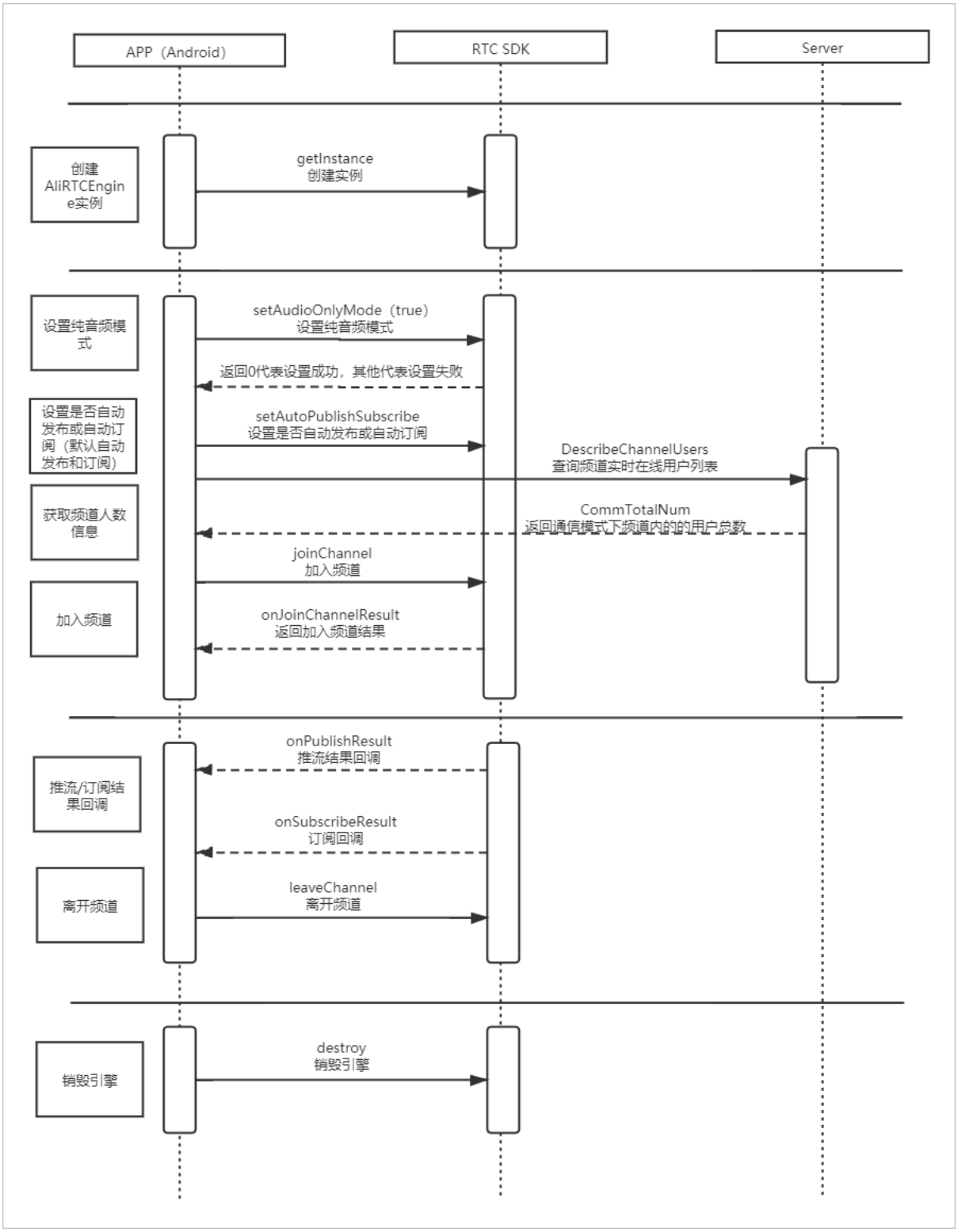
1对1语音聊天是一款双人实时通话产品（不具备视频通话功能），常用于私密聊、语音陪聊等社交场景。拥有实时通话、设置背景音乐等功能。同时能够为开发者提供高音质、低延迟、便捷接入、多平台互通的服务。

#### 主要功能

| 功能 | 描述 |
| --- | --- |
| 实时语音通话 | 超低延时下用户之间实现语音通话功能。 |
| 背景音乐 | 可以播放指定背景音乐。 |
| 伴奏混音 | 将本地或在线的音频和用户声音，同时发送并播放给另一名用户。 |
| 高音质 | 支持48kHz采样的高音质，支持左右声道。 |
| 3A音频处理 | 行业领先的音频3A（AGC、AEC、ANS），支持针对人声、乐器等场景定制化调优。 |

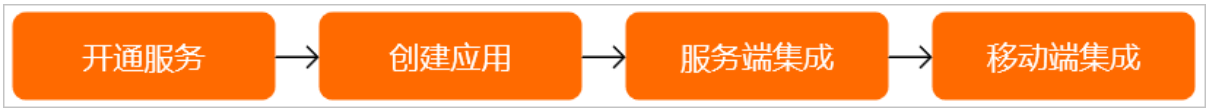
#### 实现方法

创建房间API时序图（Android）：




## 实现流程

实现流程如下图所示：



| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务 | 进行服务端端集成之前，您必须开通音视频通信服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |
| 3 | 服务端集成 | 您可以通过源码快速搭建服务端1对1语音聊天。 |
| 4 | 移动端集成：<br>• iOS端集成<br>• Android端集成 | 您可以通过源码快速搭建移动端1对1语音聊天。 |

### Demo体验

您可以通过钉钉扫描以下二维码，下载安装1对1语音聊天Demo体验。

## 2.1.2. 服务端集成

通过阅读本文，您可以了解1对1语音聊天服务端的集成操作。

### 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 环境中已安装Java JDK 8的版本。具体操作，请参见安装JDK。

> **说明** 如果服务端为Linux环境，推荐安装Oracle JDK，不推荐使用Open JDK进行服务端集成。

### 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。

ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。

iii. 在**AppID/名称**列获取AppID。

iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 获取AccessKey。

> **说明** 由于主账号AccessKey泄露会威胁您所有资源的安全，因此出于安全考虑，需要创建RAM用户（子账号）并获取RAM用户（子账号）的AccessKey，用于访问您的云资源。

i. 登录RAM控制台。

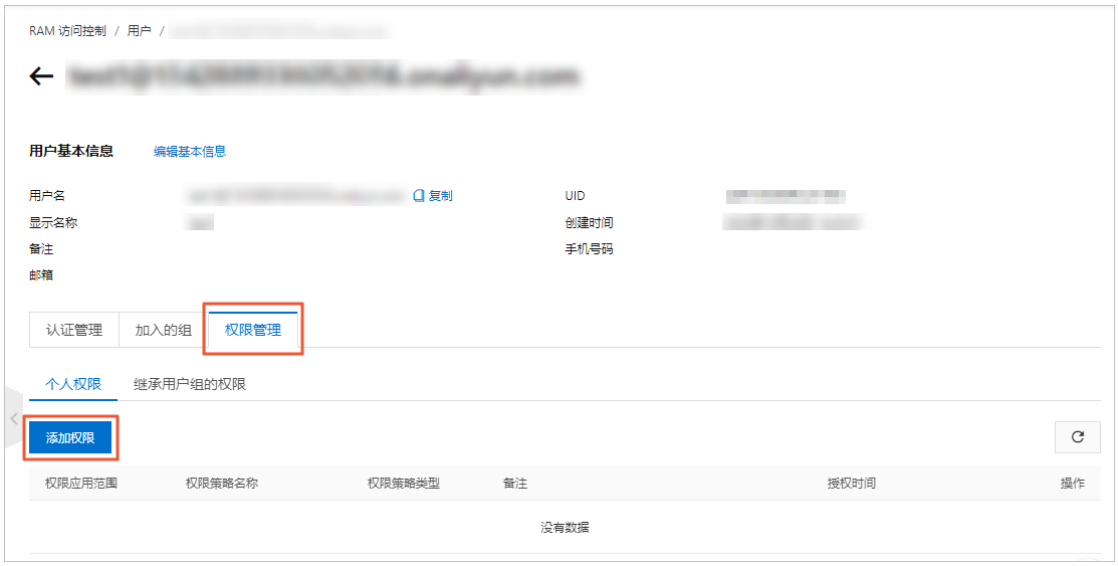
ii. 在左侧导航栏选择**身份管理 > 用户**，进入用户页面。

iii. 单击**创建用户**，填写用户账号信息，选中**Open API 调用访问**创建用户。

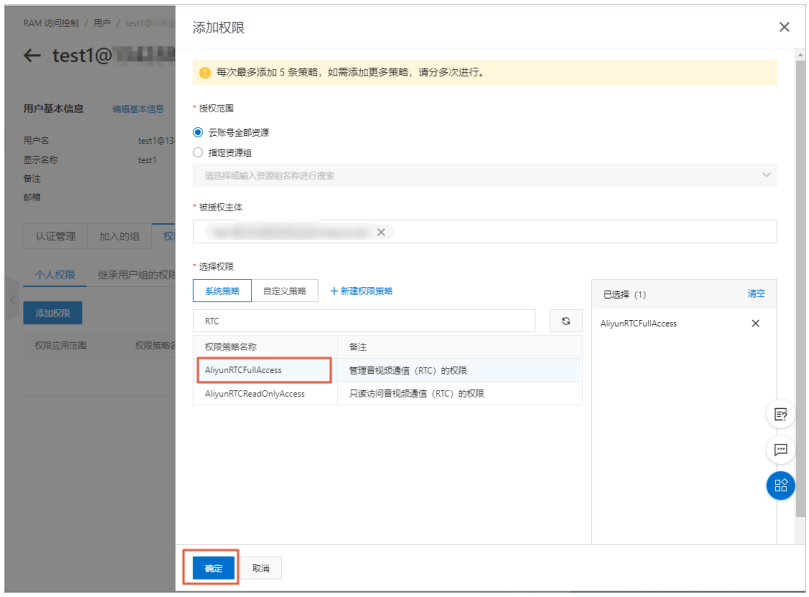

iv. 单击**确定**。

v. 重新返回用户页面。在**用户登录名称/显示名称**列下，单击目标RAM子账户，进入管理页面。

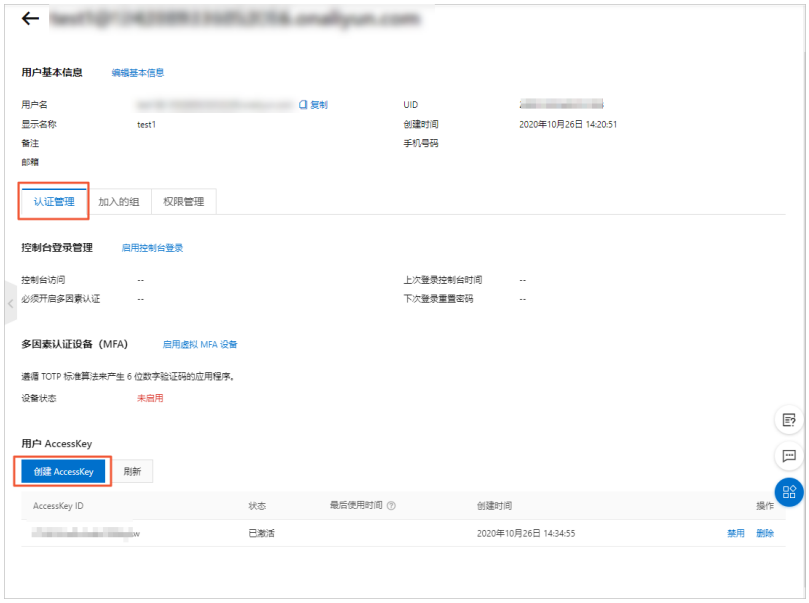
vi. 单击**权限管理**，再单击**添加权限**。




vii. 选择AliyunRTCFullAccess（管理音视频通信的权限，可输入RTC进行搜索）。单击**确定**。



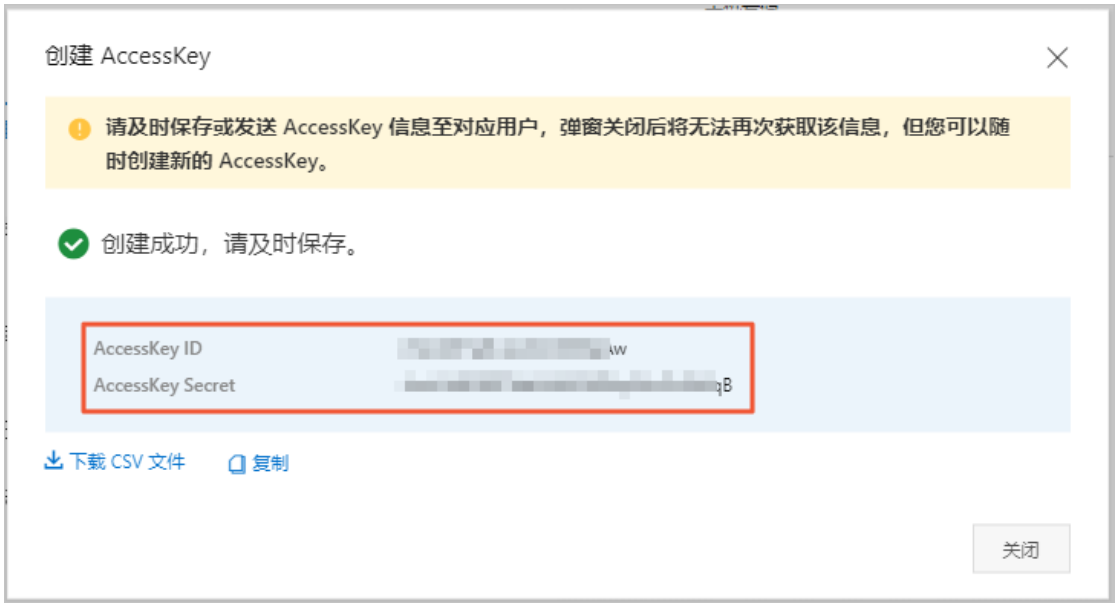
viii. 单击**认证管理**，在用户AccessKey区域单击**创建AccessKey**。




> **说明**
>
> - 首次创建时需填写手机验证码。
> - 如果AccessKey泄露或丢失，则需要创建新的AccessKey，最多可以创建2个AccessKey。

ix. 获取AccessKey ID和AccessKey Secret。

创建AccessKey完成后，会弹出**创建AccessKey**对话框，包含AccessKey ID和AccessKey Secret信息。此处建议记录一下AccessKey ID和AccessKey Secret，方便后续操作中使用。

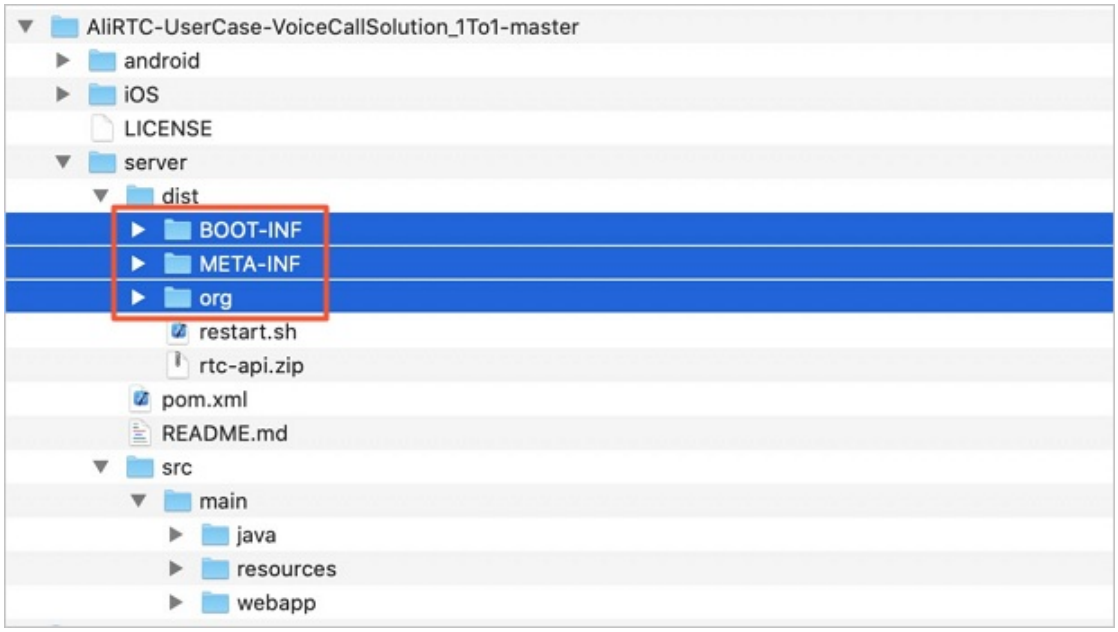



3. 下载并解压Demo，更多信息，请参见Demo源码下载。

> 说明
> - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

4. 配置Demo工程。
   i. 将*server/dist/rtc-api.zip*文件解压到*dist*文件夹下。

   解压成功之后如下图所示：




> 说明 解压成功后，可以看到BOOT-INF、META-INF及org文件夹，这三个文件夹需要和restart.sh保持在同一个目录下。

ii. 修改配置文件。

打开*BOOT-INF/classes/application.properties*文件，修改配置。

> **说明** 使用文本编辑器打开即可，若找不到打开方式，推荐安装VSCode等轻量级代码编辑器打开。

```
#################################################log###################
logging.level.com.live.dao=debug
logging.level.com.live.dao.user=debug
logging.level.com.alivc.vod.dao=debug
logging.level.com.alivc.longVideo.dao=debug
logging.file: my.log
logging.pattern.file= %d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %l
logging.pattern.console= %d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread]


rtc.1v1audio.appId = *****
rtc.1v1audio.appKey = *****
rtc.gslb = https://rgslb.rtc.aliyuncs.com

accessKeyID=*****
accessKeySecret=*****

roleName=alivc-demo-role


durationlimit=-1
```

- 设置RTC应用AppId和AppKey，请参见步骤1获取。

  rtc.1v1audio.appId = *

  rtc.1v1audio.appKey = *

- 设置AK，需要添加AliyunRTCFullAccess权限，请参见步骤2获取。

  accessKeyID=*

  accessKeySecret=*

5. 运行服务，执行*restart.sh*文件。如果编译过程中出现问题，请参见服务端运行常见问题。
   - Mac或Linux环境下请将终端定位至*dist*目录下，执行如下命令：

     ```
     sh restart.sh
     ```

     后台执行可以使用如下命令以确保退出终端时程序能够继续运行。

     ```
     nohup sh restart.sh >./run_log.log 2>&1 &
     ```

   - Windows环境需要打开CMD终端定位到*server/dist*文件夹下，执行如下命令：

     ```
     java org.springframework.boot.loader.JarLauncher
     ```

     若提示8080端口被占用，请尝试使用netstat命令查看占用8080端口的进程pid号，并使用taskkill关闭相关进程。

     ```
     netstat -ano | findstr 8080
     taskkill /pid 占用端口的进程pid号 /f
     ```

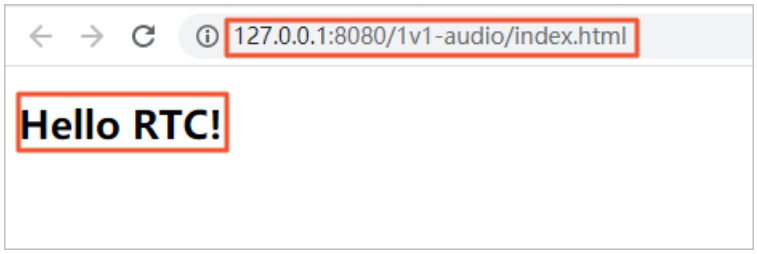
```
Microsoft Windows [版本 10.0.18363.1139]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\[redacted]>netstat -ano | findstr 8080
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING       23536
  TCP    [::]:8080              [::]:0                 LISTENING       23536

C:\[redacted]>taskkill /pid 23536 /f
成功: 已终止 PID 为 23536 的进程。

C:\[redacted]>
```

终端成功运行后可以看到服务端启动成功的日志信息。

```
tializing ExecutorService 'taskExecutor'
2020-10月-26 17:53:39.877 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapte
r.initControllerAdviceCache - Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigE
mbeddedWebApplicationContext@7ec7ffd3: startup date [Mon Oct 26 17:53:36 CST 2020]; root of context hierarchy
2020-10月-26 17:53:40.054 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappir
g.register - Mapped "{[/app/descChannelStartTime],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rt
c.controller.RtcController.descChannelStartTime(java.lang.String)
2020-10月-26 17:53:40.056 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappir
g.register - Mapped "{[/app/token],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rtc.controller.Rt
cController.getRtcToken(java.lang.String)
2020-10月-26 17:53:40.057 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappir
g.register - Mapped "{[/app/descChannelUsers],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rtc.co
ntroller.RtcController.descChannelUsers(java.lang.String)
2020-10月-26 17:53:40.060 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappir
g.register - Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, jav
a.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequ
est)
2020-10月-26 17:53:40.065 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappir
g.register - Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.sprin
gframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest, javax.servlet.htt
p.HttpServletResponse)
2020-10月-26 17:53:40.111 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
 Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestH
andler]
2020-10月-26 17:53:40.112 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
 Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2020-10月-26 17:53:40.173 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
 Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequ
estHandler]
2020-10月-26 17:53:40.211 INFO [main] org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration$WelcomePageHar
dlerMapping.<init> - Adding welcome page: class path resource [resources/index.html]
2020-10月-26 17:53:40.503 INFO [main] org.springframework.jmx.export.annotation.AnnotationMBeanExporter.afterSingletons
Instantiated - Registering beans for JMX exposure on startup
2020-10月-26 17:53:40.623 INFO [main] org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.s
tart - Tomcat started on port(s): 8080 (http)
2020-10月-26 17:53:40.632 INFO [main] com.alivc.Application.logStarted - Started Application in 5.391 seconds (JVM runn
ing for 6.906)
2020-10月-26 17:53:40.633 INFO [main] com.alivc.Application.main - start success
```

6. 检查服务端是否已经启动。

   在浏览器中访问*http://127.0.0.1:8080/1v1-audio/index.html*，如果显示如下图所示，表示服务端已经启动。




## 主要功能说明

- 生成joinChannel所需鉴权信息。

  访问地址：*/app/token*

客户端调用RTC SDK加入房间的Token信息就是从这个接口获得。具体生成方式，请参见RTC帮助文档操作步骤。

```
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        digest.update(appId.getBytes());
        digest.update(appKey.getBytes());
        digest.update(channelId.getBytes());
        digest.update(userId.getBytes());
        digest.update(nonce.getBytes());
        digest.update(Long.toString(timestamp).getBytes());
        String token = DatatypeConverter.printHexBinary(digest.digest()).toLowerCase();
        return token;
```

- 查询频道实时在线用户列表。

  访问地址：*/app/descChannelUsers*

  AppServer通过调用DescribeChannelUsers接口查询频道的实时在线人数。

```
        DefaultAcsClient client = initVodClient();
        DescribeChannelUsersRequest request = new DescribeChannelUsersRequest();
        request.setAppId(appId);
        request.setChannelId(channelId);
        DescribeChannelUsersResponse response = client.getAcsResponse(request);
```

- 查询房间开启时间。

  访问地址：*/app/descChannelStartTime*

  当客户端请求加入房间所需的token时，服务端查询当前房间人数，若房间人数为0，保存当前时刻和房间id。并把当前时刻作为此房间的创建时间。

```
            DescribeChannelUsersResponse response = RtcOpenAPI.describeChannelUsers(appId
, channelId);
            if (CollectionUtils.isEmpty(response.getUserList())) {
                ImmutablePair<String, String> appChannel = new ImmutablePair<>(appId, cha
nnelId);
                ScheduledFuture scheduledFuture = TASKS.getOrDefault(appChannel, new JSON
Object()).getObject("scheduledFuture", ScheduledFuture.class);
                TASKS.put(appChannel, channelInfo);
            }
                        JSONObject scheduledTask = ScheduledDeleteChannel.TASKS.get(Immut
ablePair.of(appId, channelId));
```

# 2.1.3. Android端集成

通过阅读本文，您可以了解1对1语音聊天Android端的集成操作。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

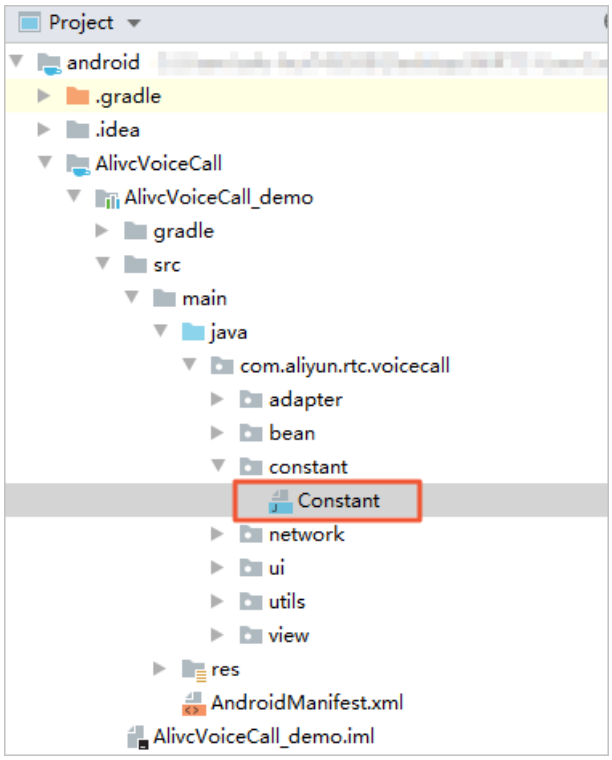
- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Android Studio 3.0或以上版本。更多信息，请参见Android Studio。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

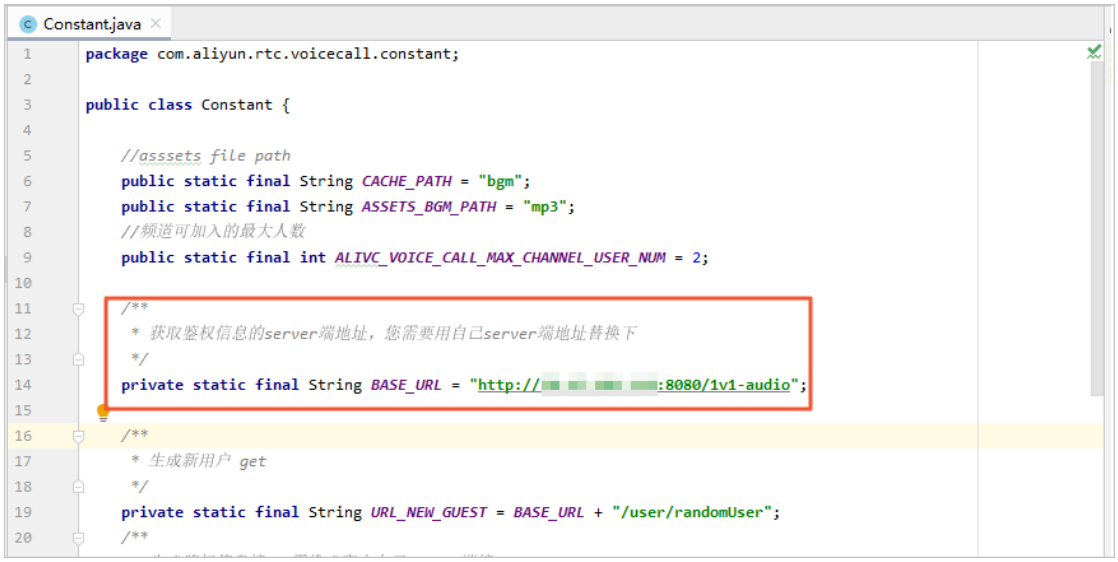
   > 说明
   > - Demo源码中已通过Maven方式集成AliRTC Android SDK（1.17版本）。
   > - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 配置Demo工程。
   i. 打开Android Studio，单击**Open an Existing Project**，选择*android*文件夹。
   ii. 打开*android/AlivcVoiceCall/AlivcVoiceCall_demo/src/main/java/com/aliyun/rtc/voicecall/constant/Constant.java*文件。



iii. 修改文件中的 `BASE_URL` 值。

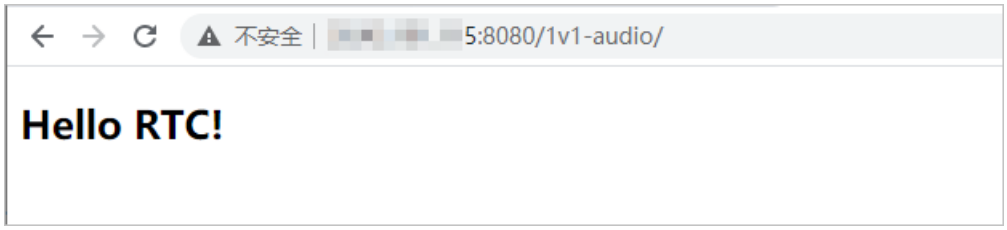
例如本地服务端IP地址为192.0.2.1，则 `BASE_URL` 的值为 `http://192.0.2.1:8080/1v1-audio` ，即 `BASE_URL="http://192.0.2.1:8080/1v1-audio"` 。本地服务端IP地址查询，请参见查询IP地址。




> **说明**
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `BASE_URL="http://<域名>/1v1-audio"` 。

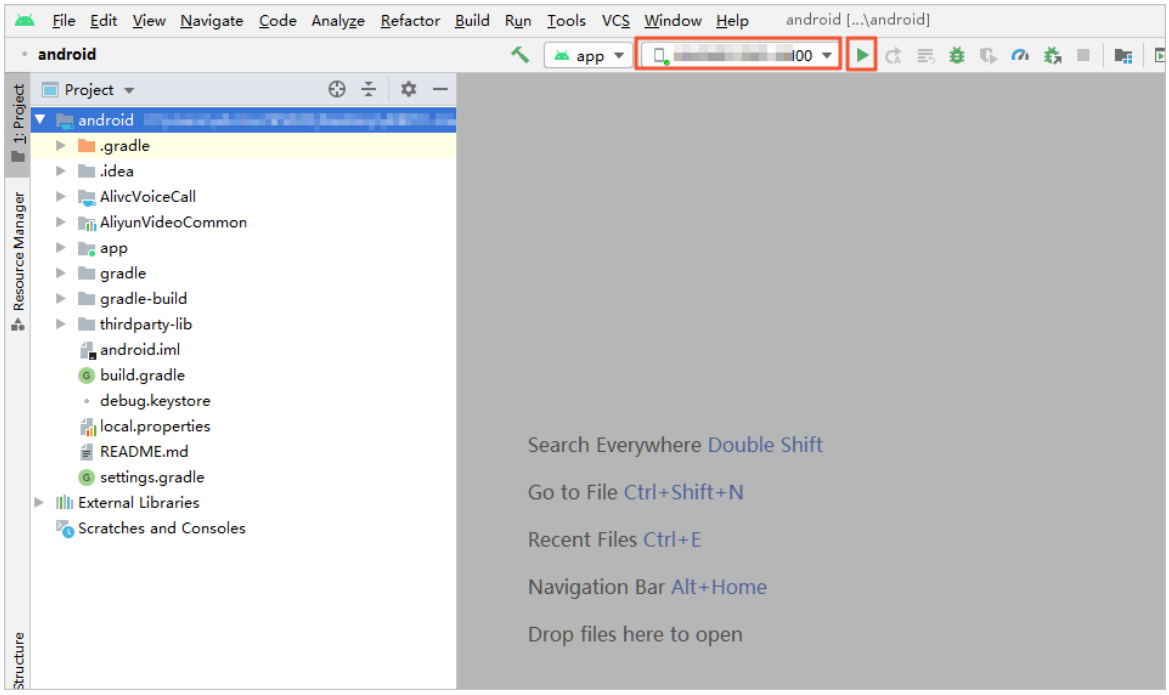
iv. 验证移动端和服务端。

分别在移动端和服务端浏览器中访问 `BASE_URL` 地址，如果显示如下图所示，表示移动端访问服务端正常。




3. 运行Demo。

将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。




> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

4. 进行1对1语音聊天。
   i. 将两台移动端设备安装Demo App。
   ii. 将两台设备都连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间号，创建房间并等待对方加入。
   iv. 在第二台设备上输入相同房间号加入房间并进行通话。

## Demo目录结构说明

项目结构如下所示：

| 文件名 | 说明 |
| --- | --- |
| AlivcVoiceCall | 1队1语音聊天功能实现库。更多信息，请参见AlivcVoiceCall组件库目录说明。 |
| AliyunVideoCommon | 公共组建库。 |
| app | 程序入口。 |
| thirdparty-lib | 第三方库的引用。 |

AlivcVoiceCall组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| constant | 常量数据管理类。 |

| 文件名 | 说明 |
| --- | --- |
| network | 网络请求。 |
| ui | 应用界面。 |
| view | 自定义view。 |

## 主要功能说明

- 创建并加入房间。

  > 注意 AliRtcEngieEventListener回调在子线程中，如果您想操作UI界面需要切换到主线程。

```
        //创建AliRtcEngin实例。
        mEngine = AliRtcEngine.getInstance(getApplicationContext());
        //设置AliRtcAuthInfo用户信息，通过Server Api获取到用户信息和RTC服务器的Token信息，拿到信息
后
        //设置给AliRtcAuthInfo就可以调用joinChannel加入频道进行通话。
        AliRtcAuthInfo userInfo = "从server端获取";
        //调用joinChannel后在AliRtcEngieEventListener.onJoinChannelResult(int i)接受回调信息。
        if (i != 0) {
            //加入房间失败。
        } else {
            //加入房间成功。
        }
```

- 用户推流。

  当用户加入房间成功时可以调用推流方法来发布自己的音频信息让对方订阅， 也可以在加入房间之前调用 `mEngine.setAutoPublish(true, true);` 来实现主动推流和订阅。

```
//true表示允许发布音频流，false表示不允许。
mEngine.configLocalAudioPublish(true);
mEngine.publish();
```

- 播放伴奏。

  startAudioAccompany方法只能播放一首歌，不支持多首同时播放 playAudioEffect可以播放之前预加载的音效。两个方法可以同时调用，可以一边推送播放背景音效，一个本地试听。

```
//播放当前音效并推送。loopCycles=-1时代表循环播放。
mEngine.startAudioAccompany(mSelectedBgmData.first.getPath(), false, false, -1);
int i = mEngine.playAudioEffect(currBgm, file.getPath(), -1, false);
```

- 暂停伴奏。

```
mEngine.pauseAudioAccompany();
mEngine.pauseAudioEffect(soundId);
```

- 停止伴奏。

```
mEngine.pauseAudioAccompany();
mEngine.stopAudioEffect(currBgm);
```

- 恢复播放伴奏。

```
mEngine.resumeAudioAccompany();
mEngine.resumeAudioEffect(currBgm);
```

- 静音模式（停止发布本地音频流）。

```
mEngine.muteLocalMic(false);
```

- 取消静音模式（发布本地音频流）。

```
mEngine.muteLocalMic(true);
```

- 扬声器模式。

```
mEngine.enableSpeakerphone(true);
```

- 取消扬声器模式。

```
mEngine.enableSpeakerphone(false);
```

- 离开房间。

```
mEngine.leaveChannel();
```

# 2.1.4. iOS端集成

通过阅读本文，您可以了解1对1语音聊天iOS端的集成操作。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

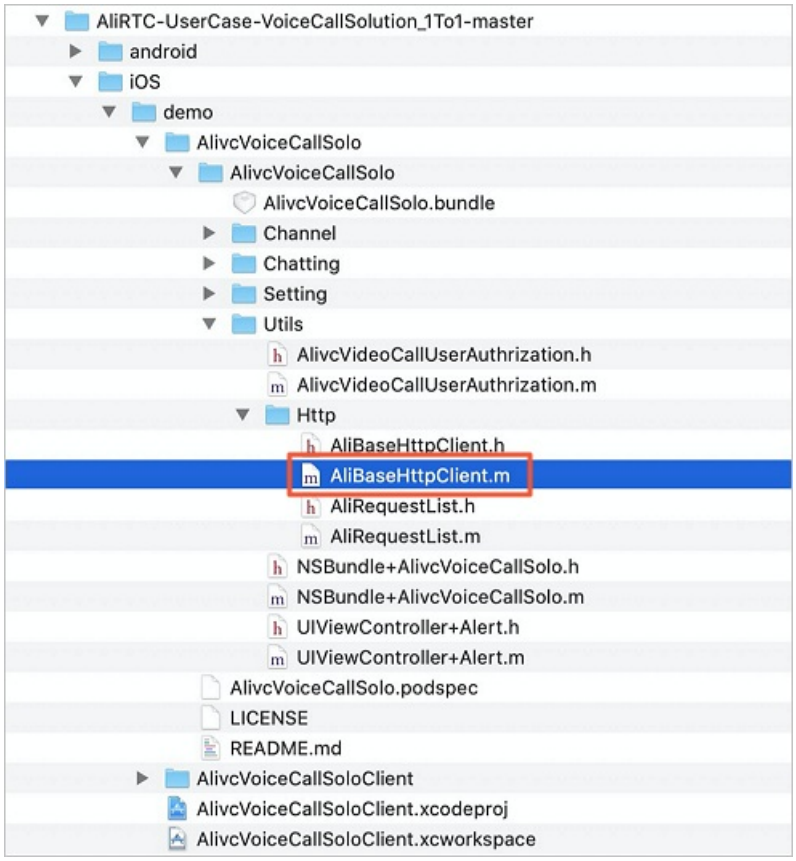
## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > **说明**
   > - Demo源码中已通过Pod方式集成AliRTC iOS SDK（1.17版本）。
   > - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 配置Demo工程。

i. 打开*iOS/demo/AlivcVoiceCallSolo/AlivcVoiceCallSolo/Utils/Http/AliBaseHttpClient.m*文件。



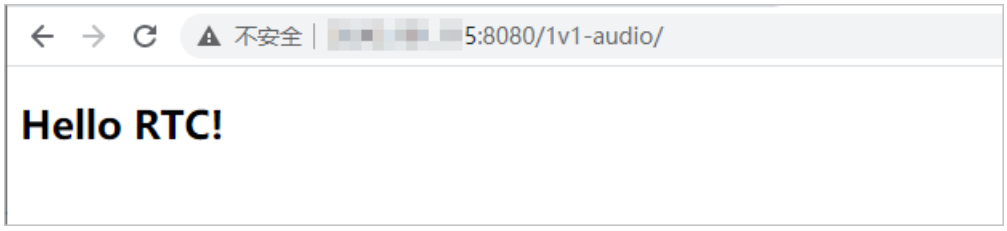

ii. 修改文件中的 `AppServer` 值。

例如本地服务端IP地址为192.0.2.1，则 `AppServer` 的值为 `http://192.0.2.1:8080/1v1-audio` 。本地服务端IP地址查询，请参见查询IP地址。

> **说明**
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `AppServer` 的值为 `http://<域名>/1v1-audio` 。

iii. 验证移动端和服务端。

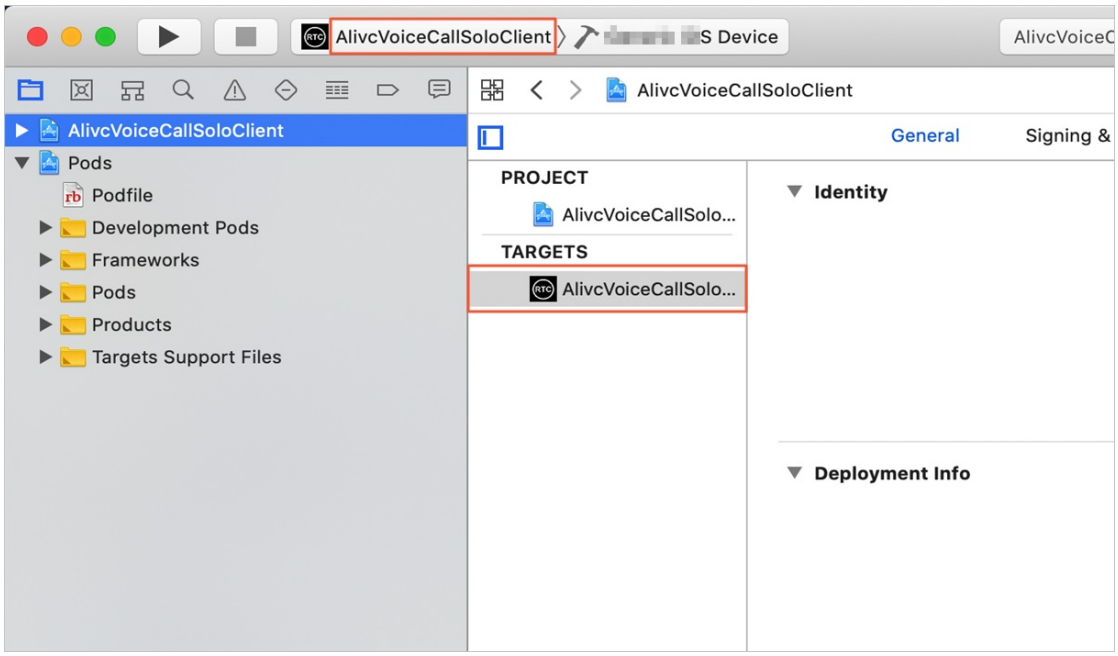
分别在移动端和服务端浏览器中访问 `AppServer` 地址，如果显示如下图所示，表示移动端访问服务端正常。




3. 运行Demo。

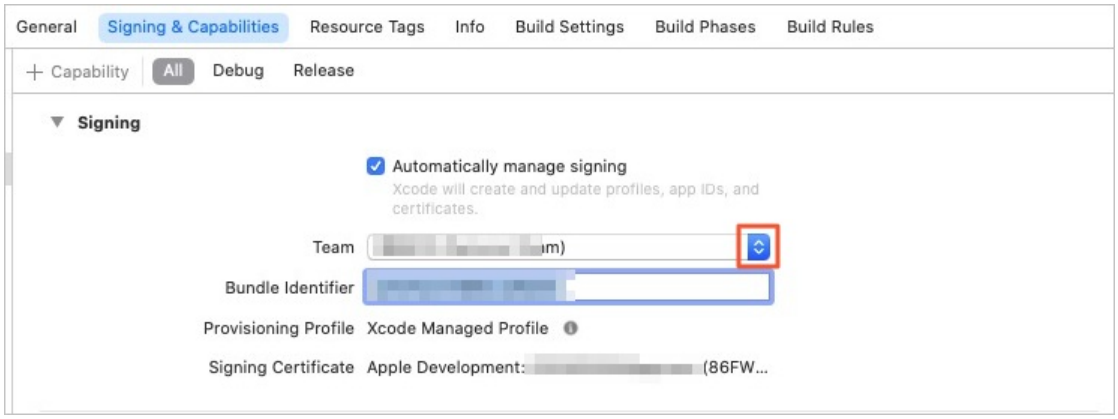
i. 使用Xcode打开*demo*目录下的*AlivcVoiceCallSoloClient.xcworkspace*工程文件。

ii. 选择运行的Target为AlivcVoiceCallSoloClient，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。



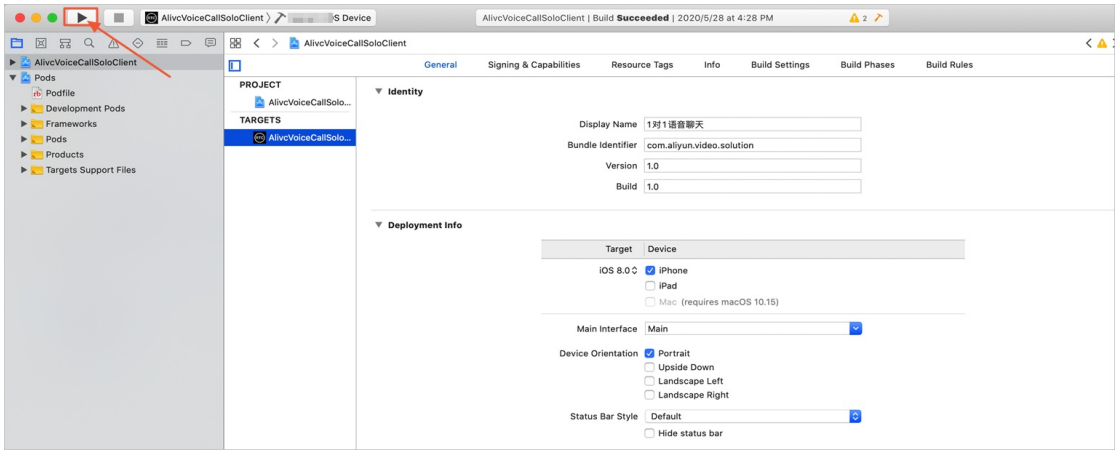

iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。


iv. 单击**Signing & Capabilities**页签，选中**Automatically manage signing**，然后单击**Team**下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择**Add an Account...**，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击 ▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。



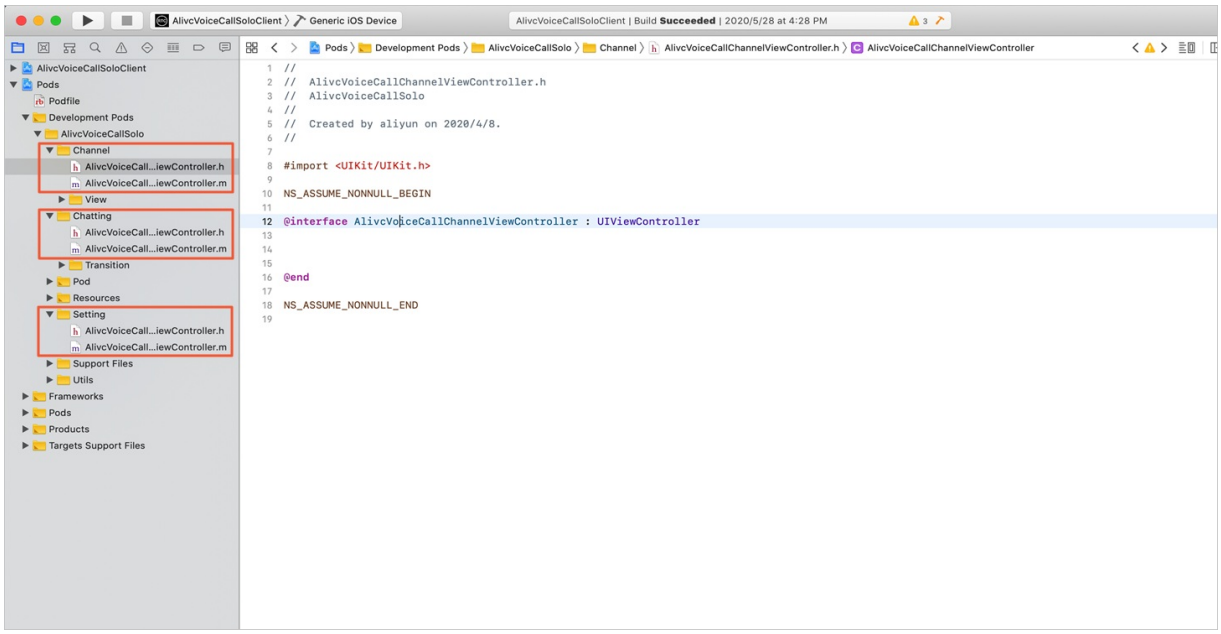

4. 进行1对1语音聊天。
   i. 将两台移动端设备安装Demo App。
   ii. 将两台设备都连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间号，创建房间并等待对方加入。
   iv. 在第二台设备上输入相同房间号加入房间并进行通话。

## Demo目录结构说明

RTC把开发的业务代码封装到AlivcVoiceCallSolo库中，因此只需在Podfile中指定AlivcVoiceCallSolo库的路径，AlivcVoiceCallSolo就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
pod 'AlivcVoiceCallSolo', :path => './AlivcVoiceCallSolo'
##   pod 'AlivcVoiceCallSolo'  说明项目依赖AlivcVoiceCallSolo库
##   path => './AlivcVoiceCallSolo'  指明AlivcVoiceCallSolo库的位置（相对于Podfile）
```

AlivcVoiceCallSolo组件库目录说明，如下所示：



| 文件名 | 说明 |
| --- | --- |
| AlivcVoiceCallSolo.podspec | 组件的描述文件。 |
| AlivcVoiceCallSolo.bundle | 存放资源的bundle。 |
| AlivcVoiceCallChannelViewController | 首页。 |
| AlivcVoiceCallChattingViewController | 聊天页。 |
| AlivcDismissTransition | 转场动画。 |
| AlivcPanInteractiveTransition | |
| AlivcPresentTransition | |
| AlivcVoiceCallSettingViewController | 设置页。 |
| AlivcVideoCallUserAuthrization | 工具类。 |
| NSBundle+AlivcVoiceCallSolo | |
| UIViewController+Alert | |
| AliBaseHttpClient | 网络请求。 |
| AliRequestList | |

## 主要功能说明

- 创建并加入频道。

```
  //实例化AliRtcEngine并设置代理。
   _engine = [AliRtcEngine sharedInstance:self extras:@""];
   //获取授权信息。
   //AliRtcAuthInfo:各项参数均需要客户AppServer（客户的server端）通过OpenAPI来获取，然后AppServ
er下发至客户端，客户端将各项参数赋值后，即可joinChannel。
   AuthInfo *authInfo = "从server端获取"
  [_engine  joinChannel:authInfo name:name onResult:^(NSInteger errCode) {
                    //加入频道回调处理
                    if (errCode == 0) {
                        //加入房间成功
                    } else {
                     //加入房间失败
                    }
                }];
```

- 离开频道。

```
//离开频道。
[self.engine leaveChannel];
//销毁SDK实例。
[AliRtcEngine destroy];
```

- 播放伴奏。

```
NSString *path = "文件的路径"
NSString *url = [path stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
//预加载音效文件。
[self.engine preloadAudioEffectWithSoundId:soundId filePath:url];
//播放音效。
[self.engine playAudioEffectWithSoundId:soundId filePath:url cycles:1 publish:YES];
```

- 暂停伴奏。

```
[self.engine pauseAudioEffectWithSoundId:soundId];
```

- 停止伴奏。

```
[self.engine stopAudioEffectWithSoundId:soundId];
```

- 恢复播放伴奏。

```
[self.engine resumeAudioEffectWithSoundId:soundId];
```

- 静音模式。

```
[self.engine muteLocalMic:YES];
```

- 取消静音模式。

```
[self.engine muteLocalMic:NO];
```

- 开启扬声器。

```
[self.engine enableSpeakerphone:YES];
```

- 关闭扬声器。

```
[self.engine enableSpeakerphone:NO];
```

# 2.2. 音视频通话

## 2.2.1. 简介

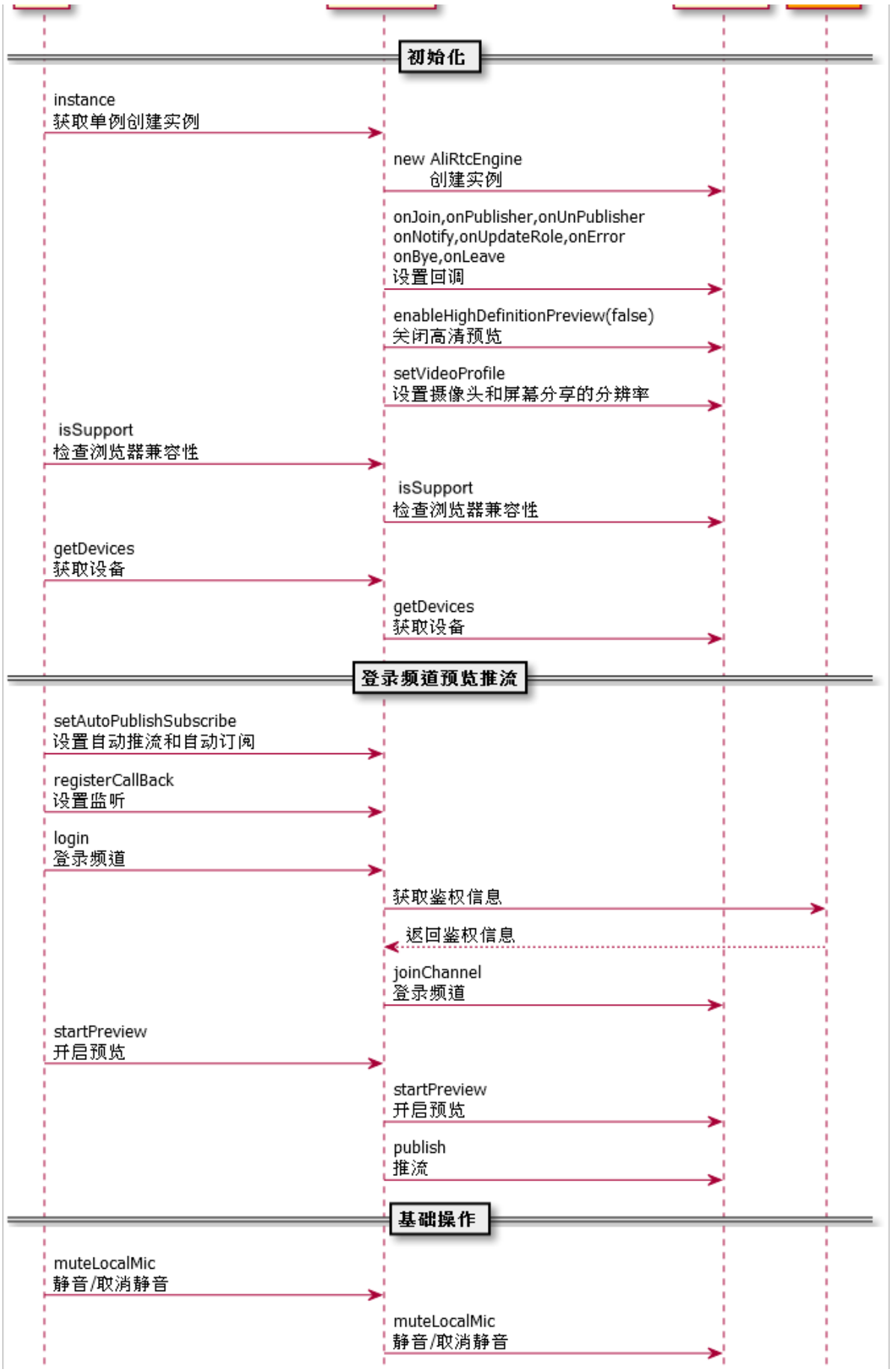
通过阅读本文，您可以快速了解音视频通话基本信息及实现方法。

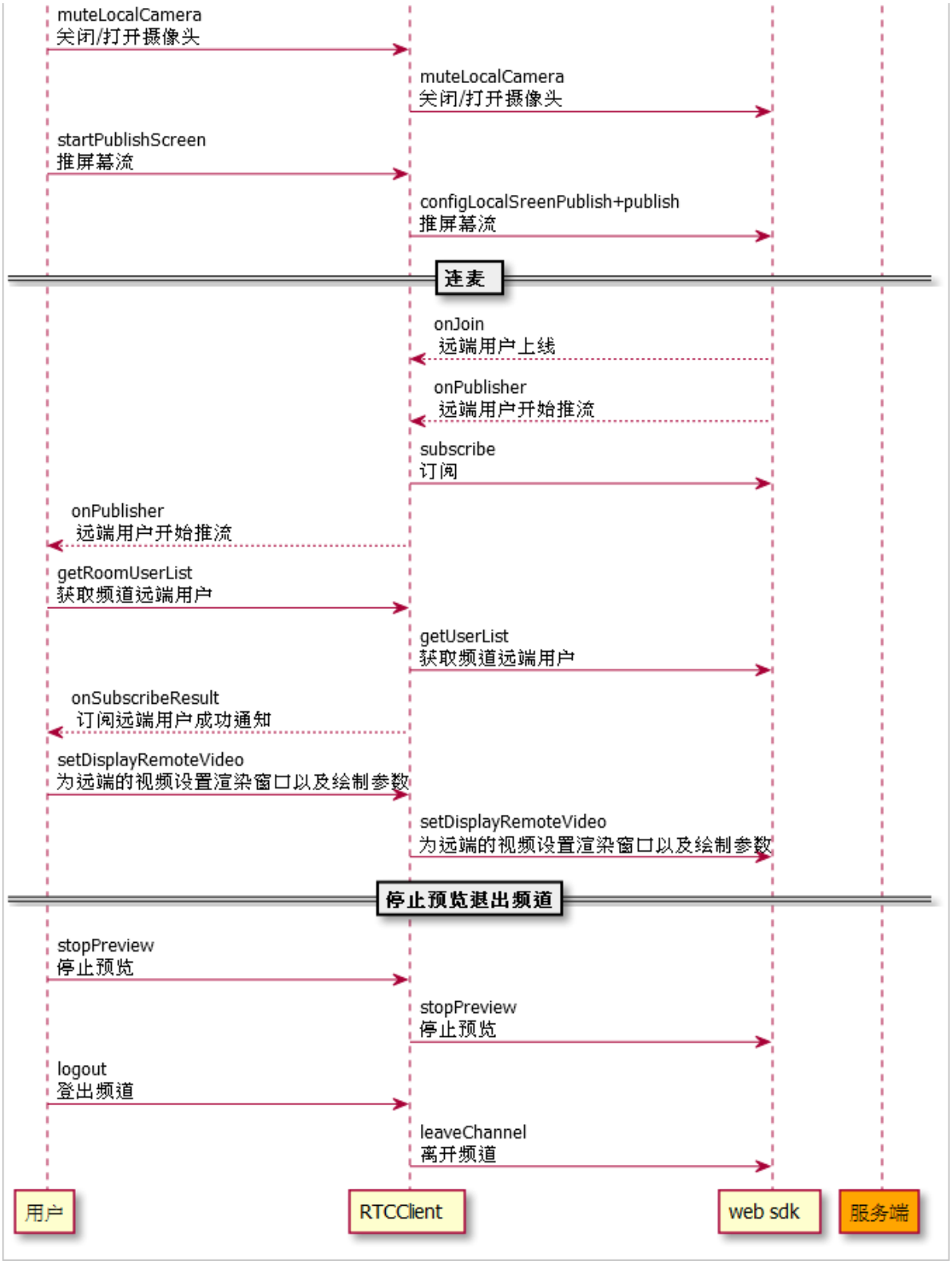
### 主要功能

| 功能 | 描述 |
| --- | --- |
| 多人实时音视频通话 | 支持多人低延时视频通话，单频道支持50人同时在线，支持16人同时开启视频。 |
| 屏幕共享 | Web端可将电脑屏幕画面共享给其它用户。 |

### 实现方法

- Web端实现的时序图如下：

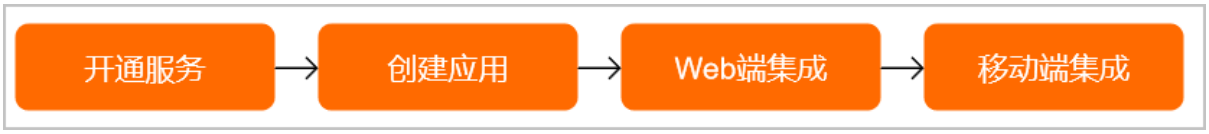

初始化
instance
获取单例创建实例
new AliRtcEngine
创建实例
onJoin,onPublisher,onUnPublisher
onNotify,onUpdateRole,onError
onBye,onLeave
设置回调
enableHighDefinitionPreview(false)
关闭高清预览
setVideoProfile
设置摄像头和屏幕分享的分辨率
isSupport
检查浏览器兼容性
isSupport
检查浏览器兼容性
getDevices
获取设备
getDevices
获取设备
登录频道预览推流
setAutoPublishSubscribe
设置自动推流和自动订阅
registerCallBack
设置监听
login
登录频道
获取鉴权信息
返回鉴权信息
joinChannel
登录频道
startPreview
开启预览
startPreview
开启预览
publish
推流
基础操作
muteLocalMic
静音/取消静音
muteLocalMic
静音/取消静音




- iOS端实现的时序图如下：


## 实现流程

实现流程如下图所示：




| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务 | 进行Web端集成之前，您必须开通音视频通信服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |

| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |
| 3 | Web集成 | 您可以通过源码快速搭建Web端音视频通话。 |
| 4 | 移动端集成：<br>• iOS集成<br>• Android集成 | 您可以通过源码快速搭建移动端音视频通话。 |

### Demo体验

您可以通过钉钉扫描以下二维码，下载安装音视频通话Demo体验。

## 2.2.2. Web集成

通过阅读本文，您可以了解到Web端音视频通话的集成方法。

### 环境要求

Web端具体环境要求，更多信息，请参见使用限制。

### 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 环境中已安装Node.js 6.0或以上版本。具体操作，请参见安装Node.js。
- 如果设备为Mac，需要为浏览器打开屏幕录制权限。

### 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。

iii. 在**AppID/名称**列获取AppID。

iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

> **说明**
> - Demo源码中已经集成AliRTC WEB SDK，可通过npm命令集成。
> - 源码压缩文件内分为Web端、Android端、iOS端三个文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

3. 配置Demo工程。

根据步骤1中获取的AppID和AppKey修改*web/src/core/data/config.js*文件中`appID`和`appKey`的值。

```
export default{
    appId:"",
    appKey:""
}
```

> **说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

4. 运行Demo。

i. 打开后台终端并进入到*web*目录下。

ii. 安装项目依赖。

```
npm install
```

iii. 运行Demo。

**npm run serve**

运行成功后，浏览器会默认打开Demo应用示例。如果没有自动打开，请在浏览器中访问 https://localhost:888。


iv. （可选）压缩静态资源文件，打包发布。

**npm run build**

## Demo源码解析

- 目录结构说明

```
├── dist                                #打包文件
├── public                              #静态资源
├── src                                 #项目文件目录
│   ├── assets                          #静态资源
│   ├── components                      #公共组件
│   ├── core                            #js文件
│   │   ├── data
│   │   │   ├── config.js               #相关配置参数
│   │   ├── http
│   │   │   ├── http.js                 #请求鉴权相关方法
│   │   ├── util
│   │   │   ├── utils.js                #一些公共方法
│   │   ├── rtc-clinet.js               #RTC实例文件
│   ├── plugins
│   ├── router                          #路由
│   ├── views                           #页面
│   │   ├── login
│   │   │   ├── login.vue               #登录页面
│   │   ├── home
│   │   │   ├── index.vue               #会议页面
│   ├── vuex
│   ├── App.vue                         #根组件
│   ├── main.js                         #入口文件
├── vue.config.js                       #vue配置文件
```

- 主要功能说明
  - 检查浏览器是否支持。

    ```
    RtcEngine.instance.isSupport().then(re => {}).catch(err=>{});
    ```

  - 获取设备信息。

    ```
    RtcEngine.instance.getDevices().then(re => {})
    ```

  - 指定摄像头。

    ```
    RtcEngine.instance.currentCamera(deviceId)
    ```

  - 指定麦克风。

    ```
    RtcEngine.instance.currentAudioCapture(deviceId)
    ```

- 开启预览。

```
/**
 * 预览
 * @parame {HtmlVideoElement} video 播放预览画面的video标签
 */
RtcEngine.instance.startPreview(video).then(re=>{})
```

- 停止预览。

```
RtcEngine.instance.stopPreview(video).then(re=>{})
```

- 设置是否自动推流、自动订阅，需要在加入频道之前设置，此接口针对频道设置。

```
/**
   * 设置是否自动推流、自动订阅，默认自动推流、自动订阅
   * @param { boolean } autoPub true表示自动推流
   * @param { boolean } autoSub true表示自动订阅
   */
RtcEngine.instance.setAutoPublishSubscribe(autoPub, autoSub)
```

- 注册回调，需要在加入频道之前设置，此接口针对频道设置。

```
/**
   * 注册回调
   * @param {*} channel 频道
   * @param {*} callback
   */
RtcEngine.instance.registerCallBack(channel, callback)
```

- 加入频道。

```
/**
   * 加入房间
   * @param {*} channel 频道
   * @param {*} userName
   */
RtcEngine.instance.login(channel, userName).then(re=>{})
```

- 开始推流。

```
/**
  * 上麦
  */
RtcEngine.instance.enterSeat(channel)
```

- 停止推流。

```
/**
  * 下麦
  */
RtcEngine.instance.leavelSeat()
```

- 设置是否停止发布本地音频。

```
/**
    * 设置是否停止发布本地音频
    * @param {*} enable
    */
RtcEngine.instance.muteLocalMic(enable)
```

- 设置是否停止发布本地视频。

```
/**
    * 设置是否停止发布本地视频
    * @param {*} enable
    */
RtcEngine.instance.muteLocalCamera(enable)
```

- 设置发布屏幕流。

```
/**
    * 开启屏幕流
    * @param {*} enable
    */
RtcEngine.instance.startPublishScreen()
```

- 设置停止发布屏幕流。

```
/**
    * 停止屏幕流
    * @param {*} enable
    */
RtcEngine.instance.stopPublishScreen()
```

- 订阅音视频。

```
/**
    * 设置远端渲染，默认订阅音频和视频（小流）
    * @param {*} userId
    */
RtcEngine.instance.subscribe(userId).then(re=>{})
```

- 订阅大流。

```
/**
    * 设置远端渲染，默认订阅音频和视频（大流）
    * @param {*} userId
    */
RtcEngine.instance.subscribeLarge(userId).then(re=>{})
```

- 设置远端渲染。

```
/**
   * 设置远端渲染
   * @param {*} userId
   * @param {*} video
   * @param {*} streamType
   */
RtcEngine.instance.setDisplayRemoteVideo(userId, video, streamType)
```

- 获取频道用户列表。

```
/**
   * 获取频道用户列表
   * @return { array | boolean }
   */
RtcEngine.instance.getRoomUserList()
```

- 获取用户信息。

```
/**
   * 获取频道用户信息
   * @param {*} channel 频道
   * @return { array | boolean }
   */
RtcEngine.instance.getUserInfo(channel, userId)
```

- 离开频道。

```
/**
   * 离开频道
   */
RtcEngine.instance.logout().then(re=>{})
```

# 2.2.3. iOS集成

通过阅读本文，您可以了解到iOS端音视频通话的集成方法。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

## 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。

iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

> **说明**
> - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

3. 配置Demo工程。

根据步骤1中获取的AppID和AppKey修改*iOS\RTCSolution\RTCBeaconTower\RTCBeaconTower\ChatRoomViewController.m*文件中 `appID` 和 `appKey` 的值。

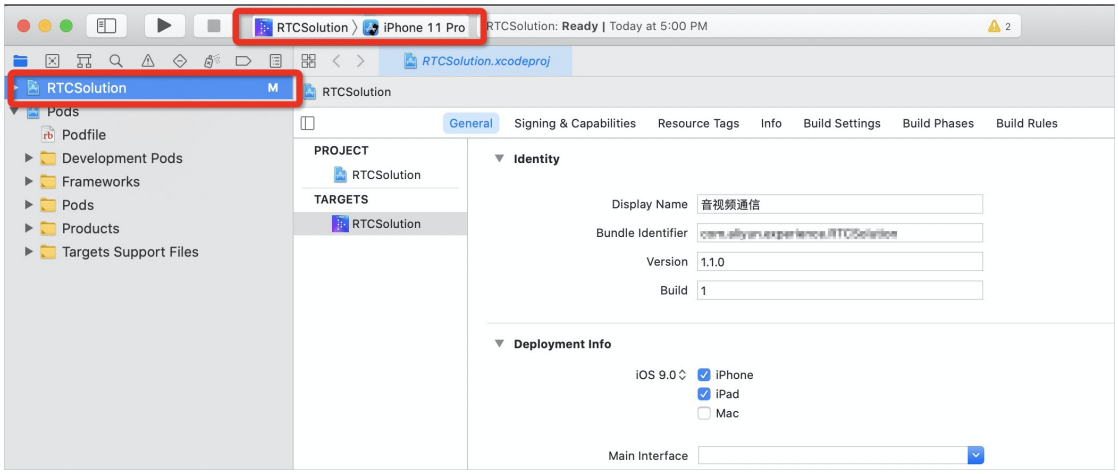
```
328
329 + (AliRtcAuthInfo *)GenerateAliRtcAuthInfoWithChnnelId:(NSString *)channelID {
330     NSString *AppID = @"" ; //修改为自己的appid 该方案仅为开发测试使用，正式上线需要使用服务端的AppServer
331     NSString *AppKey = @""; //修改为自己的appkey 该方案仅为开发测试使用，正式上线需要使用服务端的AppServer
332
333     NSString *userID = [NSString stringWithFormat:@"%d%d",(int)[[NSDate new] timeIntervalSince1970],(int)arc4random()];
334
335     NSString *nonce = [NSString stringWithFormat:@"AK-%@",[[NSUUID UUID] UUIDString]];
336
337     NSTimeInterval interval = 48 * 60 * 60;//48小时时间戳
338     NSDate *datenow = [[NSDate date] initWithTimeIntervalSinceNow:interval];//现在时间
339     long long timestamp = (long)(long)([datenow timeIntervalSince1970]*1000);
340
341     //获取到token
342     NSString *token = [NSString stringWithFormat:@"%@%@%@%@%@%@%lld",AppID,AppKey,channelID,userID,nonce,timestamp];
343
344     //将token加密
345     const char *s = [token cStringUsingEncoding:NSUTF8StringEncoding];
346     NSData *keyData = [NSData dataWithBytes:s length:strlen(s)];
347     uint8_t digest[CC_SHA256_DIGEST_LENGTH] = {0};
348     CC_SHA256(keyData.bytes, (CC_LONG)keyData.length, digest);
349     NSData *out = [NSData dataWithBytes:digest length:CC_SHA256_DIGEST_LENGTH];
350     const unsigned *hashBytes = [out bytes];
351     NSString *hash = [NSString stringWithFormat:@"%08x%08x%08x%08x%08x%08x%08x%08x",
352     ntohl(hashBytes[0]), ntohl(hashBytes[1]), ntohl(hashBytes[2]),
353     ntohl(hashBytes[3]), ntohl(hashBytes[4]), ntohl(hashBytes[5]),
354     ntohl(hashBytes[6]), ntohl(hashBytes[7])];
```

> **说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

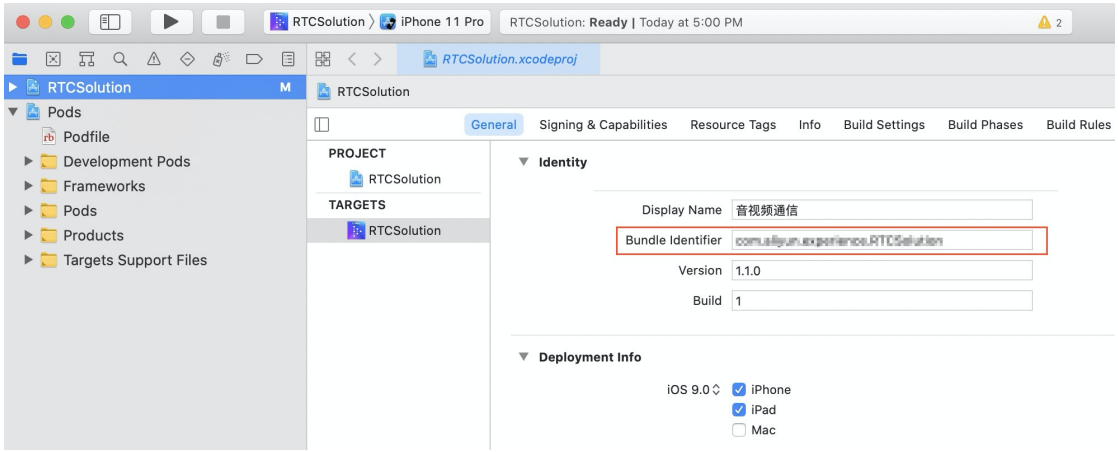
4. 运行Demo。

i. 使用Xcode打开*\RTCSolution\RTCSolution.xcworkspace*工程文件。

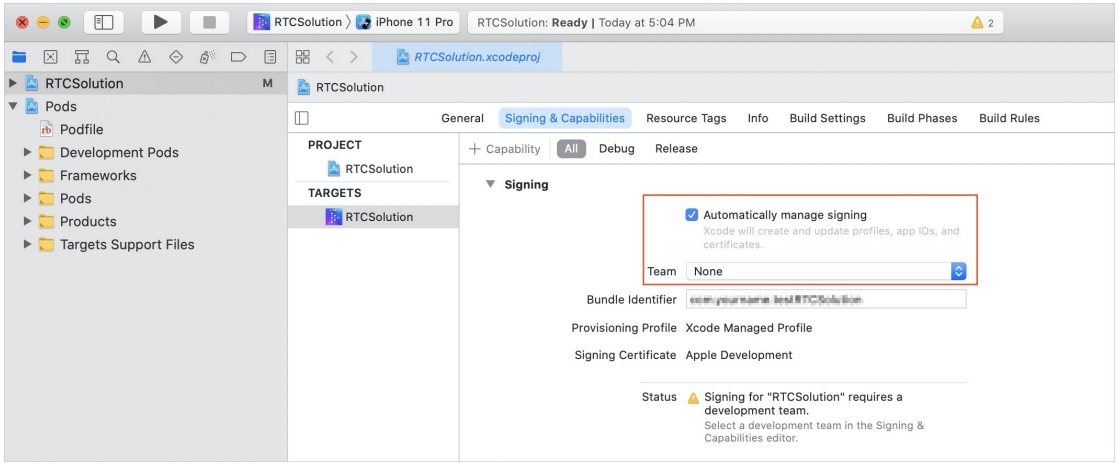
ii. 选择运行的Target为RTCSolution，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。




iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。

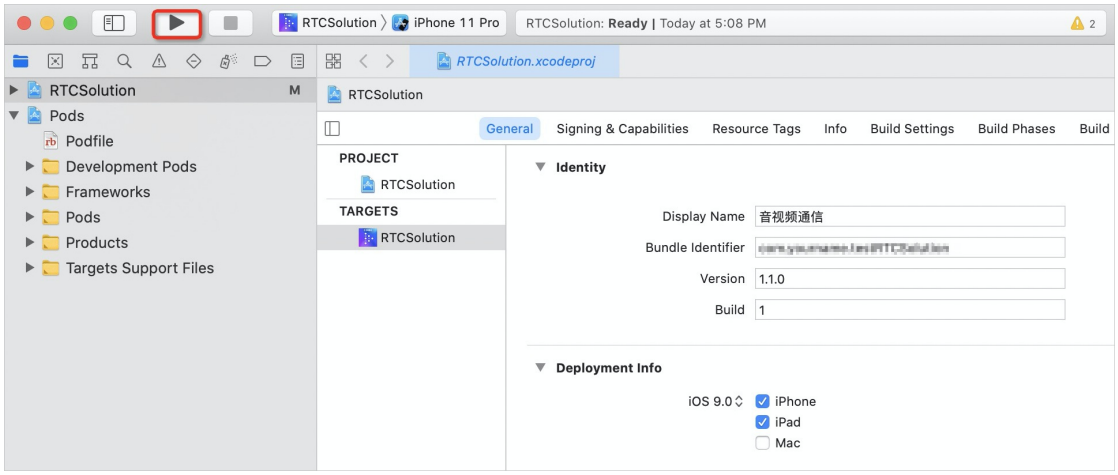



iv. 单击**Signing & Capabilities**页签，选中**Automatically manage signing**，然后单击**Team**下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择**Add an Account...**，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击▶，编译并运行。




如果在编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。

## Demo目录结构说明

RTC把业务代码封装到RTCBeaconTower的库中，因此只需在Podfile中指定RTCBeaconTower库的路径，RTCBeaconTower就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
pod 'RTCBeaconTower', :path => 'RTCBeaconTower'
   ##   pod 'RTCBeaconTower'  说明项目依赖RTCBeaconTower库
   ##   path => 'RTCBeaconTower'  指明RTCBeaconTower库的位置(相对于Podfile)
```

## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取AliRTCBeaconTower的实例对象，初始化RTC SDK。 |
| destorySharedInstance | 销毁AliRTCBeaconTower的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。 |
| login | 根据输入的房间号、用户名加入RTC频道。 |
| logout | 退出RTC频道。 |
| setDelegate | 设置监听。 |
| enterSeat | 连麦。 |
| leavelSeat | 断开连麦。 |
| switchCamera | 切换前后摄像头。 |
| muteLocalCamera | 是否停止本地视频采集。 |
| muteLocalMic | 是否停止本地音频采集。 |

| API | 描述 |
| --- | --- |
| setLocalViewConfig | 为本地预览设置参数及绘制窗口。 |
| setRemoteViewConfig | 为远端的视频设置参数及绘制窗口。 |
| startPreview | 开始本地预览。 |
| stopPreview | 停止本地预览。 |
| configRemoteAudio | 设置是否拉取音频流。 |
| configRemoteCameraTrack | 设置是否拉取camera视频流。 |
| configRemoteScreenTrack | 设置是否拉取screen视频流。 |
| displayName | 获取用户名。 |
| enableSpeakPhone | 设置是否开启扬声器。 |
| subscribe | 手动拉视频和音频流。 |

回调接口

| API | 描述 |
| --- | --- |
| onRemoteUserOffLineNotify | 远端用户下线的通知。 |
| onRemoteUserOnLineNotify | 远端用户上线的通知。 |
| onEnterSeatResult | 用户上麦的通知。 |
| onLeaveSeatResult | 用户下麦的通知。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onOccurError | 错误信息的通知。 |
| onOccurWarning | 警告信息的通知。 |
| onSDKError | SDK发生异常需要销毁实例。 |
| onJoinChannelResult | 加入房间的通知。 |
| onLeaveChannelResult | 离开房间的通知。 |
| onRemoteTrackAvailableNotify | 远端用户音视频流发生变化时的回调。 |
| onSubscribeChangedNotify | 订阅情况发生变化时的回调。 |
| onUserAudioMuted | 用户取消音频的通知。 |
| onUserVideoMuted | 用户取消视频的通知。 |

| API | 描述 |
| --- | --- |
| onNetworkQualityChanged | 网络质量变化时的回调。 |
| onUpdateRoleNotifyWithOldRole | 角色切换成功的通知。 |

功能实现接口

- sharedInstance：获取AliRTCBeaconTower的实例对象，初始化RTC SDK。

```
/**
* @brief 获取单例
* @return AliRTCBeaconTower 单例对象
*/
+ (AliRTCBeaconTower *) sharedInstance;
```

- destorySharedInstance：销毁AliRTCBeaconTower的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/// 销毁RTC SDK
- (void)destroySharedInstance;
```

- login：根据输入的房间号、用户名加入RTC频道。

```
/**
* @brief 加入频道
* @param authInfo    频道号
* @param name    任意用于显示的用户名称，不是User ID
*/
- (void)login:(AliRtcAuthInfo *)authInfo name:(NSString *)name;
```

- logout：退出RTC频道。

```
/// 离开频道
- (void)logout;
```

- setDelegate：设置监听。

```
- (void)setDelegate:(id<RTCVideoliveRoomDelegate> _Nullable)delegate;
```

- enterSeat：连麦。

```
/// 连麦
- (int)enterSeat;
```

- leavelSeat：断开连麦。

```
//断开连麦
- (void)leaveSeat;
```

- switchCamera：切换前后摄像头。

```
/**
* @brief 切换前后摄像头
* @return 0表示Success 非0表示Failure
* @note 只有iOS和android提供这个接口
*/
- (int)switchCamera;
```

- muteLocalCamera：是否停止本地视频采集。

```
/**
* @brief 是否将停止本地视频采集
* @param mute     YES表示停止视频采集；NO表示恢复正常
* @param track    需要停止采集的track
* @return 0表示Success 非0表示Failure
* @note 发送黑色的视频帧。本地预览也呈现黑色。采集，编码，发送模块仍然工作，只是视频内容是黑色帧
*/
- (int)muteLocalCamera:(BOOL)mute forTrack:(AliRtcVideoTrack)track;
```

- muteLocalMic：是否停止本地音频采集。

```
/**
* @brief mute或unmute本地音频采集
* @param mute  YES表示本地音频采集空帧；NO表示恢复正常
* @note mute是指采集和发送静音帧。采集和编码模块仍然在工作
* @return 0表示成功放入队列，-1表示被拒绝
*/
- (int)muteLocalMic:(BOOL)mute;
```

- setLocalViewConfig：为本地预览设置参数及绘制窗口。

```
/**
* @brief 为本地预览设置参数以及绘制窗口
* @param viewConfig 包含了窗口以及渲染方式
* @param track      must be AliVideoTrackCamera
* @return 0表示Success，非0表示Failure
* @note 支持joinChannel之前和之后切换窗口。如果viewConfig或者viewConfig中的view为nil，则停止渲染
* 如果在播放过程中需要重新设置render mode，请保持canvas中其他成员变量不变，仅修改renderMode
* 如果在播放过程中需要重新设置mirror mode，请保持canvas中其他成员变量不变，仅修改mirrorMode
*/
- (int)setLocalViewConfig:(AliVideoCanvas *_Nullable)viewConfig forTrack:(AliRtcVideoTrack)track;
```

- setRemoteViewConfig：为远端的视频设置参数及绘制窗口。

```
/**
* @brief 为远端的视频设置参数及绘制窗口
* @param canvas canvas包含了窗口以及渲染方式
* @param uid    User ID。从App server分配的唯一标示符
* @param track  需要设置的track
* @return 0表示Success，非0表示Failure
* @note 支持joinChannel之前和之后切换窗口。如果canvas为nil或者view为nil，则停止渲染相应的流
*       如果在播放过程中需要重新设置render mode，请保持canvas中其他成员变量不变，仅修改renderMode
*       如果在播放过程中需要重新设置mirror mode，请保持canvas中其他成员变量不变，仅修改mirrorMode
*/
- (int)setRemoteViewConfig:(AliVideoCanvas *)canvas uid:(NSString *)uid forTrack:(AliRtcV
ideoTrack)track;
```

- startPreview：开始本地预览。

```
/**
* @brief 开始本地预览
* @return 0表示Success 非0表示Failure
* @note 如果没有设置view，则无法预览。可以在joinChannel之前就开启预览
*       会自动打开摄像头
*/
- (int)startPreview;
```

- stopPreview：停止本地预览。

```
/**
* @brief 停止本地预览
* @return 0表示Success 非0表示Failure
* @note leaveChannel会自动停止本地预览
*       会自动关闭摄像头 （如果正在publish camera流，则不会关闭摄像头）
*/
- (int)stopPreview;
```

- configRemoteAudio：设置是否拉取音频流。

```
/**
 * @brief 设置是否拉取音频流
 * @param uid userId 从App server分配的唯一标示符
 * @param enable  YES：拉取；NO：不拉取
 * @note 可以在joinChannel之前或者之后设置。如果已经订阅该用户的流，需要调用subscribe:(NSString
*)uid onResult:才生效
 */
- (void)configRemoteAudio:(NSString *)uid enable:(BOOL)enable;
```

- configRemoteCameraTrack：设置是否拉取camera视频流。

```
/**
 * @brief 设置是否拉取camera视频流
 * @param uid userId 从App server分配的唯一标示符。
 * @param master  是否优先拉取大流
 * @param enable  YES：拉取；NO：不拉取
 * @note 可以在joinChannel之前或者之后设置。如果已经订阅该用户的流，需要调用subscribe:(NSString *)uid onResult:才生效
 */
- (void)configRemoteCameraTrack:(NSString *)uid preferMaster:(BOOL)master enable:(BOOL)enable;
```

- configRemoteScreenTrack：设置是否拉取screen视频流。

```
/**
 * @brief 设置是否拉取screen视频流
 * @param uid  uerId从App server分配的唯一标示符
 * @param enable  YES：拉取；NO：不拉取
 * @note 可以在joinChannel之前或者之后设置。如果已经订阅该用户的流，需要调用subscribe:(NSString *)uid onResult:才生效
 */
- (void)configRemoteScreenTrack:(NSString *)uid enable:(BOOL)enable;
```

- displayName：获取用户名。

```
/// 获取用户名
/// @param userid userId
- (NSString *)displayName:(NSString *)userid;
```

- enableSpeakPhone：设置是否开启扬声器。

```
- (int)enableSpeakPhone:(BOOL)enable;
```

- subscribe：手动拉视频和音频流。

```
/**
* @brief 手动拉视频和音频流
* @param uid        User ID。不允许为nil
* @param onResult   当subscribe执行结束后调用这个回调
* @note 如果需要手动选择拉取的流，调用configRemoteAudio, configRemoteTrack,
*       configRemoteScreenTrack来设置。缺省是拉取audio和camera track
*       如果需要unsub所有的流，先通过configRemoteAudio, configRemoteTrack,
*       configRemoteScreenTrack来清除设置，然后调用subscribe
*/
- (void)subscribe:(NSString *)uid onResult:(void (^)(NSString *uid, AliRtcVideoTrack vt, AliRtcAudioTrack at))onResult;
```

回调接口

- onRemoteUserOffLineNotify：远端用户下线的通知。

```
/// 远端用户下线通知
/// @param uid 用户id
- (void)onRemoteUserOffLineNotify:(NSString *)uid;
```

- onRemoteUserOnLineNotify：远端用户上线的通知。

```
/// 远端用户上线通知
/// @param uid 用户id
- (void)onRemoteUserOnLineNotify:(NSString *)uid;
```

- onEnterSeatResult：用户上麦的通知。

```
/// 自己上麦通知
/// @param errorCode 结果
- (void)onEnterSeatResult:(int)errorCode;
```

- onLeaveSeatResult：用户下麦的通知。

```
/// 自己下线通知
/// @param errorCode 结果
- (void)onLeaveSeatResult:(int)errorCode;
```

- onRoomDestroy：房间被销毁的回调。

```
/// 房间被销毁通知
- (void)onRoomdestroy;
```

- onOccurError：错误信息的通知。

```
/**
 * @brief 如果engine出现error，通过这个回调通知app
 * @param error  Error type
 */
- (void)onOccurError:(int)error;
```

- onOccurWarning：警告信息的通知。

```
/**
* @brief 如果engine出现warning，通过这个回调通知app
* @param warn  Warning type
*/
- (void)onOccurWarning:(int)warn;
```

- onSDKError：SDK发生异常需要销毁实例。

```
/**
* @brief 如果engine出现严重error，通过这个回调通知app
* @param error  错误类型
*/
- (void)onSDKError:(int)error;
```

- onJoinChannelResult：加入房间的通知。

```
/**
* @brief 加入频道结果
* @param result 加入频道结果，成功返回0，失败返回错误码
* @note 此回调等同于joinChannel接口的block，二者择一处理即可
*/
- (void)onJoinChannelResult:(int)result authInfo:(AliRtcAuthInfo *)authInfo;
```

- onLeaveChannelResult：离开房间的通知。

```
/**
* @brief 离开频道结果
* @param result 离开频道结果，成功返回0，失败返回错误码
* @note 调用leaveChannel接口后返回，如果leaveChannel后直接调用destroy，将不会收到此回调
*/
- (void)onLeaveChannelResult:(int)result;
```

- onRemoteTrackAvailableNotify：远端用户音视频流发生变化时的回调。

```
/**
* @brief 当远端用户的流发生变化时，返回这个消息
* @note 远方用户停止推流，也会发送这个消息
*/
- (void)onRemoteTrackAvailableNotify:(NSString *)uid audioTrack:(AliRtcAudioTrack)audioTr
ack videoTrack:(AliRtcVideoTrack)videoTrack;
```

- onSubscribeChangedNotify：当订阅情况发生变化时的回调。

```
/**
 * @brief 当订阅情况发生变化时，返回这个消息
 */
- (void)onSubscribeChangedNotify:(NSString *)uid audioTrack:(AliRtcAudioTrack)audioTrack
videoTrack:(AliRtcVideoTrack)videoTrack;
```

- onUserAudioMuted：用户取消音频的通知

```
/**
 * @brief 用户muteAudio通知
 * @param uid 执行muteAudio的用户
 * @param isMute YES:静音 NO:未静音
 */
- (void)onUserAudioMuted:(NSString *)uid onUserAudioMuted:(BOOL)isMute;
```

- onUserVideoMuted：用户取消视频通知。

```
/**
 * @brief 用户muteVideo通知
 * @param uid 执行muteVideo的用户
 * @param isMute YES:推流黑帧 NO:正常推流
 */
- (void)onUserVideoMuted:(NSString *)uid videoMuted:(BOOL)isMute;
```

- onNetworkQualityChanged：网络质量变化时的回调。

```
/**
 * @brief 网络质量变化时发出的消息
 * @param uid 网络质量发生变化的uid
 * @param upQuality  上行网络质量
 * @param downQuality  下行网络质量
 * @note 当网络质量发生变化时触发，uid为@""时代表self的网络质量变化
 */
- (void)onNetworkQualityChanged:(NSString *)uid
                onNetworkQualityChanged:(AliRtcNetworkQuality)upQuality
              downNetworkQuality:(AliRtcNetworkQuality)downQuality;
```

- onUpdateRoleNotifyWithOldRole：角色切换成功的通知。

```
/**
 * @brief 当用户角色发生变化化时通知
 * @param oldRole 变化前角色类型
 * @param newRole 变化后角色类型
 * @note 调用setClientRole方法切换角色成功时触发此回调
 */
- (void)onUpdateRoleNotifyWithOldRole:(AliRtcClientRole)oldRole newRole:(AliRtcClientRole
)newRole;
```

# 2.2.4. Android集成

通过阅读本文，您可以了解到Android端音视频通话的集成方法。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

环境中已安装Android Studio 3.0或以上版本，更多信息，请参见Android Studio。

## 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。
   iv. 单击**操作**列的**查询AppKey**，获取AppKey。

   > **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > **说明**
   > - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

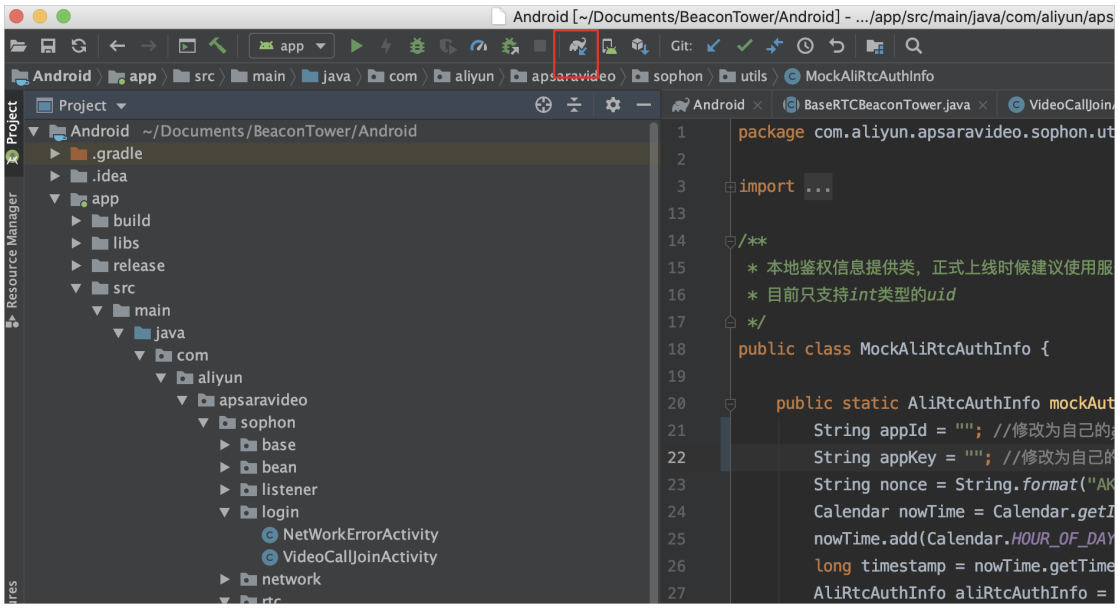
3. 配置Demo工程。

   根据步骤1中获取的AppID和AppKey修改*Android/app/src/main/java/com/aliyun/apsaravideo/sophon/utils/MockAliRtcAuthInfo.java*文件中 `appID` 和 `appKey` 的值。

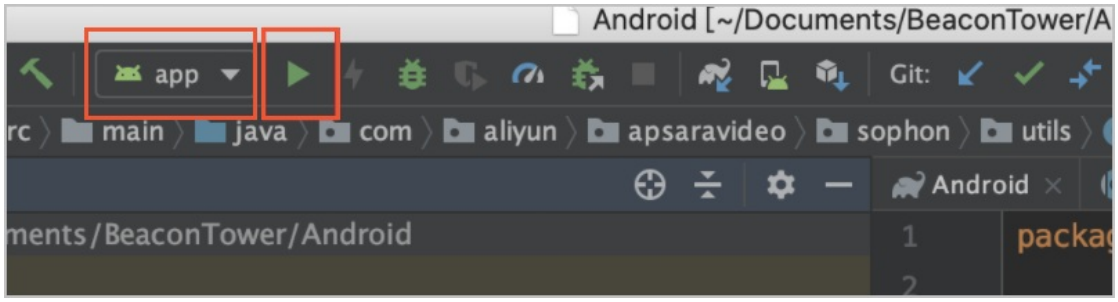
```
public class MockAliRtcAuthInfo {

    public static AliRtcAuthInfo mockAuthInfo(String channelId, String userId){
        String appId = ""; //修改为自己的appid 该方案仅为开发测试使用，正式上线需要使用服务端的AppServer
        String appKey = ""; //修改为自己的appkey 该方案仅为开发测试使用，正式上线需要使用服务端的AppServer
        String nonce = String.format("AK-%s", UUID.randomUUID().toString());
        Calendar nowTime = Calendar.getInstance();
        nowTime.add(Calendar.HOUR_OF_DAY, amount: 48);
        long timestamp = nowTime.getTimeInMillis() / 1000;
        AliRtcAuthInfo aliRtcAuthInfo = new AliRtcAuthInfo();
        try {
            aliRtcAuthInfo.setAppid(appId);
            aliRtcAuthInfo.setConferenceId(channelId);
            aliRtcAuthInfo.setUserId(userId);
            aliRtcAuthInfo.setToken(createToken(appId,appKey,channelId,userId,nonce,timestamp));
            aliRtcAuthInfo.setNonce(nonce);
            aliRtcAuthInfo.setTimestamp(timestamp);
            String[] gslb = new String[]{"https://rgslb.rtc.aliyuncs.com"};
            aliRtcAuthInfo.setGslb(gslb);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return aliRtcAuthInfo;
```

**说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

4. 运行Demo。
   i. 打开Android Studio，单击**Open an Existing Project**，选择Android文件夹。
   ii. 单击  ，同步工程。



iii. 将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。




> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

## Demo目录结构说明

应用程序入口目录说明：

| 文件名 | 说明 |
| --- | --- |
| base | activity基础类。 |
| bean | 实体类。 |
| listener | 监听回调类。 |
| login | 登录页。 |
| network | 网络请求。 |
| adapter | 列表控件adapter。 |
| rtc | RTC。 |
| util | 工具类。 |
| videocall | 通话页面。 |
| widget | 控件。 |

## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取RTCBeaconTowerImpl的实例对象，初始化RTC SDK。 |
| destorySharedInstance | 销毁RTCBeaconTowerImpl的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。 |

| API | 描述 |
| --- | --- |
| joinChannel | 加入RTC频道。 |
| logout | 退出RTC频道。 |
| muteLocalMic | 是否停止本地音频采集。 |
| muteLocalCamera | 是否停止本地视频采集。 |
| switchCamera | 切换前后摄像头。 |
| setLocalViewConfig | 为本地预览设置参数及绘制窗口。 |
| startPreview | 开始本地预览。 |
| stopPreview | 停止本地预览。 |
| configRemoteCameraTrack | 设置是否拉取相机、屏幕、音频流，必须调用subscribe才能生效。 |
| subscribe | 手动拉视频和音频流。 |
| setRemoteViewConfig | 为远端的视频设置参数及绘制窗口。 |
| setDelegate | 设置监听。 |
| getUserInfo | 获取用户信息。 |

回调接口

| API | 描述 |
| --- | --- |
| onJoinChannelResult | 加入房间的通知。 |
| onRemoteTrackAvailableNotify | 远端用户音视频流发生变化时的回调。 |
| onSubscribeChangedNotify | 订阅情况发生变化时的回调。 |
| onNetworkQualityChanged | 网络质量变化时的回调。 |
| onRemoteUserOnLineNotify | 远端用户上线的通知。 |
| onRemoteUserOffLineNotify | 远端用户下线的通知。 |

功能实现接口

- sharedInstance：获取RTCBeaconTowerImpl的实例对象，初始化RTC SDK。

```
/**
 * 获取单例
 */
public static RTCBeaconTowerImpl sharedInstance() {
    return RTCBeaconTowerImpl.sharedInstance();
}
```

- destorySharedInstance：销毁RTCBeaconTowerImpl的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/**
     * 销毁实例
     */
    public abstract void destorySharedInstance();
```

- joinChannel：加入RTC频道。

```
    /**
     * 加入房间
     */
    public abstract void joinChannel(AliRtcAuthInfo authInfo, String displayName);
```

- logout：退出RTC频道。

```
    /**
     * 登出
     */
    public abstract void logout();
```

- muteLocalMic：是否停止本地音频采集。

```
    /**
     * 停止发布音频
     */
    public abstract int muteLocalMic(boolean isMute);
```

- muteLocalCamera：是否停止本地视频采集。

```
  /**
     * 停止发布视频
     */
    public abstract int muteLocalCamera(boolean isMute);
```

- switchCamera：切换前后摄像头。

```
  /**
     * 切换摄像头
     */
    public abstract int switchCamera();
```

- setLocalViewConfig：为本地预览设置参数及绘制窗口。

```
  /**
     * 设置本地预览渲染参数
     */
    public abstract void setLocalViewConfig(AliRtcEngine.AliVideoCanvas localAliVideoCanv
as, AliRtcEngine.AliRtcVideoTrack aliRtcVideoTrackCamera);
```

- startPreview：开始本地预览。

```
/**
     * 开启预览
     */
    public abstract void startPreview();
```

- stopPreview：停止本地预览。

```
   /**
     * 停止预览
     */
    public abstract void stopPreview();
```

- configRemoteCameraTrack：设置是否拉取相机、屏幕、音频流，必须调用subscribe才能生效。

```
    /**
     * 设置是否订阅远端相机流。默认为订阅大流。当对流进行操作时（如手动订阅，关闭订阅），必须调用subscribe才能生效
     */
    public abstract void configRemoteCameraTrack(String userId, boolean master, boolean enable);
```

- subscribe：手动拉视频和音频流。

```
  /**
     * 订阅
     */
    public abstract void subscribe(String userId);
```

- setRemoteViewConfig：为远端的视频设置参数及绘制窗口。

```
/**
     * 设置远端渲染参数
     */
    public abstract void setRemoteViewConfig(AliRtcEngine.AliVideoCanvas aliVideoCanvas, String uid, AliRtcEngine.AliRtcVideoTrack aliRtcVideoTrack);
```

- setDelegate：设置监听。

```
  /**
     * 设置监听
     */
    public abstract void setDelegate(RTCInteractiveClassDelegate callback);
```

- getUserInfo：获取用户信息。

```
    /**
     * 获取用户信息
     */
      public abstract AliRtcRemoteUserInfo getUserInfo(String s);
```

回调接口

- onJoinChannelResult：加入房间的通知。

```
    /**
     * 加入房间通知
     * @param result 0为成功，反之失败
     */
    void onJoinChannelResult(int result);
```

- onRemoteTrackAvailableNotify：远端用户音视频流发生变化时的回调。

```
   /**
    *
    * 当订阅情况发生变化时，返回这个消息onSubscribeChangedNotify
    * @param userId  用户ID
    * @param videoTrack     订阅成功的视频流
    * @param audioTrack     订阅成功的音频流
    */
   void onRemoteTrackAvailableNotify(String userId, AliRtcEngine.AliRtcAudioTrack audioTr
ack, AliRtcEngine.AliRtcVideoTrack videoTrack);
```

- onSubscribeChangedNotify：当订阅情况发生变化时的回调。

```
/**
    *
    * 订阅结果回调
    * @param userId  用户ID
    * @param videoTrack     订阅成功的视频流
    * @param audioTrack     订阅成功的音频流
    */
   void onSubscribeChangedNotify(String userId, AliRtcEngine.AliRtcAudioTrack audioTrack,
AliRtcEngine.AliRtcVideoTrack videoTrack);
```

- onNetworkQualityChanged：网络质量变化时的回调。

```
/**
     * 网络状态回调
     *
     * @param aliRtcNetworkQuality1 下行网络质量
     * @param aliRtcNetworkQuality  上行网络质量
     * @param s                     用户ID
     */
    void onNetworkQualityChanged(String s, AliRtcEngine.AliRtcNetworkQuality aliRtcNetwor
kQuality, AliRtcEngine.AliRtcNetworkQuality aliRtcNetworkQuality1);
```

- onRemoteUserOnLineNotify：远端用户上线的通知。

```
   /**
    * 用户上线通知
    *
    * @param userId 用户ID
    */
   void onRemoteUserOnLineNotify(String userId);
```

- onRemoteUserOffLineNotify：远端用户下线的通知。

```
/**
 * 用户下线通知
 * @param userId 用户ID
 */
void onRemoteUserOffLineNotify(String userId);
```

# 2.3. 互动大班课

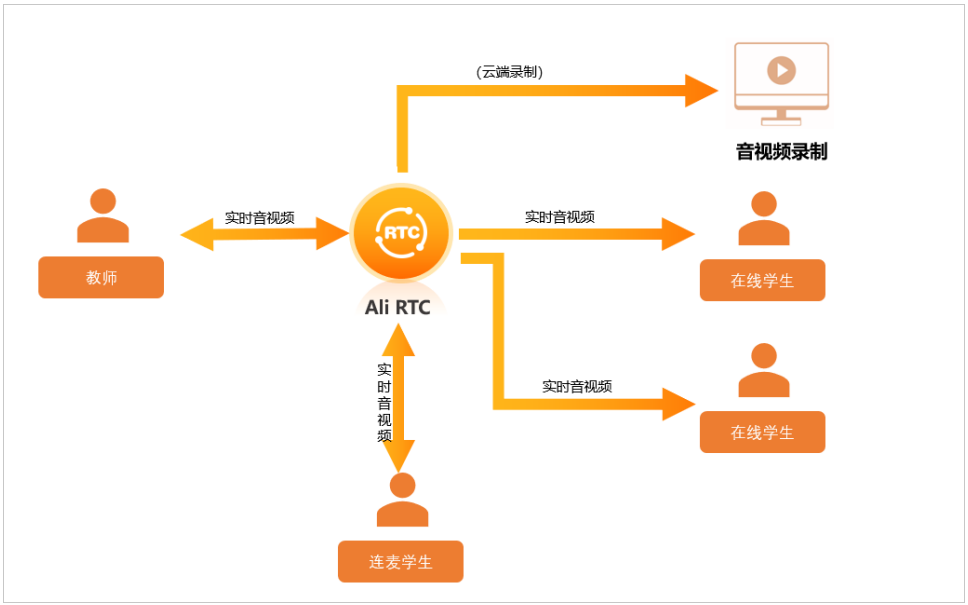
## 2.3.1. 简介

通过阅读本文，您可以快速了解互动大班课基本信息及实现方法。

### 使用场景

互动大班课一般由一名讲师和上千名学生组成，讲师在线进行视频授课，学生通过网络实时收看或收听授课内容。授课过程中学生可以与讲师进行音视频连麦，其他学生也可以同步收看或收听连麦学生。

### 构架方案

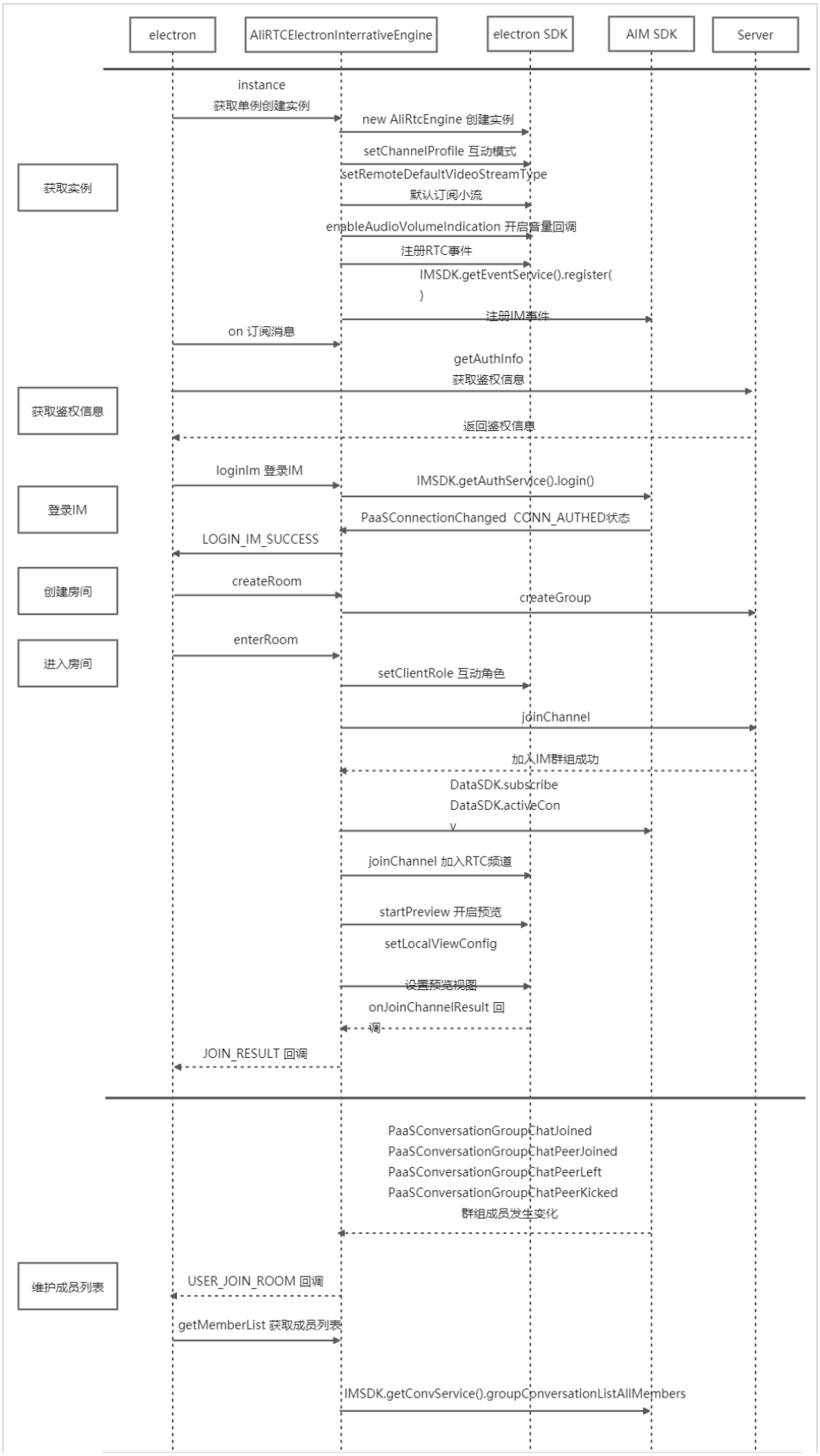


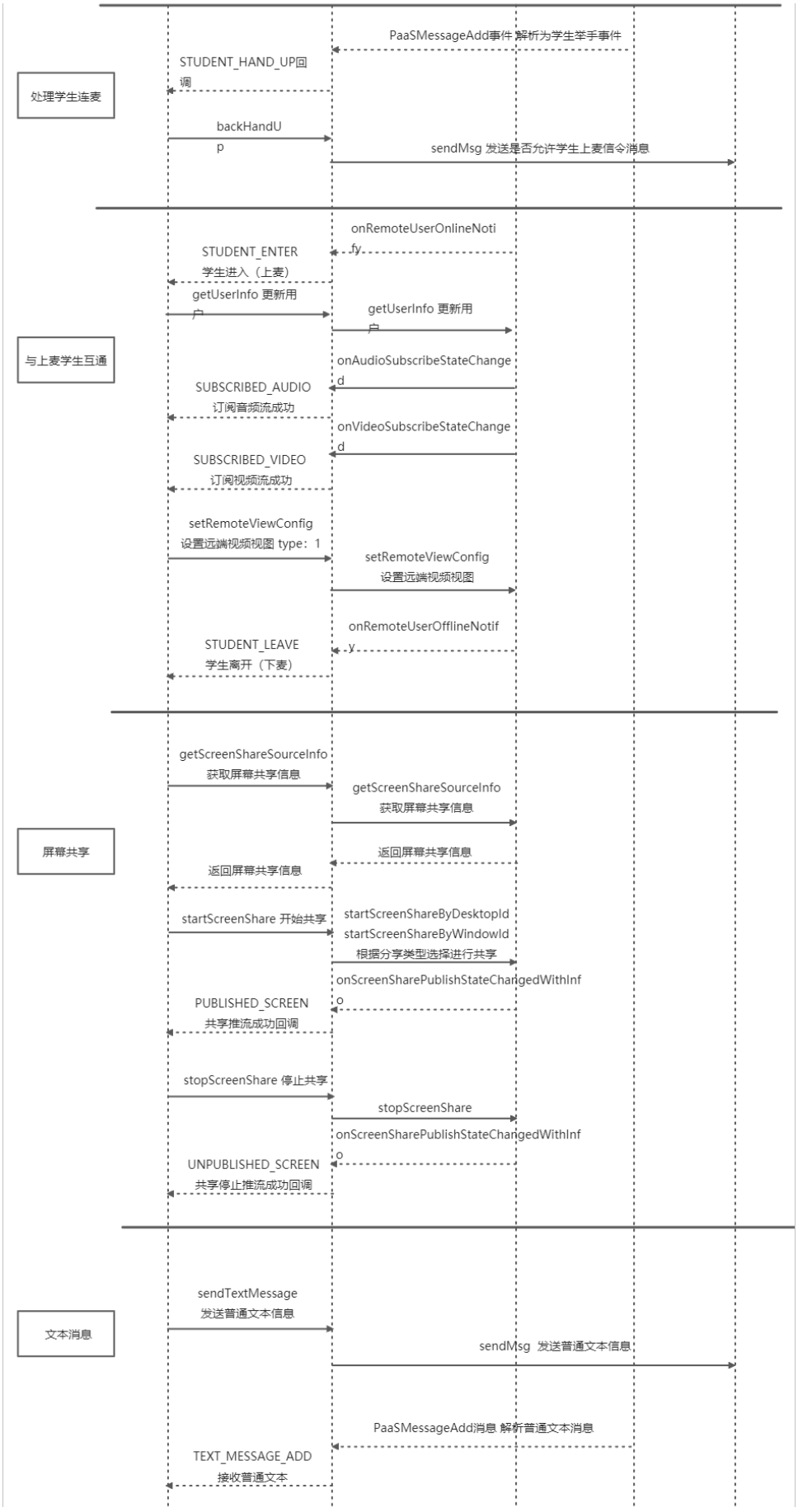

### 主要功能

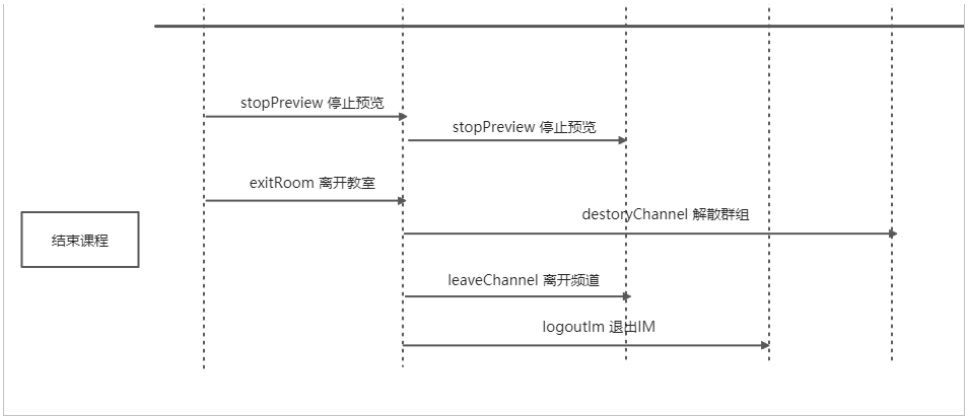
| 功能 | 描述 |
| --- | --- |
| 实时音视频通话 | • 学生可以实时收看或收听讲师的音视频。<br>• 学生可以与讲师进行音视频连麦。<br>• 所有学生均可以收看或收听讲师和学生连麦的音视频。 |
| 屏幕共享 | 老师可以将电脑画面共享给学生观看，丰富教学内容。 |
| 多人互动 | 支持多名学生同时与老师进行音视频互动连麦。 |
| 用户聊天 | 支持老师、学生在房间内发送文字信息聊天。 |

## 实现方法

老师端（Electron）

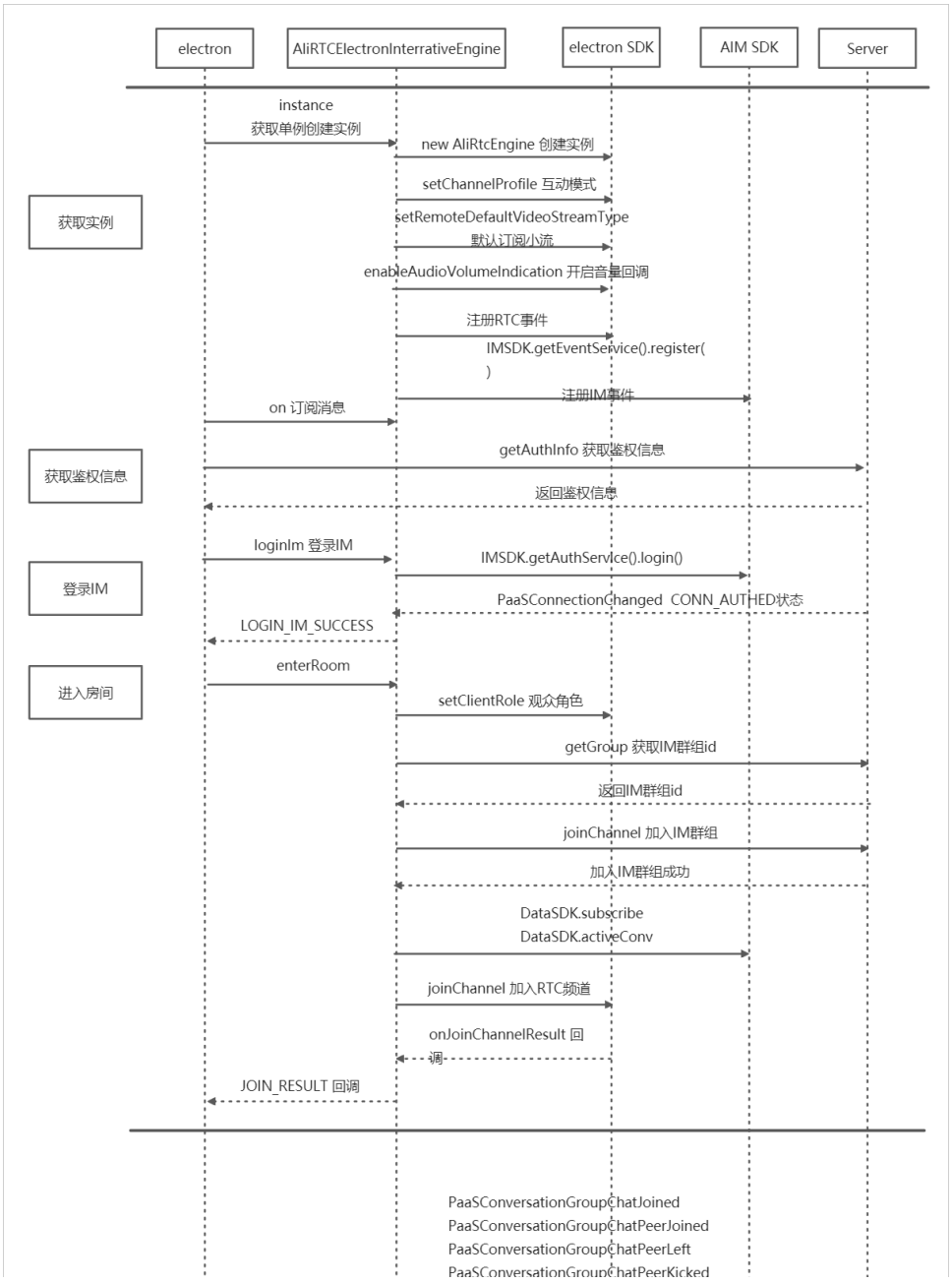


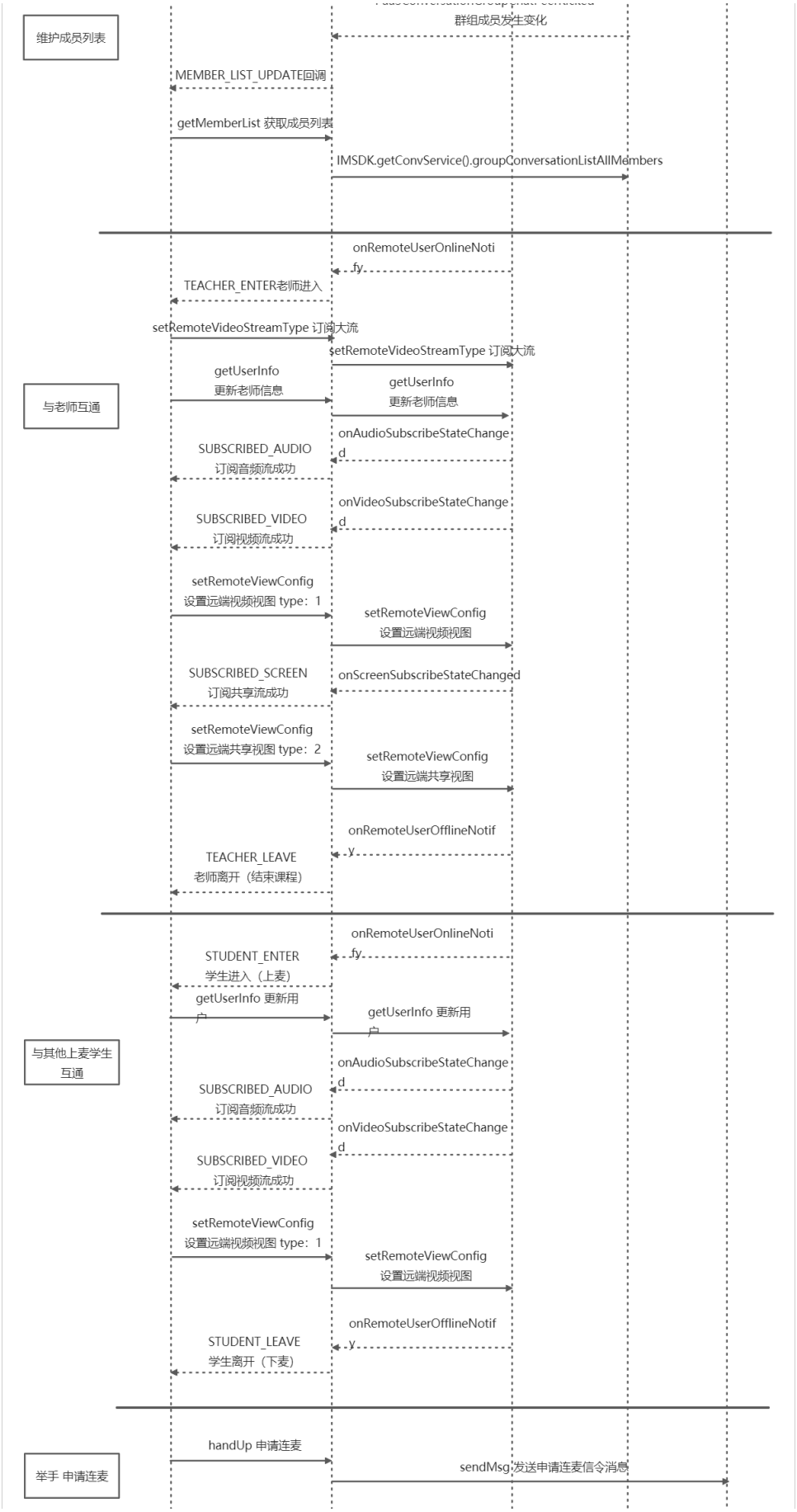

处理学生连麦
PaaSMessageAdd事件 解析为学生举手事件
STUDENT_HAND_UP回调
backHandUp
sendMsg 发送是否允许学生上麦信令消息
与上麦学生互通
onRemoteUserOnlineNotify
STUDENT_ENTER
学生进入（上麦）
getUserInfo 更新用户
getUserInfo 更新用户
onAudioSubscribeStateChanged
SUBSCRIBED_AUDIO
订阅音频流成功
onVideoSubscribeStateChanged
SUBSCRIBED_VIDEO
订阅视频流成功
setRemoteViewConfig
设置远端视频视图 type：1
setRemoteViewConfig
设置远端视频视图
onRemoteUserOfflineNotify
STUDENT_LEAVE
学生离开（下麦）
屏幕共享
getScreenShareSourceInfo
获取屏幕共享信息
getScreenShareSourceInfo
获取屏幕共享信息
返回屏幕共享信息
返回屏幕共享信息
startScreenShare 开始共享
startScreenShareByDesktopId
startScreenShareByWindowId
根据分享类型选择进行共享
onScreenSharePublishStateChangedWithInfo
PUBLISHED_SCREEN
共享推流成功回调
stopScreenShare 停止共享
stopScreenShare
onScreenSharePublishStateChangedWithInfo
UNPUBLISHED_SCREEN
共享停止推流成功回调
文本消息
sendTextMessage
发送普通文本信息
sendMsg 发送普通文本信息
PaaSMessageAdd消息 解析普通文本消息
TEXT_MESSAGE_ADD
接收普通文本

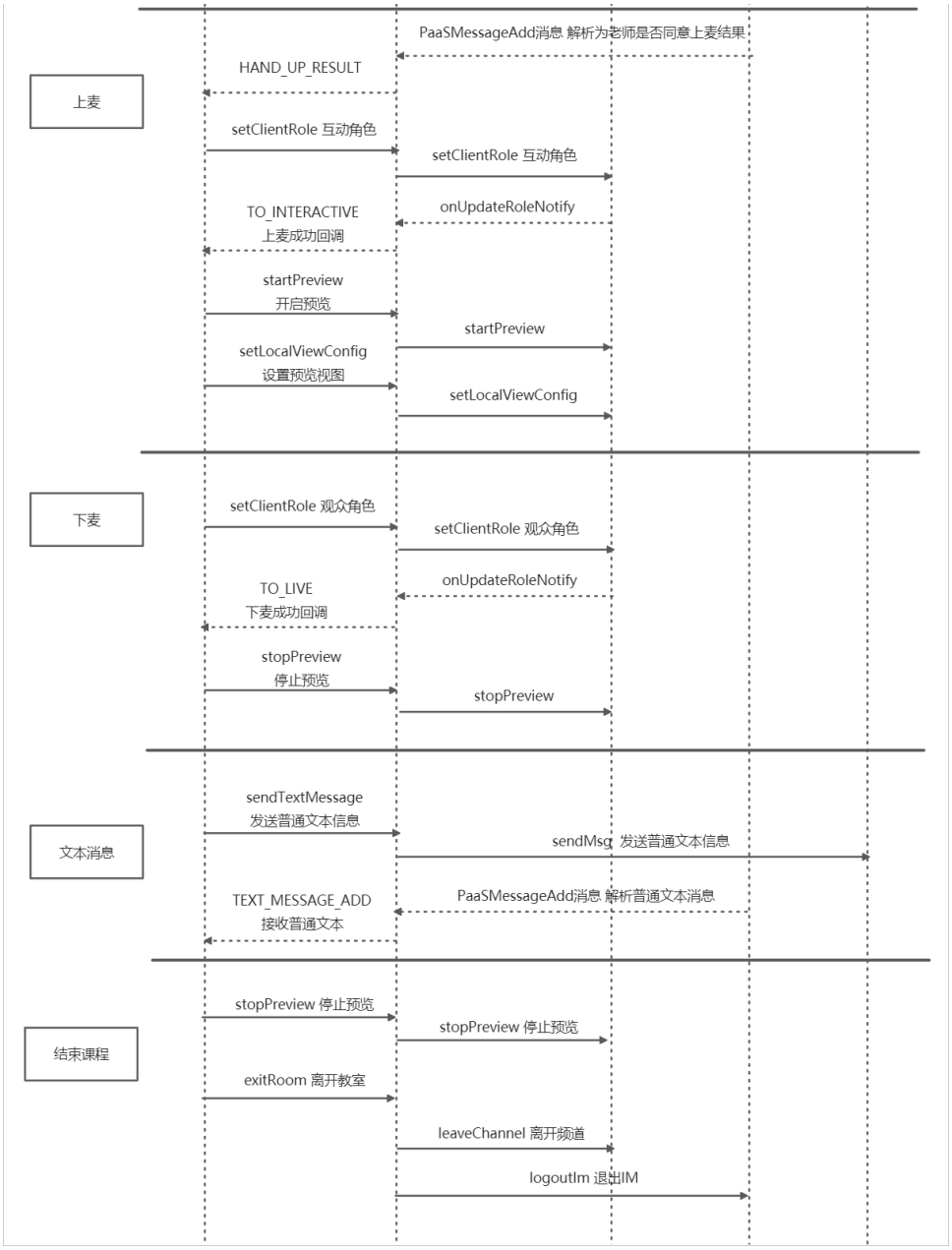



学生端

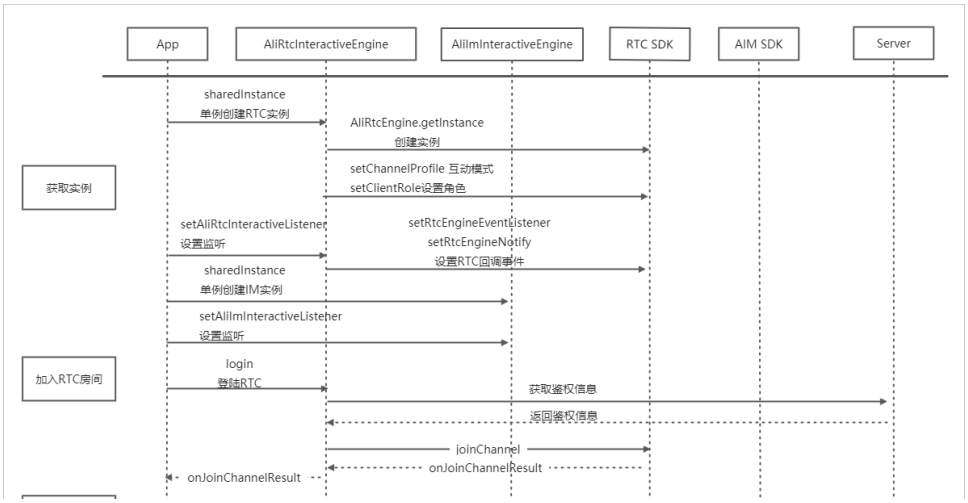
- Electron

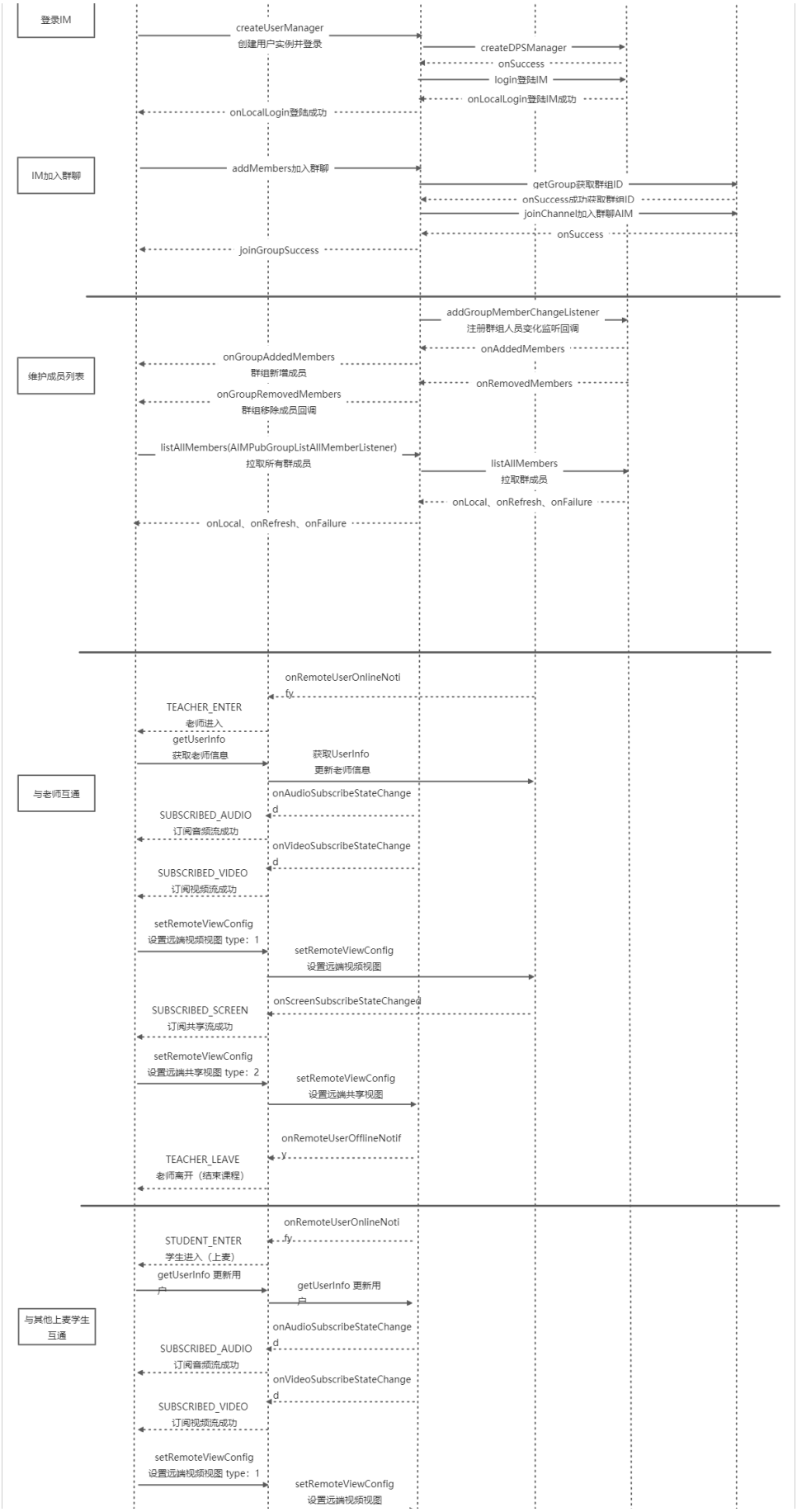


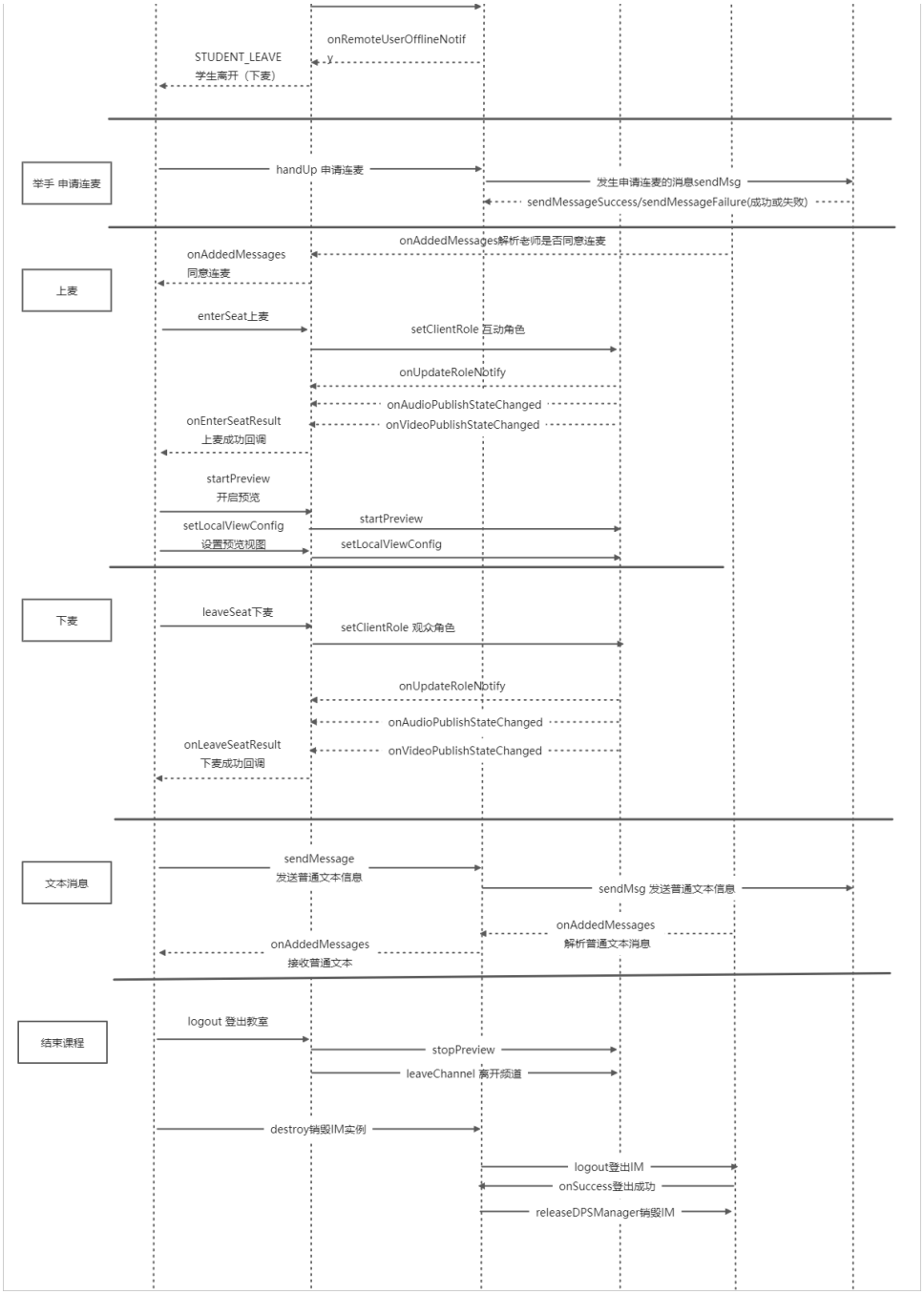

维护成员列表
群组成员发生变化
MEMBER_LIST_UPDATE回调
getMemberList 获取成员列表
IMSDK.getConvService().groupConversationListAllMembers
onRemoteUserOnlineNotify
TEACHER_ENTER老师进入
setRemoteVideoStreamType 订阅大流
setRemoteVideoStreamType 订阅大流
与老师互通
getUserInfo
更新老师信息
getUserInfo
更新老师信息
onAudioSubscribeStateChanged
SUBSCRIBED_AUDIO
订阅音频流成功
onVideoSubscribeStateChanged
SUBSCRIBED_VIDEO
订阅视频流成功
setRemoteViewConfig
设置远端视频视图 type：1
setRemoteViewConfig
设置远端视频视图
SUBSCRIBED_SCREEN
订阅共享流成功
onScreenSubscribeStateChanged
setRemoteViewConfig
设置远端共享视图 type：2
setRemoteViewConfig
设置远端共享视图
onRemoteUserOfflineNotify
TEACHER_LEAVE
老师离开（结束课程）
onRemoteUserOnlineNotify
STUDENT_ENTER
学生进入（上麦）
getUserInfo 更新用户
getUserInfo 更新用户
与其他上麦学生互通
onAudioSubscribeStateChanged
SUBSCRIBED_AUDIO
订阅音频流成功
onVideoSubscribeStateChanged
SUBSCRIBED_VIDEO
订阅视频流成功
setRemoteViewConfig
设置远端视频视图 type：1
setRemoteViewConfig
设置远端视频视图
onRemoteUserOfflineNotify
STUDENT_LEAVE
学生离开（下麦）
handUp 申请连麦
sendMsg 发送申请连麦信令消息
举手 申请连麦


PaaSMessageAdd消息 解析为老师是否同意上麦结果
HAND_UP_RESULT
上麦
setClientRole 互动角色
setClientRole 互动角色
TO_INTERACTIVE
上麦成功回调
onUpdateRoleNotify
startPreview
开启预览
startPreview
setLocalViewConfig
设置预览视图
setLocalViewConfig
下麦
setClientRole 观众角色
setClientRole 观众角色
TO_LIVE
下麦成功回调
onUpdateRoleNotify
stopPreview
停止预览
stopPreview
文本消息
sendTextMessage
发送普通文本信息
sendMsg 发送普通文本信息
TEXT_MESSAGE_ADD
接收普通文本
PaaSMessageAdd消息 解析普通文本消息
结束课程
stopPreview 停止预览
stopPreview 停止预览
exitRoom 离开教室
leaveChannel 离开频道
logoutIm 退出IM

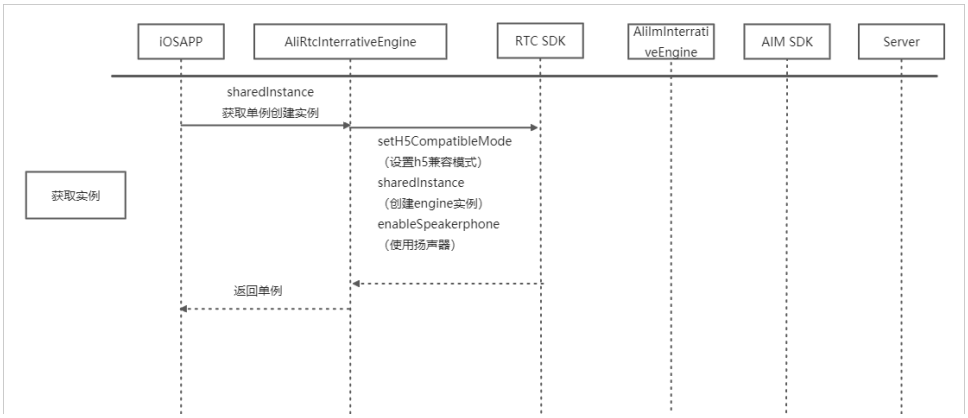
- Android

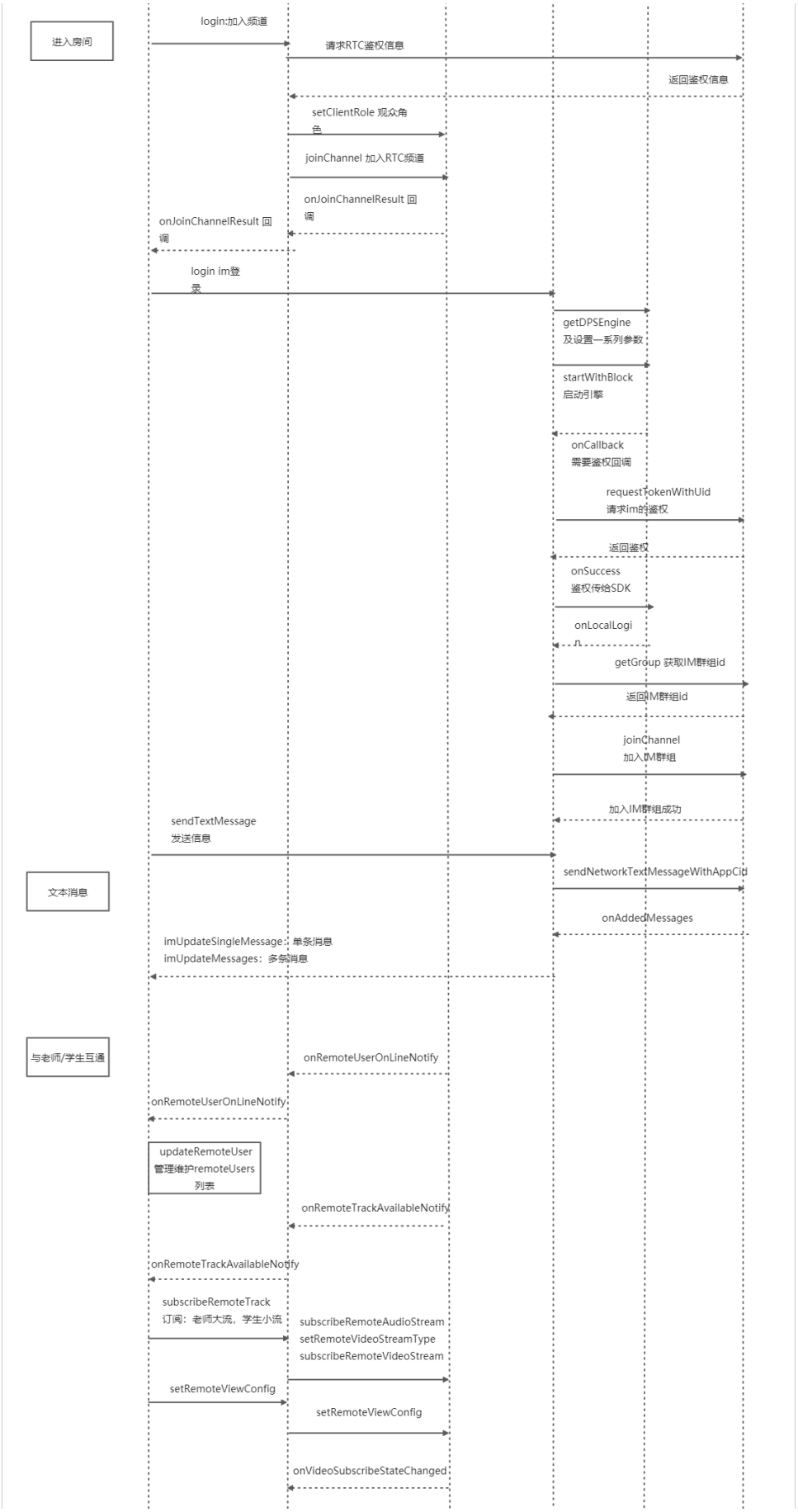

App
AliRtcInteractiveEngine
AliImInteractiveEngine
RTC SDK
AIM SDK
Server
sharedInstance
单例创建RTC实例
AliRtcEngine.getInstance
创建实例
获取实例
setChannelProfile 互动模式
setClientRole设置角色
setAliRtcInteractiveListener
设置监听
setRtcEngineEventListener
setRtcEngineNotify
设置RTC回调事件
sharedInstance
单例创建IM实例
setAliImInteractiveListener
设置监听
加入RTC房间
login
登陆RTC
获取鉴权信息
返回鉴权信息
joinChannel
onJoinChannelResult
onJoinChannelResult

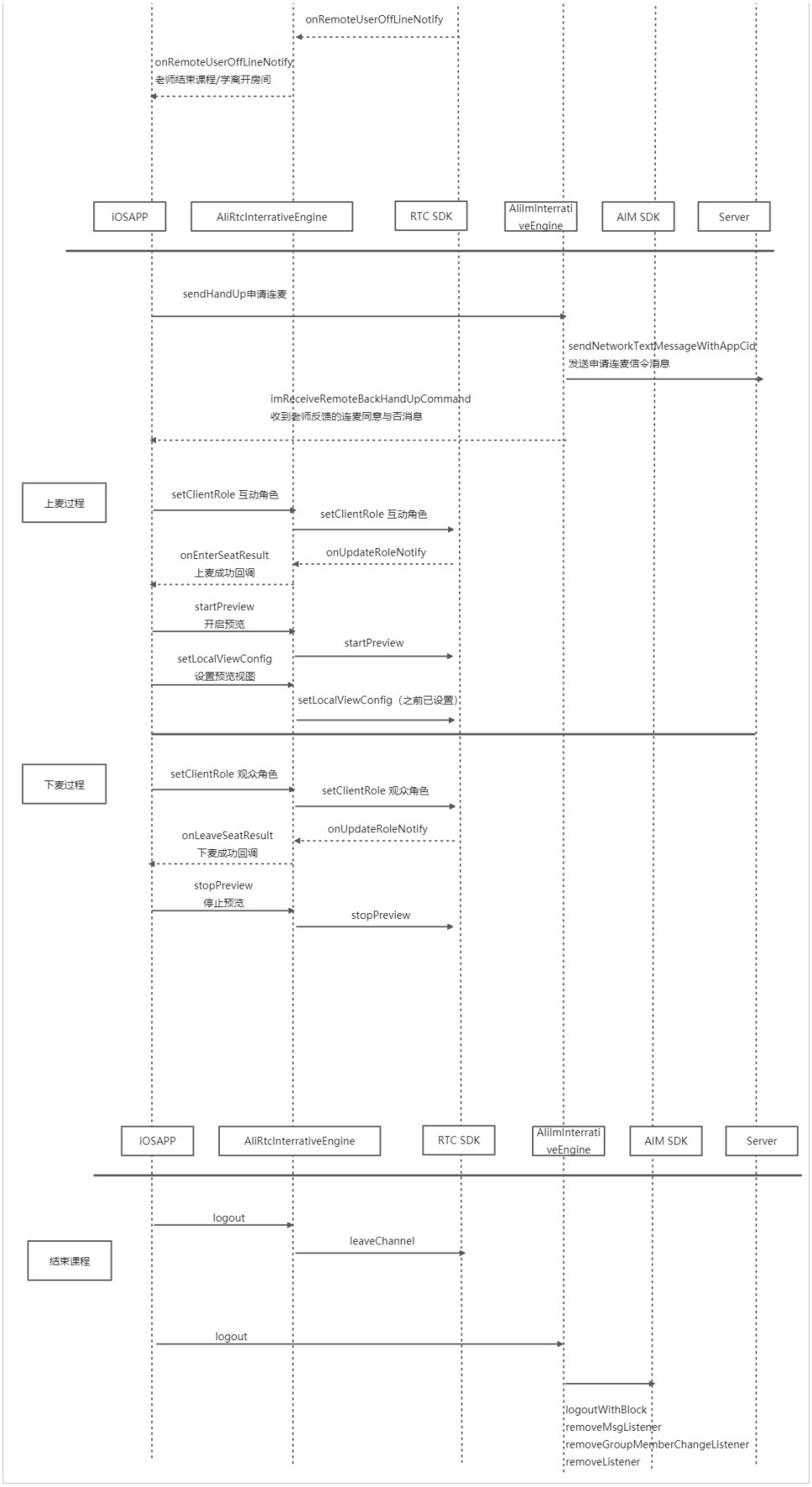

登录IM
createUserManager
创建用户实例并登录
createDPSManager
onSuccess
login登陆IM
onLocalLogin登陆IM成功
onLocalLogin登陆成功
IM加入群聊
addMembers加入群聊
getGroup获取群组ID
onSuccess成功获取群组ID
joinChannel加入群聊AIM
onSuccess
joinGroupSuccess
addGroupMemberChangeListener
注册群组人员变化监听回调
onAddedMembers
维护成员列表
onGroupAddedMembers
群组新增成员
onRemovedMembers
onGroupRemovedMembers
群组移除成员回调
listAllMembers(AIMPubGroupListAllMemberListener)
拉取所有群成员
listAllMembers
拉取群成员
onLocal、onRefresh、onFailure
onLocal、onRefresh、onFailure
onRemoteUserOnlineNotify
TEACHER_ENTER
老师进入
getUserInfo
获取老师信息
获取UserInfo
更新老师信息
与老师互通
onAudioSubscribeStateChanged
SUBSCRIBED_AUDIO
订阅音频流成功
onVideoSubscribeStateChanged
SUBSCRIBED_VIDEO
订阅视频流成功
setRemoteViewConfig
设置远端视频视图 type：1
setRemoteViewConfig
设置远端视频视图
onScreenSubscribeStateChanged
SUBSCRIBED_SCREEN
订阅共享流成功
setRemoteViewConfig
设置远端共享视图 type：2
setRemoteViewConfig
设置远端共享视图
onRemoteUserOfflineNotify
TEACHER_LEAVE
老师离开（结束课程）
onRemoteUserOnlineNotify
STUDENT_ENTER
学生进入（上麦）
getUserInfo 更新用户
getUserInfo 更新用户
与其他上麦学生互通
onAudioSubscribeStateChanged
SUBSCRIBED_AUDIO
订阅音频流成功
onVideoSubscribeStateChanged
SUBSCRIBED_VIDEO
订阅视频流成功
setRemoteViewConfig
设置远端视频视图 type：1
setRemoteViewConfig
设置远端视频视图


onRemoteUserOfflineNotif
y
STUDENT_LEAVE
学生离开（下麦）
举手 申请连麦
handUp 申请连麦
发生申请连麦的消息sendMsg
sendMessageSuccess/sendMessageFailure(成功或失败)
onAddedMessages解析老师是否同意连麦
onAddedMessages
同意连麦
上麦
enterSeat上麦
setClientRole 互动角色
onUpdateRoleNotify
onAudioPublishStateChanged
onVideoPublishStateChanged
onEnterSeatResult
上麦成功回调
startPreview
开启预览
setLocalViewConfig
设置预览视图
startPreview
setLocalViewConfig
下麦
leaveSeat下麦
setClientRole 观众角色
onUpdateRoleNotify
onAudioPublishStateChanged
onVideoPublishStateChanged
onLeaveSeatResult
下麦成功回调
文本消息
sendMessage
发送普通文本信息
sendMsg 发送普通文本信息
onAddedMessages
解析普通文本消息
onAddedMessages
接收普通文本
结束课程
logout 登出教室
stopPreview
leaveChannel 离开频道
destroy销毁IM实例
logout登出IM
onSuccess登出成功
releaseDPSManager销毁IM

- iOS


iOSAPP
AliRtcInterrativeEngine
RTC SDK
AliImInterrativeEngine
AIM SDK
Server
sharedInstance
获取单例创建实例
setH5CompatibleMode
（设置h5兼容模式）
sharedInstance
（创建engine实例）
enableSpeakerphone
（使用扬声器）
返回单例
获取实例


进入房间
login:加入频道
请求RTC鉴权信息
返回鉴权信息
setClientRole 观众角色
joinChannel 加入RTC频道
onJoinChannelResult 回调
onJoinChannelResult 回调
login im登录
getDPSEngine
及设置一系列参数
startWithBlock
启动引擎
onCallback
需要鉴权回调
requestTokenWithUid
请求im的鉴权
返回鉴权
onSuccess
鉴权传给SDK
onLocalLogi
getGroup 获取IM群组id
返回IM群组id
joinChannel
加入IM群组
加入IM群组成功
文本消息
sendTextMessage
发送信息
sendNetworkTextMessageWithAppCid
onAddedMessages
imUpdateSingleMessage：单条消息
imUpdateMessages：多条消息
与老师/学生互通
onRemoteUserOnLineNotify
onRemoteUserOnLineNotify
updateRemoteUser
管理维护remoteUsers
列表
onRemoteTrackAvailableNotify
onRemoteTrackAvailableNotify
subscribeRemoteTrack
订阅：老师大流，学生小流
subscribeRemoteAudioStream
setRemoteVideoStreamType
subscribeRemoteVideoStream
setRemoteViewConfig
setRemoteViewConfig
onVideoSubscribeStateChanged


onRemoteUserOffLineNotify
onRemoteUserOffLineNotify
老师结束课程/学离开房间
iOSAPP
AliRtcInterrativeEngine
RTC SDK
AliImInterrativeEngine
AIM SDK
Server
sendHandUp申请连麦
sendNetworkTextMessageWithAppCid
发送申请连麦信令消息
imReceiveRemoteBackHandUpCommand
收到老师反馈的连麦同意与否消息
上麦过程
setClientRole 互动角色
setClientRole 互动角色
onEnterSeatResult
上麦成功回调
onUpdateRoleNotify
startPreview
开启预览
startPreview
setLocalViewConfig
设置预览视图
setLocalViewConfig（之前已设置）
下麦过程
setClientRole 观众角色
setClientRole 观众角色
onLeaveSeatResult
下麦成功回调
onUpdateRoleNotify
stopPreview
停止预览
stopPreview
iOSAPP
AliRtcInterrativeEngine
RTC SDK
AliImInterrativeEngine
AIM SDK
Server
logout
leaveChannel
结束课程
logout
logoutWithBlock
removeMsgListener
removeGroupMemberChangeListener
removeListener

## 实现流程

实现流程如下图所示：


| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务 | 进行Web端集成之前，您必须开通音视频通信服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |
| 3 | 服务端集成 | 用于解决方案App与服务端之间数据交互、业务逻辑处理等，您可以通过源码快速搭建服务端互动大班课。 |
| 4 | Electron集成 | 目前老师端仅支持Electron集成，您可以通过源码快速搭建Web端互动大班课。 |
| 5 | 移动端集成：<br>• iOS集成<br>• Android集成 | 您可以通过源码快速搭建移动端互动大班课。 |

## Demo体验

您可以通过钉钉扫描以下二维码，下载安装互动大班课Demo体验。

# 2.3.2. 服务端集成

通过阅读本文，您可以了解互动大班课服务端的集成操作。

## 环境要求

开发工具：IntellIj IDEA。

## 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经注册用于服务配置的域名并且已备案。更多信息，请参见域名注册和备案。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 您已经开通了直播互动服务，单击此处申请邀测。
- 环境中已安装Java JDK 8或以上的版本。具体操作，请参见安装JDK。

  > **说明** 如果服务端为Linux环境，推荐安装Oracle JDK，不推荐使用Open JDK进行服务端集成。

- 服务端环境已安装Redis或已经购买阿里云Redis服务。更多信息，请参见Redis概览。

## 操作步骤

1. 获取AppID和AppKey。此处建议记录AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。
   iv. 单击**操作**列的**查询AppKey**，获取AppKey。

   > **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 获取AccessKey。

   > **说明** 阿里云账号（主账号）的AccessKey泄露会威胁您所有资源的安全。从安全考虑，请创建RAM用户（子账号）并获取对应的AccessKey，用于访问您的云资源。

   i. 登录RAM控制台。
   ii. 在左侧导航栏选择**身份管理 > 用户**，进入用户页面。

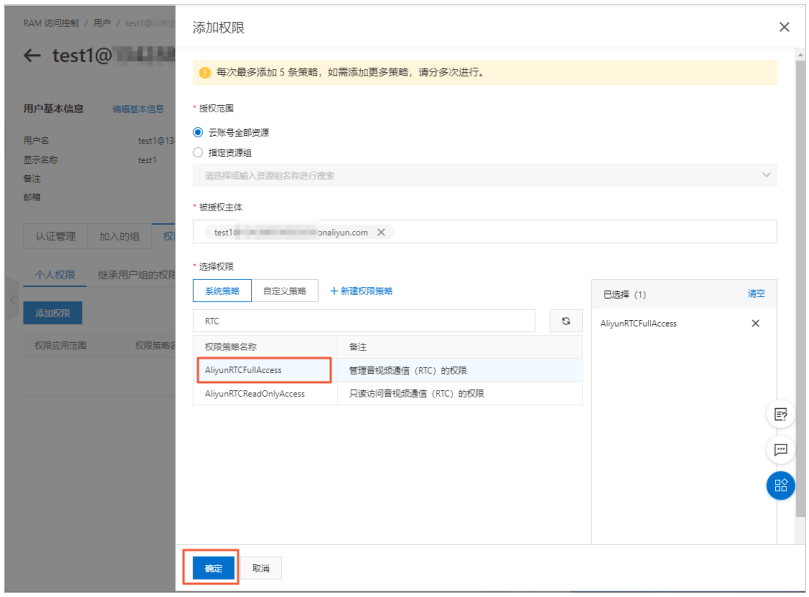
iii. 单击**创建用户**，填写用户账号信息，选中**Open API 调用访问**创建用户。


iv. 单击**确定**。

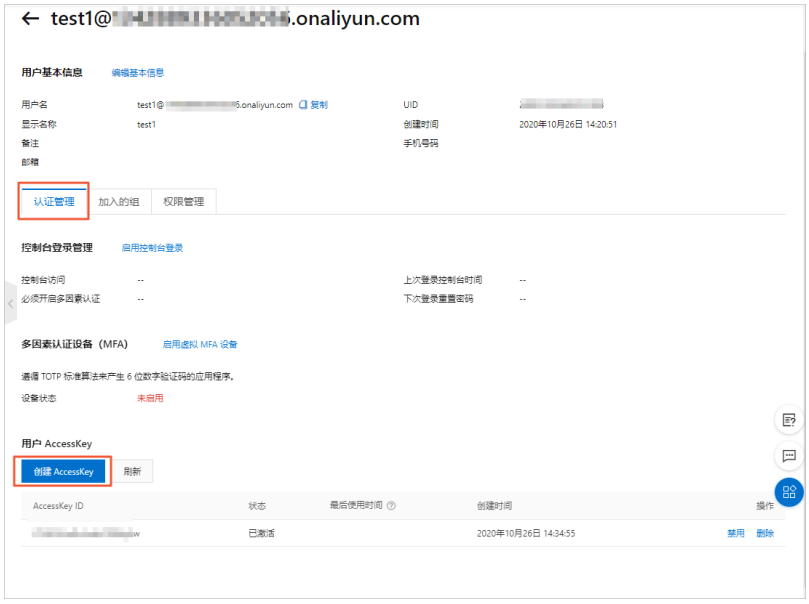
v. 重新返回用户页面。在**用户登录名称/显示名称**列下，单击目标RAM用户（子账号），进入管理页面。


vi. 单击**权限管理**，再单击**添加权限**。

vii. 选择AliyunRTCFullAccess（管理音视频通信的权限，可输入RTC进行搜索）。单击**确定**。



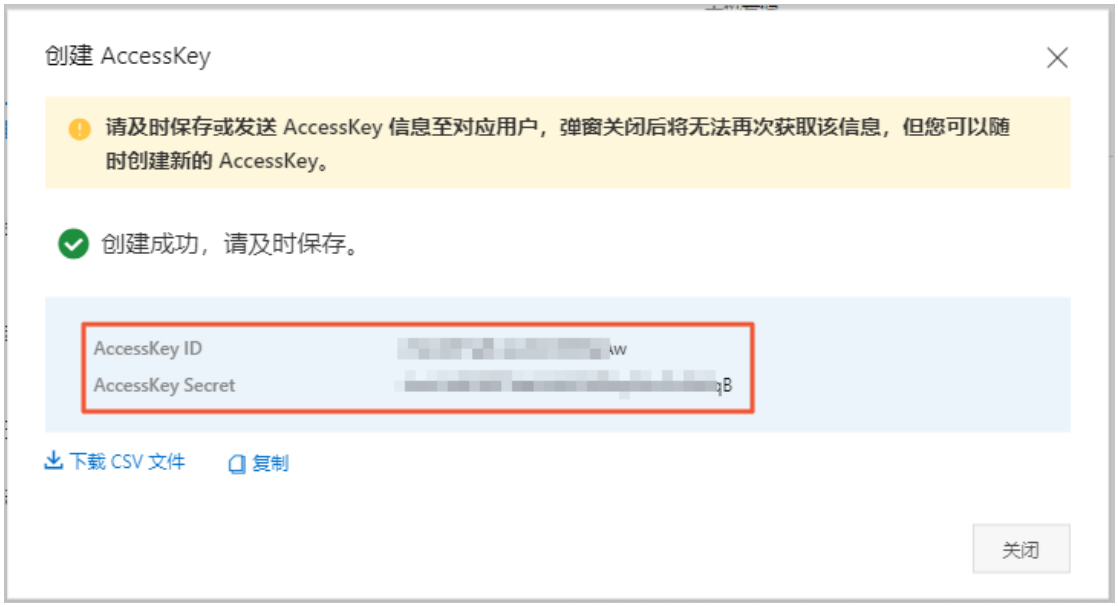

viii. 单击**认证管理**，在用户AccessKey区域单击**创建AccessKey**。




> **说明**
>
> - 首次创建时需填写手机验证码。
> - 如果AccessKey泄露或丢失，则需要创建新的AccessKey，最多可以创建2个AccessKey。

ix. 获取AccessKey ID和AccessKey Secret。

创建AccessKey完成后，会弹出**创建AccessKey**对话框，包含AccessKey ID和AccessKey Secret信息。此处建议记录AccessKey ID和AccessKey Secret，方便后续操作中使用。




3. 下载并解压Demo，更多信息，请参见Demo源码下载。

> **说明**
> - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

4. 使用IntelliJ IDEA打开Demo工程（包含pom.xml的目录）。
5. 修改配置文件application.properties。

```
##服务启动 端口号;
server.port=8080
server.tomcat.accesslog.buffered=false
##日志文件夹配置，可以根据自己的实际情况进行配置;
server.tomcat.accesslog.directory=/home/alivcrtc/liveClassJar
server.tomcat.accesslog.enabled=true
server.servlet.context-path=/interactive-live-class/
## 日志配置，可以根据需要进行修改;
logging.level.com.live.dao=debug
logging.level.com.live.dao.user=debug
logging.level.com.alivc.vod.dao=debug
logging.file:my.log
logging.pattern.file=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread]
%logger{15}.%M : %msg%n
logging.pattern.console=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread]
%logger{15}.%M - %msg%n
##阿里云账号AK，用于服务端调用OpenAPI时候使用;
accessKeyID=*****
accessKeySecret=*****
##RTCappid 和 appkey 在开通RTC服务和创建应用时可以获取;
rtc.liveclass.appId =
*****
rtc.liveclass.appKey = *****
rtc.gslb = https://rgslb.rtc.aliyuncs.com（请不要修改）
##每个房间通话限时 ： <0 不限制，单位：分钟;
durationlimit=-1
#------------ Redis -----------------
spring.redis.host=****
#spring.redis.host=localhost
# Redis 服务器连接端口
spring.redis.port=6379
# Redis 数据库索引（默认为 0）
spring.redis.database=0
# Redis 服务器连接密码（默认为空）
spring.redis.password=****
#连接池最大连接数（使用负值表示没有限制）
spring.redis.jedis.pool.max-active=8
# 连接池最大阻塞等待时间（使用负值表示没有限制）
spring.redis.jedis.pool.max-wait=-1
# 连接池中的最大空闲连接
spring.redis.jedis.pool.max-idle=8
# 连接池中的最小空闲连接
spring.redis.jedis.pool.min-idle=0
## 内部消息系统，在开通互动直播时获取;
IM_AccessKeyId = ***
IM_AccessKeySecret = ***
IM_AppId = ***
```

6. 编译运行。
   - 方法一：

     在IntelliJ IDEA中运行Application.java。
   - 方法二：

通过Maven命令进行编译打包成jar，然后通过**java -jar**运行，具体的命令如下所示：

```
mvn clean package -Dmaven.test.skip
java -jar target/live-class-api.jar
```

7. 获取服务URL。此处建议记录一下，方便后续其它端集成操作中使用。

   服务URL结构为http://{ip}:{port}/{contextPath}/{api}，详细说明如下所示：

   - ip：运行server的服务器或者本地PC的IP地址。
   - port：服务启动端口号，即步骤中 `server.port` 的值。
   - contextPath：步骤中 `server.servlet.context-path` 的值。
   - api：详情请参见API说明。

   例如，服务URL的地址为http://127.0.0.1:8080/interactive-live-class/getRtcAuth。

## Demo目录结构说明

此Demo为标准的通过SpringBoot框架对外提供REST接口的项目，也是一个标准的Maven项目。项目结构如下所示：

| 文件名 | 说明 |
| --- | --- |
| \java\com\alivc\base | 基础工具类。 |
| \java\com\alivc\liveclass | 业务逻辑包，主要包含4个Package：<br>• controller：对外提供的REST接口。<br>• model：业务封装类。<br>• service：和Redis交互的主要业务逻辑。<br>• utils：Redis配置。 |
| resources | SpringBoot工程的配置文件。 |
| webapp | webapp模块。 |

## API说明

RTC相关：

getRtcAuth：下发RTC相关鉴权信息。

IM相关：

- login：下发IM相关鉴权信息给客户端（客户端登录IM系统时需要用到）。
- createGroup：创建一个群组（本项目设计为一个课堂对应一个群组）。
- getGroup：通过课堂ID获取群组ID。
- leaveChannel：离开课堂。
- destoryChannel：删除课堂。
- joinChannel：加入课堂。
- sendMsg：通过服务端发送消息。
- callback：接收IM服务的回调信息。

关于接口的详细信息，请参见Demo中README.md。

# 2.3.3. Electron集成

通过阅读本文，您可以了解互动大班课Electron的集成操作。

## 环境要求

- Electron版本：6.0.10。
- 开发平台：Mac、Windows（建议Windows端使用32位Node文件）。

## 前提条件

环境中已安装Node.js 6.0或以上版本。具体操作，请参见安装Node.js。

## 操作步骤

1. 下载并解压SDK和Demo，更多信息，请参见SDK下载和Demo源码下载。

   > 说明
   > - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 修改*src/renderer/core/data/rtc-config.js*文件，根据服务端集成中获取的服务URL配置`baseUrl`的值。
3. 修改*src/renderer/core/data/aim-config.js*文件，配置IM的`appName`、`appKey`和`baseServer`的值。
   i. 创建IM应用，具体操作，请参见应用管理。
   ii. 获取AppID和AppKey，更多信息，请参见功能配置。
   iii. 根据获取到的AppID、AppKey以及步骤中的`baseUrl`分别配置IM的`appName`、`appKey`和`baseServer`。
4. 修改*package.json*文件，根据Electron SDK实际版本配置`postinstall`的值。

```
"scripts": {
  "build": "node .electron-vue/build.js && electron-builder",
  "build:win32": "node .electron-vue/build.js && electron-builder --win --ia32",
  "build:dir": "node .electron-vue/build.js && electron-builder --dir",
  "build:clean": "cross-env BUILD_TARGET=clean node .electron-vue/build.js",
  "build:web": "cross-env BUILD_TARGET=web node .electron-vue/build.js",
  "dev": "node .electron-vue/dev-runner.js",
  "pack": "npm run pack:main && npm run pack:renderer",
  "pack:main": "cross-env NODE_ENV=production webpack --progress --colors --config .electron-vue/webpack.main.config.js",
  "pack:renderer": "cross-env NODE_ENV=production webpack --progress --colors --config .electron-vue/webpack.renderer.config.js",
  "postinstall": "node node_modules/aliyun-webrtc-electron-sdk/dist/bin/alirtcdown -v 6.0.10"
},
```

   > 说明
   > - `win32`表示Windows系统，如果是Mac系统，用`darwin`表示。
   > - 如果是Windows系统，参数为`ia32`；如果是Mac系统，参数为`x64`。
   > - `postinstall`中当前Electron可选的版本为6.0.10。

5. 运行Demo。
   i. 在*package.json*文件所在的目录下执行以下命令，安装项目依赖。

   ```
   npm install
   ```

ii. 运行项目。运行成功后，浏览器会默认打开示例程序。

**npm run dev**

iii. （可选）打包。

**npm run build**

> **说明**
>
> - 此命令在Mac上运行时会生成dmg包，在windows上运行时会生成exe包。
> - 如果需要在Mac上生成exe包，需要修改*package.json*文件，具体操作，请参见步骤，然后执行npm run build:win32进行打包（首次打包需要下载Windows对应打包环境可能会比较慢，可以选择使用国内镜像解决）。

## Demo目录结构说明

项目结构如下所示：

```
├── .electron-vue
├── build                                                  #打包后文件
├── dist
├── src                                                    #项目文件目录
│   ├── main                                               #主进程
│   │   ├── index.dev.js
│   │   └── index.js                                       #主进程配置
│   ├── renderer                                           #渲染进程
│   │   ├── assets                                         #本地资源
│   │   ├── components                                     #公共组件
│   │   ├── core
│   │   │   ├── client
│   │   │   │   ├── aim-request-path.js                    #aim请求路径
│   │   │   │   ├── aim-message-type.js                    #aim消息类型
│   │   │   │   ├── alirtc-electron-interrative-engine.js  #rtc electron教育场景类
│   │   │   │   └── eventtype.js                           #教育场景类 相关事件
│   │   │   ├── data
│   │   │   │   ├── aim-config.js                          #aim相关配置参数
│   │   │   │   ├── rtc-config.js                          #rtc相关配置参数
│   │   │   │   └── main.js                                #数据类型
│   │   │   ├── http
│   │   │   │   └── http.js                                #请求鉴权相关方法
│   │   │   └── utils
│   │   │   │   ├── array.js                               #数组拓展方法
│   │   │   │   ├── eventdispatcher.js                     #事件抛发
│   │   │   │   └── utils.js                               #其他公共方法
│   │   ├── filters                                        #vue过滤方法
│   │   ├── lib                                            #UI组件库
│   │   ├── plugins
│   │   ├── router                                         #路由
│   │   ├── store                                          #vuex
│   │   ├── views                                          #页面
│   │   │   ├── login
│   │   │   │   └── login.vue                              #登录页面
│   │   │   ├── student
│   │   │   │   └── student.vue                            #学生页面
│   │   │   └── teacher
│   │   │       └── teacher.vue                            #教师页面
│   │   ├── App.vue                                        #根组件
│   │   └── main.js                                        #入口文件
│   └── index.ejs                                          #模版页面
├── static                                                 #静态资源目录
└── package.json
```

# API说明

目录

| API | 描述 |
| --- | --- |
| instance | 获取实例。 |
| loginIm | 登录IM。 |
| logoutIm | 登出IM。 |
| createRoom | 创建IM房间（老师调用）。 |
| enterRoom | 根据角色进入房间。 |
| setClientRole | 设置角色（学生调用）。 |
| setVideoEncoderConfiguration | 设置视频分辨率。 |
| muteLocalMic | 关闭或打开本地麦克风。 |
| muteLocalCamera | 关闭或打开本地摄像头。 |
| startPreview | 开启预览。 |
| stopPreview | 关闭预览。 |
| setLocalViewConfig | 设置本地预览视图。 |
| setRemoteViewConfig | 设置远端渲染视图。 |
| setRemoteVideoStreamType | 设置远端大小流。 |
| getScreenShareSourceInfo | 获取屏幕分享信息（老师调用）。 |
| startScreenShare | 开启屏幕共享（老师调用）。 |
| stopScreenShare | 停止屏幕共享（老师调用）。 |
| getUserInfo | 获取连麦用户信息。 |
| getMemberList | 获取房间成员列表。 |
| exitRoom | 退出房间。 |
| sendTextMessage | 发送普通文本消息。 |
| muteStudentMic | 关闭或打开学生麦克风（老师调用）。 |
| muteStudentCamera | 关闭或打开学生摄像头（老师调用）。 |
| muteAllStudentMic | 关闭或打开所有学生麦克风（老师调用）。 |
| handUp | 申请连麦（学生调用）。 |

| API | 描述 |
| --- | --- |
| backHandUp | 是否允许学生上麦（老师调用）。 |
| on | 监听事件。 |
| off | 取消监听事件。 |
| destroy | 销毁实例。 |

详情

- instance：获取实例。

```
AliRTCElectronInterrativeEngine.instance
```

- loginIm：登录IM。

```
/**
 * 登录IM
 * @param {String} userid RTC鉴权信息中的userid
 * @returns {Promise}
 */
AliRTCElectronInterrativeEngine.instance.loginIm(userid).then(()=>{
    // 登录成功
}).catch((err)=>{
    // 登录失败
})
```

- logoutIm：登出IM。

```
/**
 * 登出IM
 * @returns {Promise}
 */
AliRTCElectronInterrativeEngine.instance.logoutIm().then(()=>{
    // 登出成功
}).catch((err)=>{
    // 登出失败
})
```

- createRoom：创建IM房间（老师调用）。

```
/**
 * 通过RTC信息创建IM房间
 * @param {String} userid RTC 用户ID
 * @param {String} channel RTC 频道号
 * @returns {Promise}
 */
AliRTCElectronInterrativeEngine.instance.createRoom(userid, channel).then(()=>{
    // 创建成功
}).catch((err)=>{
    // 创建失败
})
```

- enterRoom：根据角色进入房间。

```
/**
 * 进入房间
 * @param {Number} role 角色 （1 老师 2 学生）
 * @param {Object} authInfo 鉴权信息
 */
AliRTCElectronInterrativeEngine.instance.enterRoom(role, authInfo)
```

- setClientRole：设置角色（学生调用）。

```
/**
 * 设置上下麦
 * @param {Number} role 角色 （1 上麦 2 下麦）
 * @returns {Number} 0：成功，其他：失败
 */
AliRTCElectronInterrativeEngine.instance.setClientRole(role)
```

> **说明**
> - 切换上下麦后需要等待TO_INTERACTIVE或TO_LIVE回调返回后才成功。
> - 上麦自动开始推音视频流，下麦自动停止推音视频流。

- setVideoEncoderConfiguration：设置视频分辨率。

```
/**
 * 设置视频分辨率
 * @param {Number} width 视频宽度
 * @param {Number} height 视频高度
 */
AliRTCElectronInterrativeEngine.instance.setVideoEncoderConfiguration(width, height)
```

- muteLocalMic：关闭或打开本地麦克风。

```
/**
 * 关闭/开启本地麦克风
 * @param {Boolean} mute 是否静音
 * @returns {Number} 0：成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.muteLocalMic(mute)
```

- muteLocalCamera：关闭或打开本地摄像头。

```
/**
 * 关闭/开启本地摄像头
 * @param {Boolean} mute 是否关闭摄像头
 * @returns {Number} 0：成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.muteLocalCamera(mute)
```

> **说明** 教育场景中关闭本地摄像头会自动关闭预览，开启后会自动打开预览。

- startPreview：开启预览。

```
/**
 * 开启预览
 * @returns {Number} 0：成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.startPreview()
```

> **说明** 需要调用setLocalViewConfig设置视图后才能显示预览画面

- stopPreview：关闭预览。

```
/**
 * 关闭预览
 * @returns {Number} 0：成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.stopPreview()
```

- setLocalViewConfig：设置本地预览视图。

```
/**
 * 设置本地预览视图
 * @param {HTMLDivElement} container
 */
AliRTCElectronInterrativeEngine.instance.setLocalViewConfig(container)
```

- setRemoteViewConfig：设置远端渲染视图。

```
/**
 * 设置远端渲染视图
 * @param {String} userid
 * @param {HTMLDivElement} container
 * @param {Number} type  1：摄像头流  2:共享流
 */
AliRTCElectronInterrativeEngine.instance.setRemoteViewConfig(userid, container, type)
```

- setRemoteVideoStreamType：设置远端大小流。

```
/**
  * 设置远端大小流
  * @param {String} userid 用户ID
  * @param {Number} streamType 1：大流 2：小流
  * @returns
  */
AliRTCElectronInterrativeEngine.instance.setRemoteVideoStreamType(userid, streamType)
```

- getScreenShareSourceInfo：获取屏幕分享信息（老师调用）。

```
/**
 * 获取屏幕分享信息
 * @param {Number} sourceType 0：屏幕信息 1：应用窗口信息
 * @returns {Promise<Array>} sourceList 获取成功返回 分享信息
 */
AliRTCElectronInterrativeEngine.instance.getScreenShareSourceInfo(sourceType).then((sourceList)=>{
    // 获取成功
}).catch(()=>{
    // 获取失败
})
```

返回的信息如下所示：

| 参数名 | 类型 | 描述 |
| --- | --- | --- |
| sourceId | Number | 分享源ID。 |
| sourceTitle | String | 分享源标题。 |
| appIcon | NativeImage | 应用的图标。 |
| thumbnail | NativeImage | 缩略图。 |

- startScreenShare：开启屏幕共享（老师调用）。

```
/**
 * 开启屏幕共享
 * @param {Number} type 0：屏幕信息 1：应用窗口信息
 * @param {Number} sourceId 分享源ID
 * @returns {Number} 0:成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.startScreenShare(type, sourceId)
```

- stopScreenShare：停止屏幕共享（老师调用）。

```
/**
 * 停止屏幕共享
 * @returns {Number} 0:成功 其他：失败
 */
AliRTCElectronInterrativeEngine.instance.stopScreenShare()
```

- getUserInfo：获取连麦用户信息。

```
/**
 * 获取RTC用户信息
 * @param {string} uid 用户ID
 */
AliRTCElectronInterrativeEngine.instance.getUserInfo(userid)
```

- getMemberList：获取房间成员列表。

```
/**
 * 获取IM房间成员列表
 * @param {string} uid 用户ID
 * @returns {Promise<Array>} memberList 获取成功返回成员列表
 */
AliRTCElectronInterrativeEngine.instance.getMemberList().then((memberList)=>{
    // 获取成员列表成功
}).catch(()=> {
    // 获取成员列表失败
})
```

返回的信息如下所示：

| 参数名 | 类型 | 描述 |
| --- | --- | --- |
| userid | String | 用户ID。 |
| isSelf | Boolean | 是否是自己。 |
| isTeacher | Boolean | 是否是老师。 |
| displayName | String | 用户displayName。 |

- exitRoom：退出房间。

```
/**
 * 退出房间 （老师会解散IM群组并退出RTC房间，学生会离开IM群组并退出RTC房间）
 * @returns {Promise<void>}
 */
AliRTCElectronInterrativeEngine.instance.exitRoom().then(()=>{
    // 退出成功
}).catch(()=> {
    // 退出失败
})
```

- sendTextMessage：发送普通文本消息。

```
/**
 * 发送普通文本消息
 * @param {String} message 文本信息
 */
AliRTCElectronInterrativeEngine.instance.sendTextMessage(message)
```

- muteStudentMic：关闭或打开学生麦克风（老师调用）。

```
/**
 * 关闭/打开学生麦克风
 * @param {String} userid 被控制学生的ID
 * @param {Boolean} muteMic 是否关闭麦克风
 */
AliRTCElectronInterrativeEngine.instance.muteStudentMic(userid, muteMic)
```

> 说明
> - 被打开或关闭麦克风的学生会收到TEACHER_CONTROL_MIC回调。
> - 收到该回调后需要调用muteLocalMic接口。

- muteStudentCamera：关闭或打开学生摄像头（老师调用）。

```
/**
 * 关闭/打开学生摄像头
 * @param {String} userid 被控制学生的ID
 * @param {Boolean} muteCamera 是否关闭摄像头
 */
AliRTCElectronInterrativeEngine.instance.muteStudentCamera(userid, muteCamera)
```

> 说明
> - 被打开或关闭麦克风的学生会收到TEACHER_CONTROL_CAMERA回调。
> - 收到该回调后需要调用muteLocalCamera接口。

- muteAllStudentMic：关闭或打开所有学生麦克风（老师调用）。

```
/**
 * 关闭/开启所有学生麦克风
 * @param {Boolean} muteAllMic 是否关闭所有麦克风
 */
AliRTCElectronInterrativeEngine.instance.muteAllStudentMic(muteAllMic)
```

- handUp：申请连麦（学生调用）。

```
/**
 * 举手申请上麦
 * @param {Boolean} handUp 是否举手，当前没有取消举手功能
 */
AliRTCElectronInterrativeEngine.instance.handUp(handUp)
```

- backHandUp：是否允许学生上麦（老师调用）。

```
/**
 * 反馈是否允许学生上麦
 * @param {String} userid 用户ID
 * @param {Boolean} backHandUp 是否同意上麦
 */
AliRTCElectronInterrativeEngine.instance.backHandUp(userid, backHandUp)
```

- on：监听事件。

```
/**
 * 订阅事件
 * @param {string} name 事件名字
 * @param {string} handler 处理事件的方法
 */
AliRTCElectronInterrativeEngine.instance.on(name, handler)
```

- off：取消监听事件。

```
/**
 * 取消订阅事件
 * @param {string} name 事件名字
 * @param {string} handler 处理事件的方法
 */
AliRTCElectronInterrativeEngine.instance.off(name, handler)
```

- destroy：销毁实例。

```
/**
 * 销毁实例（静态方法）
 */
AliRTCElectronInterrativeEngine.destroy()
```

## 回调说明

目录

| 回调 | 描述 |
| --- | --- |
| LOGIN_IM_SUCCESS | 登录IM成功。 |
| JOIN_RESULT | 加入房间回调。 |
| LEAVE_RESULT | 离开房间回调。 |
| TEACHER_ENTER | 老师加入房间（学生订阅）。 |
| TEACHER_LEAVER | 老师离开房间（学生订阅）。 |
| STUDENT_ENTER | 远端学生上麦成功回调。 |
| STUDENT_LEAVE | 远端学生下麦成功回调。 |
| TO_INTERACTIVE | 本地上麦成功回调（学生订阅）。 |
| TO_LIVE | 本地下麦成功回调（学生订阅）。 |
| SUBSCRIBED_AUDIO | 订阅音频成功回调。 |
| UNSUBSCRIBED_AUDIO | 取消订阅音频成功回调。 |
| SUBSCRIBED_VIDEO | 订阅视频成功回调。 |
| UNSUBSCRIBED_VIDEO | 取消订阅视频成功回调。 |
| SUBSCRIBED_SCREEN | 订阅共享成功回调（学生订阅）。 |
| UNSUBSCRIBED_SCREEN | 取消订阅共享流成功回调（学生订阅）。 |
| PUBLISHED_SCREEN | 推送共享流流回调（老师订阅）。 |
| UNPUBLISHED_SCREEN | 取消推送共享流回调（老师订阅）。 |

| 回调 | 描述 |
| --- | --- |
| USER_AUDIO_MUTE | 远端用户开启或关闭麦克风回调。 |
| USER_VIDEO_MUTE | 远端用户开启或关闭摄像头回调。 |
| ERROR | 错误回调。 |
| BYE | 被服务器踢出或频道关闭时回调。 |
| MEMBER_LIST_UPDATE | 成员列表更新回调。 |
| TEXT_MESSAGE_ADD | 收到普通文本信息回调。 |
| TEACHER_CONTROL_MIC | 老师控制本地麦克风回调。 |
| TEACHER_CONTROL_CAMERA | 老师控制本地摄像头回调。 |
| STUDENT_HAND_UP | 学生举手回调。 |
| HAND_UP_RESULT | 老师返回是否允许上麦回调。 |

详情

- LOGIN_IM_SUCCESS：登录IM成功。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.LOGIN_IM_SUCCESS, ()=>{
    // 登录IM成功
})
```

- JOIN_RESULT：加入房间回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.JOIN_RESULT, (result)=>{
    // result: 加入频道结果，成功返回0，失败返回错误码
})
```

- LEAVE_RESULT：离开房间回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.LEAVE_RESULT, (result)=>{
    // result: 离开频道结果 成功返回0，失败返回错误码信息。
})
```

- TEACHER_ENTER：老师加入房间（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TEACHER_ENTER, (userid)=>{
    // userid: 老师的ID
})
```

- TEACHER_LEAVER：老师离开房间（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TEACHER_LEAVE, (userid)=>{
    // userid: 老师的ID
})
```

- STUDENT_ENTER：远端学生上麦成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.STUDENT_ENTER, (userid)=>{
    // userid: 学生的ID
})
```

- STUDENT_LEAVE：远端学生下麦成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.STUDENT_LEAVE, (userid)=>{
    // userid: 学生的ID
})
```

- TO_INTERACTIVE：本地上麦成功回调（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TO_INTERACTIVE, ()=>{
  // 上麦成功，自动推音视频流
})
```

- TO_LIVE：本地下麦成功回调（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TO_LIVE, (userid)=>{
    // 下麦成功，自动停推音视频流
})
```

- SUBSCRIBED_AUDIO：订阅音频成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.SUBSCRIBED_AUDIO, (userid)=>{
    // userid: 被订阅音频用户的ID
})
```

- UNSUBSCRIBED_AUDIO：取消订阅音频成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.UNSUBSCRIBED_AUDIO, (userid)=>{
    // userid: 被取消订阅音频用户的ID
})
```

- SUBSCRIBED_VIDEO：订阅视频成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.SUBSCRIBED_VIDEO, (userid)=>{
    // userid: 被订阅视频用户的ID
})
```

- UNSUBSCRIBED_VIDEO：取消订阅视频成功回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.UNSUBSCRIBED_VIDEO, (userid)=>{
    // userid: 被取消订阅视频用户的ID
})
```

- SUBSCRIBED_SCREEN：订阅共享成功回调（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.SUBSCRIBED_SCREEN, (userid)=>{
    // userid: 被订阅共享用户的ID
})
```

- UNSUBSCRIBED_SCREEN：取消订阅共享流成功回调（学生订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.UNSUBSCRIBED_SCREEN, (userid)=>{
    // userid: 被取消订阅共享用户的ID
})
```

- PUBLISHED_SCREEN：推送共享流流回调（老师订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.PUBLISHED_SCREEN, ()=>{
    // 成功开始推送共享流
})
```

- UNPUBLISHED_SCREEN：取消推送共享流回调（老师订阅）。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.UNPUBLISHED_SCREEN, ()=>{
    // 成功停止推送共享流
})
```

- USER_AUDIO_MUTE：远端用户开启或关闭麦克风回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.USER_AUDIO_MUTE, (userid, isMuted)=
>{
    // userid: 对端ID
  // isMuted: 是否静音了麦克风
})
```

- USER_VIDEO_MUTE：远端用户开启或关闭摄像头回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.USER_VIDEO_MUTE, (userid, isMuted)=
>{
    // userid: 对端ID
  // isMuted: 是否关闭了摄像头
})
```

- ERROR：错误回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.ERROR, (code)=>{
    // code: 错误码
})
```

- BYE：被服务器踢出或频道关闭时回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.BYE, (code)=>{
    // code: 0: 被踢出  1:频道关闭  2:账号在另一地点登录
})
```

- MEMBER_LIST_UPDATE：成员列表更新回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.MEMBER_LIST_UPDATE, ()=>{
    // 使用getMemberList接口获取最新成员列表
  // 该成员列表指IM群组成员列表
})
```

- TEXT_MESSAGE_ADD：收到普通文本信息回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TEXT_MESSAGE_ADD, (data)=>{
    // data.userid: 用户ID
  // data.isSelf: 是否是自己发送的消息
  // data.isTeacher: 是否是老师发送的消息
  // data.displayName: 用户displayName
  // data.message: 信息内容
})
```

- TEACHER_CONTROL_MIC：老师控制本地麦克风回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TEACHER_CONTROL_MIC, (muteMic)=>{
    // muteMic：是否关闭麦克风
})
```

- TEACHER_CONTROL_CAMERA：老师控制本地摄像头回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.TEACHER_CONTROL_CAMERA, (muteCamera
)=>{
    // muteCamera：是否关闭摄像头
})
```

- STUDENT_HAND_UP：学生举手回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.STUDENT_HAND_UP, (data)=>{
    // data.userid：举手学生的ID
  // data.displayName：举手学生的displayName
  // data.handUp：是否举手
})
```

- HAND_UP_RESULT：老师返回是否允许上麦回调。

```
AliRTCElectronInterrativeEngine.instance.on(EventType.HAND_UP_RESULT, (backHandUp)=>{
    // backHandUp：是否同意上麦
})
```

# 2.3.4. iOS集成

通过阅读本文，您可以了解到iOS端互动大班课的集成方法。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 老师端（Electron）已集成并开启。具体操作，请参见Electron集成。
- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

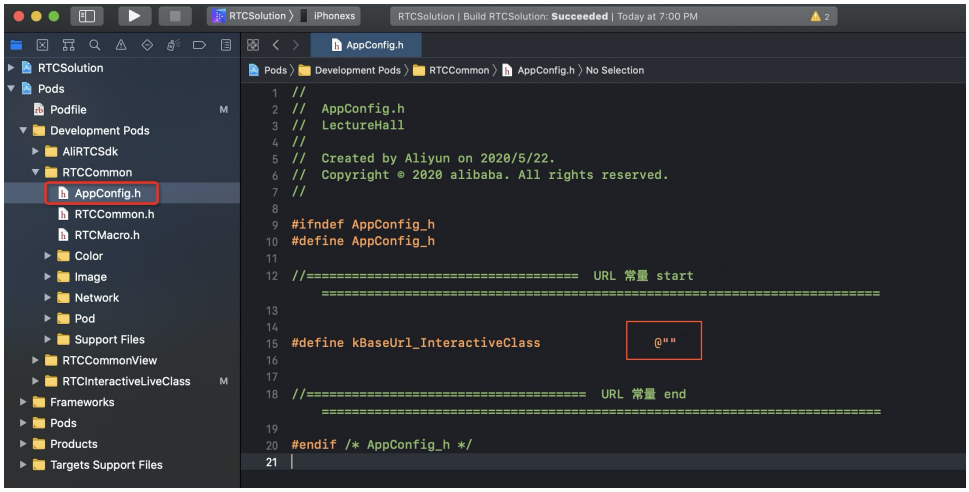
## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > 说明
   > - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

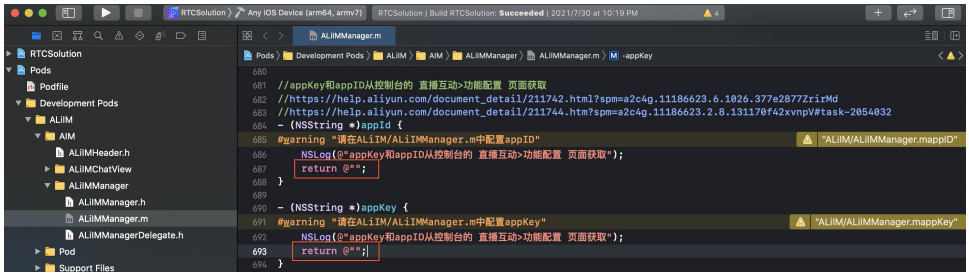
2. 获取RTC鉴权。

   RTC登录鉴权是通过服务端获取的，需要在图中的位置输入Server地址：




> **说明**
>
> - 此处Server地址和Electron端的 `baseServer` 保持一致，更多信息，请参见Electron集成。
> - Server地址末尾需要加/，例如：https://www.aliyun.com/。

3. 配置IM的AppID和AppKey。




> **说明** 此处的AppID、AppKey和Electron端保持一致，更多信息，请参见Electron集成。

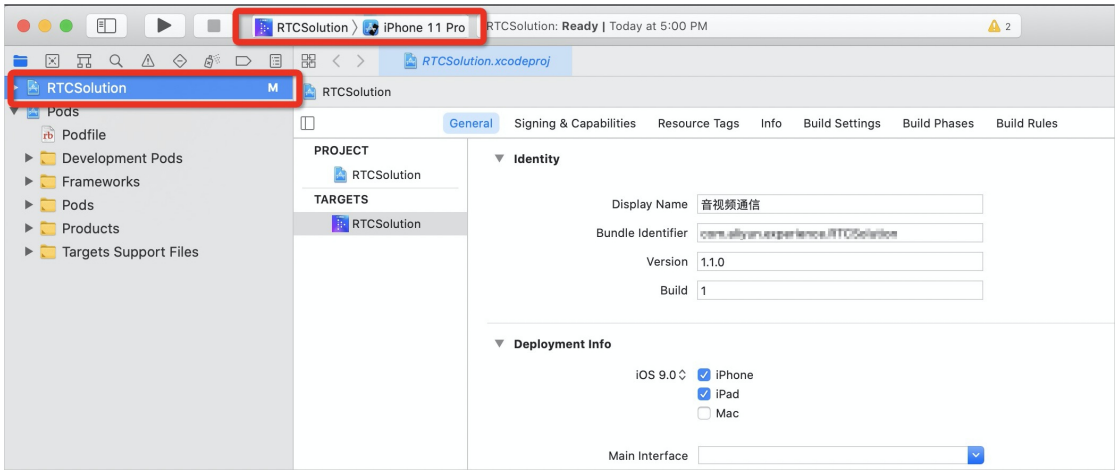
4. 在Podfile文件所在的目录执行以下命令安装功能模块。

   **pod install**

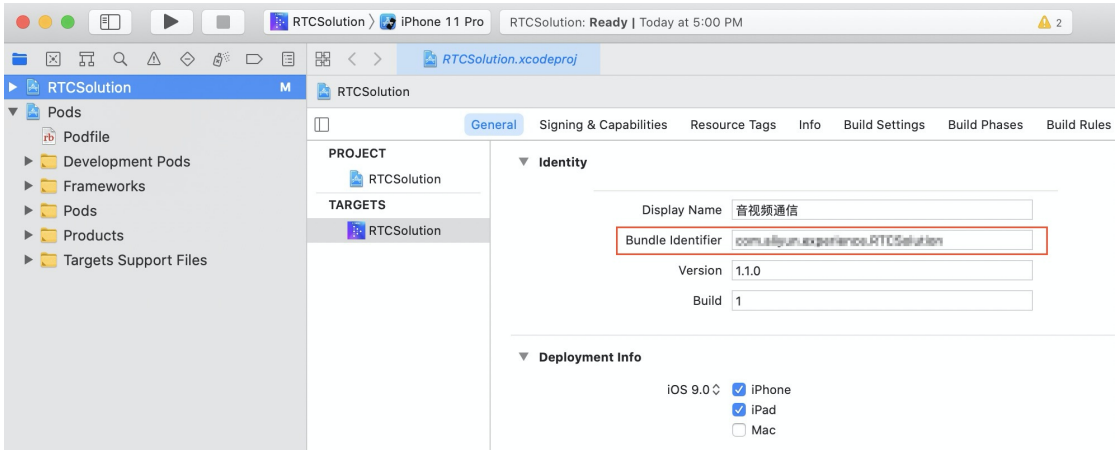
5. 运行Demo。

   i. 使用Xcode打开 *\RTCSolution\RTCSolution.xcworkspace*工程文件。

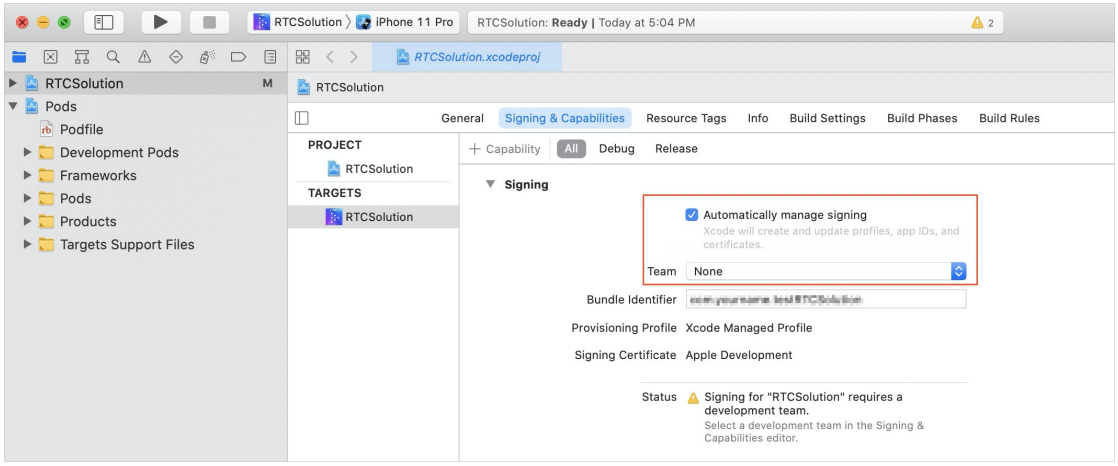
ii. 选择运行的Target为RTCSolution，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。




iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。

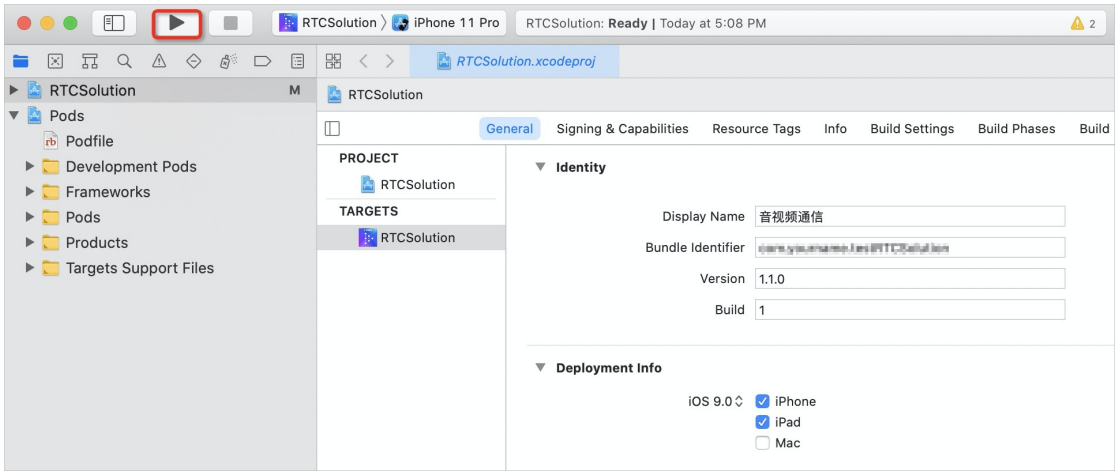



iv. 单击**Signing & Capabilities**页签，选中**Automatically manage signing**，然后单击**Team**下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择**Add an Account...**，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击▶，编译并运行。




如果在编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。

## Demo目录结构说明

RTC把业务代码封装到RTCInteractiveLiveClass-RTC的库中，因此只需在Podfile中指定RTCInteractiveLiveClass-RTC库的路径，RTCInteractiveLiveClass-RTC就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
pod 'RTCInteractiveLiveClass', :path => '../RTCInteractiveLiveClass-RTC'
    ##   pod 'RTCInteractiveLiveClass'  说明项目依赖RTCInteractiveLiveClass库
    ##   path => '../RTCInteractiveLiveClass-RTC'  指明RTCInteractiveLiveClass库的位置（相对于Podfile）
```

此外，工程中还依赖的其他组件如下所示：

```
target 'RTCSolution' do
  # Uncomment the next line if you're using Swift or would like to use dynamic frameworks
  #   use_frameworks!
  # Pods for RTCSolution
  #begin
  #IMSDK 本地集成
  pod 'libdps', :path => 'AIMSDK-3.0.0.23/'
  pod 'libaim', :path => 'AIMSDK-3.0.0.23/'
  pod 'gaea', :path => 'AIMSDK-3.0.0.23/'
  pod 'dmojo_support', :path => 'AIMSDK-3.0.0.23/'
  pod 'xlite', :path => 'AIMSDK-3.0.0.23/'
  pod 'OpenSSL', :path => 'AIMSDK-3.0.0.23/'
  #IM 快速集成使用组件（只需要填写appKey和appId）
  pod 'ALiIM', :path => '../ALiIM'
  pod 'RTCCommon', :path => '../RTCCommon'
  pod 'RTCCommonView', :path => '../RTCCommonView'
  #大班课核心代码
  pod 'RTCInteractiveLiveClass', :path => '../RTCInteractiveLiveClass-RTC'
  #RTCSDK 本地集成
  pod 'AliRTCSdk',         :path => 'AliRtcSDK_iOS'
  pod 'AFNetworking',      :git => 'https://github.com/AFNetworking/AFNetworking.git',  :tag
=>'4.0.1'
end
```

RTC和IM管理类的说明如下所示：

| 类别 | 文件名 | 说明 |
| --- | --- | --- |
| RTC | AliRtcInterrativeEngine.h | RTC对外接口。 |
| | AliRtcInterrativeEngine.m | 接口内部实现。 |
| | AliRtcInterrativeEngineDelegate.h | 回调接口。 |
| IM | AliImInterrativeEngine.h | IM对外接口。 |
| | AliImInterrativeEngine.m | 接口内部实现。 |
| | AliImInterrativeEngineDelegate.h | 回调接口。 |

## API说明

RTC管理类功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取RTCSuperClassImpl的实例对象，初始化RTC SDK。 |
| destorySharedInstance | 销毁RTCSuperClassImpl的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。 |
| login | 根据输入的房间号、用户名加入RTC频道。 |

| API | 描述 |
| --- | --- |
| logout | 退出RTC频道。 |
| enterSeat | 连麦。 |
| leavelSeat | 断开连麦。 |
| muteLocalMic | 是否停止本地音频采集。 |
| muteLocalCamera | 是否停止本地视频采集。 |
| setLocalViewConfig | 为本地预览设置参数及绘制窗口。 |
| startPreview | 开始本地预览。 |
| stopPreview | 停止本地预览。 |
| configRemoteTrack | 设置是否拉取相机、屏幕、音频流，必须调用subscribe才能生效。 |
| subscribe | 手动拉视频和音频流。 |
| setRemoteViewConfig | 为远端的视频设置参数及绘制窗口。 |
| setDelegate | 设置代理。 |
| switchCamera | 切换前后摄像头。 |
| subscribeRemoteTrack | 设置是否拉取远端用户的流（相机流、屏幕流、音频流）。 |
| getDisplayNameWithUid | 获取用户名。 |

RTC回调接口

| API | 描述 |
| --- | --- |
| onEnterSeatResult | 用户上麦的通知。 |
| onLeaveSeatResult | 用户下麦的通知。 |
| onOccurError | 错误信息的通知。 |
| onOccurWarning | 警告信息的通知。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onSDKError | SDK发生异常需要销毁实例。 |
| onJoinChannelResult | 加入房间的通知。 |
| onLeaveChannelResult | 离开房间的通知。 |
| onRemoteTrackAvailableNotify | 远端用户音视频流发生变化时的回调。 |

| API | 描述 |
| --- | --- |
| onSubscribeChangedNotify | 订阅情况发生变化时的回调。 |
| onUserAudioMuted | 用户取消音频的通知。 |
| onUserVideoMuted | 用户取消视频的通知。 |
| onNetworkQualityChanged | 网络质量变化时的回调。 |
| onUpdateRoleNotify | 角色切换成功的通知。 |
| onRemoteUserOnLineNotify | 远端用户上线的通知。 |
| onRemoteUserOffLineNotify | 远端用户下线的通知。 |
| onAudioVolumeCallback | 订阅的音频音量、语音状态、UID。 |

IM功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取IM管理类单例。 |
| destroySharedInstance | 销毁IM管理类。 |
| login | 登录频道。 |
| logout | 退出登录。 |
| sendTextMessage | 发送文本消息。 |
| sendHandUp | 发送举手。 |
| getGroupMessages | 获取群消息。 |

IM回调接口

| API | 描述 |
| --- | --- |
| imUpdateMessages | 消息更新（包含从服务器拉取消息及新增消息）。 |
| imUpdateSingleMessage | 更新单条消息。 |
| imGroupMembersChanged | 群成员加入离开变化更新。 |
| imReceiveRemoteBackHandUpCommand | 收到老师反馈举手信令。 |
| imReceiveRemoteMuteMicCommand | 收到静音指令。 |

| API | 描述 |
| --- | --- |
| imReceiveRemoteMuteCameraCommand | 收到相机流指令。 |
| imReceiveRemoteMuteAllMicCommand | 收到全员静音指令。 |
| imSendMessageFailure | 发送消息失败通知。 |

RTC管理类功能实现接口

- sharedInstance：获取RTCSuperClassImpl的实例对象，初始化RTC SDK。

```
/**
     * 获取单例
     */
   [RTCManager sharedInstance];
```

- destorySharedInstance：销毁RTCSuperClassImpl的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/// 销毁RTC SDK
- (void)destroySharedInstance;
```

- login：根据输入的房间号、用户名加入RTC频道。

```
/// 加入频道
/// @param channel 频道
/// @param userName 用户昵称
- (void)login:(NSString *)channel userName:(NSString *)userName;
```

- logout：退出RTC频道。

```
/// 离开频道
- (void)logout;
```

- enterSeat：连麦。

```
/// 连麦
- (int)enterSeat;
```

- leavelSeat：断开连麦。

```
//断开连麦
- (void)leaveSeat;
```

- muteLocalMic：是否停止本地音频采集。

```
/**
* @brief mute或unmute本地音频采集
* @param mute  YES表示本地音频采集空帧；NO表示恢复正常
* @note mute是指采集和发送静音帧。采集和编码模块仍然在工作
* @return 0表示成功放入队列，-1表示被拒绝
*/
- (int)muteLocalMic:(BOOL)mute;
```

- muteLocalCamera：是否停止本地视频采集。

```
/**
* @brief 是否将停止本地视频采集
* @param mute     YES表示停止视频采集；NO表示恢复正常
* @param track    需要停止采集的track
* @return 0表示Success 非0表示Failure
* @note 发送黑色的视频帧。本地预览也呈现黑色。采集，编码，发送模块仍然工作，只是视频内容是黑色帧
*/
- (int)muteLocalCamera:(BOOL)mute forTrack:(AliRtcVideoTrack)track;
```

- setLocalViewConfig：为本地预览设置参数及绘制窗口。

```
/**
* @brief 为本地预览设置参数以及绘制窗口
* @param viewConfig 包含了窗口以及渲染方式
* @param track      must be AliVideoTrackCamera
* @return 0表示Success，非0表示Failure
* @note 支持joinChannel之前和之后切换窗口。如果viewConfig或者viewConfig中的view为nil，则停止渲染
* 如果在播放过程中需要重新设置render mode，请保持canvas中其他成员变量不变，仅修改renderMode
* 如果在播放过程中需要重新设置mirror mode，请保持canvas中其他成员变量不变，仅修改mirrorMode
*/
- (int)setLocalViewConfig:(AliVideoCanvas *)viewConfig forTrack:(AliRtcVideoTrack)track;
```

- startPreview：开始本地预览。

```
/**
* @brief 开始本地预览
* @return 0表示Success 非0表示Failure
* @note 如果没有设置view，则无法预览。可以在joinChannel之前就开启预览
*       会自动打开摄像头
*/
- (int)startPreview;
```

- stopPreview：停止本地预览。

```
/**
* @brief 停止本地预览
* @return 0表示Success 非0表示Failure
* @note leaveChannel会自动停止本地预览
*       会自动关闭摄像头 (如果正在publish camera流，则不会关闭摄像头)
*/
- (int)stopPreview;
```

- configRemoteTrack：设置是否拉取相机、屏幕、音频流，必须调用subscribe才能生效。

```
/**
 * @brief 设置是否拉取相机、屏幕、音频流
 * @param uid userId 从App server分配的唯一标示符。
 * @param master  是否优先拉取大流
 * @param enable  YES：拉取；NO：不拉取
 * @note 可以在joinChannel之前或者之后设置。如果已经订阅该用户的流，需要调用subscribe:(NSString *)uid onResult:才生效
 */
- (void)configRemoteTrack:(NSString *)uid preferMaster:(BOOL)master enable:(BOOL)enable;
```

- subscribe：手动拉视频和音频流。

```
/**
* @brief 手动拉视频和音频流
* @param uid        User ID。不允许为nil
* @param onResult   当subscribe执行结束后调用这个回调
* @note 如果需要手动选择拉取的流，调用configRemoteAudio, configRemoteTrack,
*       configRemoteScreenTrack来设置。缺省是拉取audio和camera track
*       如果需要unsub所有的流，先通过configRemoteAudio, configRemoteTrack,
*       configRemoteScreenTrack来清除设置，然后调用subscribe
*/
- (void)subscribe:(NSString *)uid onResult:(void (^)(NSString *uid, AliRtcVideoTrack vt, AliRtcAudioTrack at))onResult;
```

- setRemoteViewConfig：为远端的视频设置参数及绘制窗口。

```
/**
* @brief 为远端的视频设置参数及绘制窗口
* @param canvas canvas包含了窗口以及渲染方式
* @param uid    User ID。从App server分配的唯一标示符
* @param track  需要设置的track
* @return 0表示Success，非0表示Failure
* @note 支持joinChannel之前和之后切换窗口。如果canvas为nil或者view为nil，则停止渲染相应的流
*       如果在播放过程中需要重新设置render mode，请保持canvas中其他成员变量不变，仅修改renderMode
*       如果在播放过程中需要重新设置mirror mode，请保持canvas中其他成员变量不变，仅修改mirrorMode
*/
- (int)setRemoteViewConfig:(AliVideoCanvas *)canvas uid:(NSString *)uid forTrack:(AliRtcVideoTrack)track;
```

- setDelegate：设置代理。

```
/// 设置代理
/// @param delegate 代理对象
- (void)setDelegate:(id<RTCInteractiveClassDelegate>)delegate;
```

- switchCamera：切换前后摄像头。

```
/**
* @brief 切换前后摄像头
* @return 0表示Success 非0表示Failure
* @note 只有iOS和android提供这个接口
*/
- (int)switchCamera;
```

- subscribeRemoteTrack：设置是否拉取远端用户的流（相机流、屏幕流、音频流）。

```
/**
 * @brief 设置是否拉取远端用户的流（相机流，屏幕流，音频流）
 * @param uid userId 从App server分配的唯一标示符。
 * @param master  是否优先拉取大流
 * @param enable  YES：拉取；NO：不拉取
 * @note 可以在joinChannel之前或者之后设置。
 */
- (void)subscribeRemoteTrack:(NSString *)uid preferMaster:(BOOL)master enable:(BOOL)enabl
e;
```

- getDisplayNameWithUid：获取用户名。

```
/// 获取用户名
/// @param userid userId
- (NSString *)getDisplayNameWithUid:(NSString *)userid;
```

RTC回调接口

- onEnterSeatResult：用户上麦的通知。

```
/// 自己上麦通知
/// @param errorCode 结果
- (void)onEnterSeatResult:(int)errorCode;
```

- onLeaveSeatResult：用户下麦的通知。

```
/// 自己下线通知
/// @param errorCode 结果
- (void)onLeaveSeatResult:(int)errorCode;
```

- onOccurError：错误信息的通知。

```
/**
 * @brief 如果engine出现error，通过这个回调通知app
 * @param error  Error type
 */
- (void)onOccurError:(int)error;
```

- onOccurWarning：警告信息的通知。

```
/**
* @brief 如果engine出现warning，通过这个回调通知app
* @param warn  Warning type
*/
- (void)onOccurWarning:(int)warn;
```

- onRoomDestroy：房间被销毁的回调。

```
/// 房间被销毁通知
- (void)onRoomdestroy;
```

- onSDKError：SDK发生异常需要销毁实例。

```
/**
* @brief 如果engine出现严重error，通过这个回调通知app
* @param error  错误类型
*/
- (void)onSDKError:(int)error;
```

- onJoinChannelResult：加入房间的通知。

```
/**
* @brief 加入频道结果
* @param result 加入频道结果，成功返回0，失败返回错误码
* @note 此回调等同于joinChannel接口的block，二者择一处理即可
*/
- (void)onJoinChannelResult:(int)result onJoinChannelResult:(AliRtcAuthInfo *)authInfo;
```

- onLeaveChannelResult：离开房间的通知。

```
/**
* @brief 离开频道结果
* @param result 离开频道结果，成功返回0，失败返回错误码
* @note 调用leaveChannel接口后返回，如果leaveChannel后直接调用destroy，将不会收到此回调
*/
- (void)onLeaveChannelResult:(int)result;
```

- onRemoteTrackAvailableNotify：远端用户音视频流发生变化时的回调。

```
/**
* @brief 当远端用户的流发生变化时，返回这个消息
* @note 远方用户停止推流，也会发送这个消息
*/
- (void)onRemoteTrackAvailableNotify:(NSString *)uid onRemoteTrackAvailableNotify:(AliRtc
AudioTrack)audioTrack videoTrack:(AliRtcVideoTrack)videoTrack;
```

- onSubscribeChangedNotify：当订阅情况发生变化时的回调。

```
/**
 * @brief 当订阅情况发生变化时，返回这个消息
 */
- (void)onSubscribeChangedNotify:(NSString *)uid onSubscribeChangedNotify:(AliRtcAudioTra
ck)audioTrack videoTrack:(AliRtcVideoTrack)videoTrack;
```

- onUserAudioMuted：用户取消音频的通知

```
/**
 * @brief 用户muteAudio通知
 * @param uid 执行muteAudio的用户
 * @param isMute YES:静音 NO:未静音
 */
- (void)onUserAudioMuted:(NSString *)uid onUserAudioMuted:(BOOL)isMute;
```

- onUserVideoMuted：用户取消视频通知。

```
/**
 * @brief 用户muteVideo通知
 * @param uid 执行muteVideo的用户
 * @param isMute YES:推流黑帧 NO:正常推流
 */
- (void)onUserVideoMuted:(NSString *)uid videoMuted:(BOOL)isMute;
```

- onNetworkQualityChanged：网络质量变化时的回调。

```
/**
 * @brief 网络质量变化时发出的消息
 * @param uid 网络质量发生变化的uid
 * @param upQuality  上行网络质量
 * @param downQuality  下行网络质量
 * @note 当网络质量发生变化时触发，uid为@""时代表self的网络质量变化
 */
- (void)onNetworkQualityChanged:(NSString *)uid
                onNetworkQualityChanged:(AliRtcNetworkQuality)upQuality
              downNetworkQuality:(AliRtcNetworkQuality)downQuality;
```

- onUpdateRoleNotify：角色切换成功的通知。

```
/**
 * @brief 当用户角色发生变化化时通知
 * @param oldRole 变化前角色类型
 * @param newRole 变化后角色类型
 * @note 调用setClientRole方法切换角色成功时触发此回调
 */
- (void)onUpdateRoleNotifyWithOldRole:(AliRtcClientRole)oldRole newRole:(AliRtcClientRole
)newRole;
```

- onRemoteUserOnLineNotify：远端用户上线的通知。

```
/// 远端用户上线通知
/// @param uid 用户id
- (void)onRemoteUserOnLineNotify:(NSString *)uid;
```

- onRemoteUserOffLineNotify：远端用户下线的通知。

```
/// 远端用户下线通知
/// @param uid 用户id
- (void)onRemoteUserOffLineNotify:(NSString *)uid;
```

- onAudioVolumeCallback：订阅的音频音量、语音状态、UID。

```
/**
 * @brief 订阅的音频音量，语音状态和UID
 * @param array 表示回调用户音量信息数组，包含用户UID,语音状态以及音量，UID为"0"表示本地说话人。
 * @param totalVolume 混音后的总音量，范围[0,255]。在本地用户的回调中，totalVolume;为本地用户混
音后的音量；在远端用户的回调中，totalVolume; 为所有说话者混音后的总音量
 */
- (void)onAudioVolumeCallback:(NSArray <AliRtcUserVolumeInfo *> *_Nullable)array totalVol
ume:(int)totalVolume;
```

IM功能实现接口

- sharedInstance：获取IM管理类单例。

```
/// 获取IM管理类单例
+ (AliImInterrativeEngine *)sharedInstance;
```

- destroySharedInstance：销毁IM管理类。

```
/// 销毁IM管理类
+ (void)destroySharedInstance;
```

- login：登录频道。

```
/// 登录
/// @param uid 用户ID
/// @param channelId 频道号ID
/// @param name 用户昵称
- (void)login:(NSString *)uid channel:(NSString *)channelId displayName:(NSString *)name;
```

- logout：退出登录。

```
/// 退出登录
- (void)logout;
```

- sendTextMessage：发送文本消息。

```
/// 发送文本消息
/// @param message 发送文本
- (void)sendTextMessage:(NSString *)message;
```

- sendHandUp：发送举手。

```
/// 发送举手
/// @param mute 是否举手(YES:举手,目前仅支持YES)
- (void)sendHandUp:(BOOL)mute;
```

- getGroupMessages：获取群消息。

```
/// 获取群消息
- (NSArray *)getGroupMessages;
```

IM回调接口

- imUpdateMessages：消息更新（包含从服务器拉取消息及新增消息）。

```
/// 消息更新 包含从服务器拉取消息及新增消息
/// @param messages 消息（全部）
- (void)imUpdateMessages:(NSArray<LCIMMessage *> *)messages;
```

- imUpdateSingleMessage：更新单条消息。

```
/// 更新单条消息
/// @param msg 新消息
- (void)imUpdateSingleMessage:(LCIMMessage *)msg;
```

- imGroupMembersChanged：群成员加入离开变化更新。

```
/// 群成员加入离开变化更新
/// @param members 成员
- (void)imGroupMembersChanged:(NSArray<AIMPubGroupMember *> *)members;
```

- imReceiveRemoteBackHandUpCommand：收到老师反馈举手信令。

```
/// 收到老师反馈举手信令
/// @param mute 是否同意举手(YES:同意举手,NO:拒绝举手)
/// @param uid 需执行的用户id
- (void)imReceiveRemoteBackHandUpCommand:(BOOL)mute userId:(NSString *)uid;
```

- imReceiveRemoteMuteMicCommand：收到静音指令。

```
/// 收到静音指令
/// @param mute 是否静音 (YES:关闭麦克风，NO:打开麦克风)
/// @param uid 需执行的用户id
- (void)imReceiveRemoteMuteMicCommand:(BOOL)mute userId:(NSString *)uid;
```

- imReceiveRemoteMuteCameraCommand：收到相机流指令。

```
/// 收到相机流指令
/// @param mute 是否关闭相机流 (YES:关闭相机流，NO:打开相机流)
/// @param uid 需执行的用户id
- (void)imReceiveRemoteMuteCameraCommand:(BOOL)mute userId:(NSString *)uid;
```

- imReceiveRemoteMuteAllMicCommand：收到全员静音指令。

```
/// 收到全员静音指令
/// @param mute 是否全员静音 (YES:全员静音，NO:解除全员静音)
- (void)imReceiveRemoteMuteAllMicCommand:(BOOL)mute;
```

- imSendMessageFailure：发送消息失败通知。

```
/// 发送消息失败通知
- (void)imSendMessageFailure:(NSString *)error;
```

# 2.3.5. Android集成

通过阅读本文，您可以了解到Android端互动大班课的集成方法。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 老师端（Electron）已集成并开启。具体操作，请参见Electron集成。
- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Android Studio 3.0或以上版本。更多信息，请参见Android Studio。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

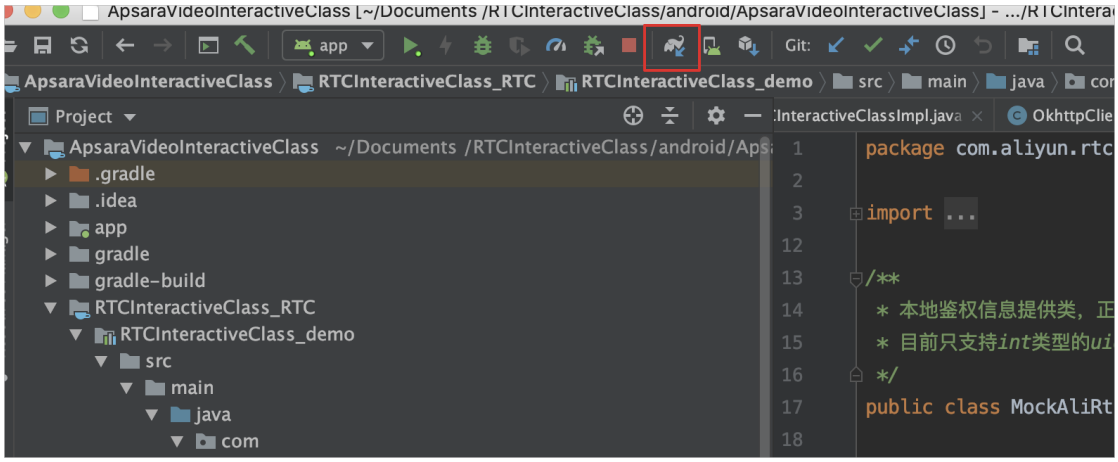
> **说明**
>
> - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 修改*android/ApsaraVideoInteractiveClass/RTCInteractiveClass_RTC/RTCInteractiveClass_demo/src/main/java/com/aliyun/rtc/rtcinteractiveclass/constant/Constant.java*文件，配置AppID、AppKey和服务端URL。

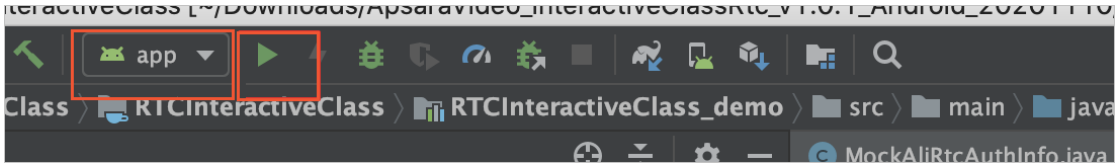
```
package com.aliyun.rtc.rtcinteractiveclass.constant;

public class Constant {

    /**
     * 设置必要信息，appKey和appID从AIM控制台的 直播互动>功能配置 页面获取
     * 实际开发中不建议将appKey和appId写在本地，容易被盗取信息造成流量的损失
     * 参考链接 https://help.aliyun.com/document_detail/272591.html?spm=a2c4g.11174283.6.546.4b004c07jhqsGe
     */
    public static final String AIM_APPKEY = ;

    /**
     * 设置必要信息，appKey和appID从AIM控制台的 直播互动>功能配置 页面获取
     * 实际开发中不建议将appKey和appId写在本地，容易被盗取信息造成流量的损失
     * 参考链接 https://help.aliyun.com/document_detail/272591.html?spm=a2c4g.11174283.6.546.4b004c07jhqsGe
     */

    public static final String AIM_APPID = ;

    public static final String NEW_TOKEN_PARAMS_KEY_USERID = "userName";
    public static final String NEW_TOKEN_PARAMS_KEY_PLATFORM = "platform";
    public static final String NEW_TOKEN_PARAMS_VALUE_PLATFORM = "android";
    public static final String NEW_GUEST_PARAMS_KEY_PACKAGE = "PACKAGE_NAME";
    public static final String NEW_TOKEN_PARAMS_KEY_CHANNELID = "channelId";
    public static final String ALIVC_VOICE_CALL_SP_KEY_USER_INFO = "user_info";

    /**
     * server端的请求域名，需要用户自己替换成自己server端的域名
     */
    private static final String BASE_URL = ;

    /**
     * 获取鉴权信息
     */
    private static final String URL_NEW_TOKEN = BASE_URL + "/interactive-live-class/getRtcAuth";

}
```

> **说明** 此处的AppID、AppKey、服务端URL和Electron端保持一致，更多信息，请参见Electron集成。

3. 运行Demo。
   i. 打开Android Studio，单击**Open an Existing Project**，选择android目录下的ApsaraVideoInteractiveClass文件夹。
   ii. 单击  ，同步工程。



iii. 将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击，编译并运行。




如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。

> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

## Demo目录结构说明

项目结构如下所示：

| 目录名 | 说明 |
| --- | --- |
| App | 程序入口。 |
| IM | AIM相关本地库。 |
| RTCInteractiveClass_RTC | 互动大班课功能实现库。更多信息，请参见RTCInteractiveClass_RTC组件库目录说明。 |
| RTCSolutionCommon | 公共组建库。 |
| RTCViewCommon | 公共UI库。 |
| thirdparty-lib | 第三方库的引用。 |

RTCInteractiveClass_RTC组件库目录说明，如下所示：

| 目录名 | 说明 |
| --- | --- |
| adapter | 列表控件adapter。 |
| bean | 实体类。 |
| constant | 常量数据管理类，在此配置服务端请求域名和分享链接的域名。 |
| im | 阿里云IM SDK接口封装。 |
| network | 网络请求。 |
| rtc | RTC SDK接口封装。 |
| ui | 工具类。 |
| view | 自定义view。 |

## API说明

RTC管理类功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取AliRtcInteractiveEngine的实例对象，初始化RTC SDK。 |
| destory | 销毁AliRtcInteractiveEngine的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。 |
| login | 根据输入的房间号、用户名加入RTC频道。 |
| logout | 退出RTC频道。 |
| enterSeat | 连麦。 |
| leavelSeat | 断开连麦。 |
| muteLocalMic | 是否停止本地音频采集。 |
| muteLocalCamera | 是否停止本地视频采集。 |
| setLocalViewConfig | 为本地预览设置参数及绘制窗口。 |
| startPreview | 开始本地预览。 |
| stopPreview | 停止本地预览。 |
| getLocalSeatUserInfo | 获取本地上麦用户信息。 |
| getRemoteSeatUserInfo | 获取远端用户信息。 |
| setRemoteViewConfig | 为远端的视频设置参数及绘制窗口。 |
| setAliRtcInteractiveListener | 设置监听。 |
| getRoomUserList | 获取房间用户列表。 |
| isActiveStudent | 判断学生是否连麦。 |

RTC回调接口

| API | 描述 |
| --- | --- |
| onEnterSeatResult | 用户上麦的通知。 |
| onLeaveSeatResult | 用户下麦的通知。 |
| onOccurError | 错误信息的通知。 |
| onOccurWarning | 警告信息的通知。 |
| onRoomDestroy | 房间被销毁的回调。 |

| API | 描述 |
| --- | --- |
| onSDKError | SDK发生异常需要销毁实例。 |
| onJoinChannelResult | 加入房间的通知。 |
| onLeaveChannelResult | 离开房间的通知。 |
| onRemoteTrackAvailableNotify | 远端用户音视频流发生变化时的回调。 |
| onActiveSpeaker | 正在发言的用户。 |
| onAudioSubscribeStateChanged | 音频订阅情况变更回调。 |
| onVideoSubscribeStateChanged | 相机流订阅情况变更回调。 |
| onSubscribeStreamTypeChanged | 大小流订阅情况变更回调。 |
| onScreenShareSubscribeStateChanged | 屏幕分享流订阅情况变更回调。 |
| onUserAudioMuted | 用户取消音频的通知。 |
| onUserVideoMuted | 用户取消视频的通知。 |
| onNetworkQualityChanged | 网络质量变化时的回调。 |
| onUpdateRoleNotify | 角色切换成功的通知。 |
| onRemoteUserOnLineNotify | 远端用户上线的通知。 |
| onRemoteUserOffLineNotify | 远端用户下线的通知。 |

IM功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取IM管理类单例。 |
| destroy | 销毁IM管理类。 |
| createUserManager | 创建用户实例并登录。 |
| addMembers | 加入群聊。 |
| sendMessage | 发送消息。 |
| handUp | 举手。 |
| listAllMembers | 拉取所有群成员。 |
| setAliImInteractiveListener | 设置回调。 |

IM回调接口

| API | 描述 |
| --- | --- |
| joinGroupSuccess | 加入群聊成功。 |
| joinGroupFailure | 加入群聊失败。 |
| onLocalLogin | 本地登录成功。 |
| sendMessageSuccess | 发送消息成功。 |
| sendMessageFailure | 发送消息失败。 |
| onAddedMessages | 收到新消息的回调，可通过newMsg.type区分是自己发送的或在线、离线状态收到的消息。 |
| onGroupAddedMembers | 新增群成员。 |
| onGroupRemovedMembers | 移除群成员。 |

RTC管理类功能实现接口

- sharedInstance：获取AliRtcInteractiveEngine的实例对象，初始化RTC SDK。

```
    /**
     * 获取AliRtcInteractiveEngine实例
     */
    public static AliRtcInteractiveEngine sharedInstance() {};
```

- destory：销毁AliRtcInteractiveEngine的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/**
     * 销毁实例
     */
    public abstract void destory();
```

- login：根据输入的房间号、用户名加入RTC频道。

```
    /**
     * 登录
     *
     * @param channelId 房间号
     * @param userName 昵称
     */
    public abstract void login(String channelId,  String userName);
```

- logout：退出RTC频道。

```
    /**
     * 登出
     */
    public abstract void logout();
```

- enterSeat：连麦。

```
/**
     * 上麦
     */
    public abstract void enterSeat();
```

- leavelSeat：断开连麦。

```
/**
     * 下麦
     */
    public abstract void leavelSeat();
```

- muteLocalMic：是否停止本地音频采集。

```
    /**
     * 停止发布音频
     */
    public abstract int muteLocalMic(boolean isMute);
```

- muteLocalCamera：是否停止本地视频采集。

```
  /**
     * 停止发布视频
     */
    public abstract int muteLocalCamera(boolean isMute);
```

- setLocalViewConfig：为本地预览设置参数及绘制窗口。

```
  /**
     * 设置本地预览渲染参数
     */
    public abstract void setLocalViewConfig(AliRtcEngine.AliVideoCanvas localAliVideoCanv
as, AliRtcEngine.AliRtcVideoTrack aliRtcVideoTrackCamera);
```

- startPreview：开始本地预览。

```
/**
     * 开启预览
     */
    public abstract void startPreview();
```

- stopPreview：停止本地预览。

```
   /**
     * 停止预览
     */
    public abstract void stopPreview();
```

- getLocalSeatUserInfo：获取本地上麦用户信息。

```
    /**
     * 获取本地上麦用户信息
     */
    public abstract RemoteUserInfo getLocalSeatUserInfo();
```

- getRemoteSeatUserInfo：获取远端用户信息。

```
    /**
     * 获取远端用户信息
     */
    public abstract AliRtcRemoteUserInfo getRemoteSeatUserInfo(String uerId);
```

- setRemoteViewConfig：为远端的视频设置参数及绘制窗口。

```
 /**
     * 设置远端渲染参数
     */
    public abstract void setRemoteViewConfig(AliRtcEngine.AliVideoCanvas aliVideoCanvas,
String uid, AliRtcEngine.AliRtcVideoTrack aliRtcVideoTrack);
```

- setAliRtcInteractiveListener：设置监听。

```
  /**
     * 设置监听
     */
    public abstract void setAliRtcInteractiveListener(AliRtcInteractiveListener callback)
;
```

- getRoomUserList：获取房间用户列表。

```
    /**
     * 获取房间用户列表
     */
    public abstract String[] getRoomUserList();
```

- isActiveStudent：

```
    /**
     * 判断学生是否连麦
     */
    public abstract boolean isActiveStudent();
```

RTC回调接口

- onEnterSeatResult：用户上麦的通知。

```
   /**
    * 自己上麦结果通知
    * @param result 0为成功，反之失败
    */
   void onEnterSeatResult(int result);
```

- onLeaveSeatResult：用户下麦的通知。

```
   /**
    * 自己下麦结果通知
    */
   void onLeaveSeatResult(int result);
```

- onOccurError：错误信息的通知，当错误码是以下几种情况时，需要销毁SDK实例。
  - ErrorCodeEnum.ERR_ICE_CONNECTION_HEARTBEAT_TIMEOUT
  - ErrorCodeEnum.ERR_SDK_INVALID_STATE

- ErrorCodeEnum.ERR_SESSION_REMOVED

```
    /**
     * SDK报错
     *
     */
    void onOccurError(int error);
```

- onOccurWarning：警告信息的通知。

```
  /**
     * SDK警告
     *
     */
    void onOccurWarning( int error);
```

- onRoomDestroy：房间被销毁的回调。

```
   /**
     * 房间被销毁的回调
     */
    void onRoomDestroy(int i);
```

- onSDKError：SDK发生异常需要销毁实例。

```
    /**
     * SDK报错，需要销毁实例
     */
    void onSDKError( int error);
```

- onJoinChannelResult：加入房间的通知。

```
     /**
      * 加入房间通知
      * @param result 0为成功，反之失败
      */
     void onJoinChannelResult(int result);
```

- onLeaveChannelResult：离开房间的通知。

```
     /**
      * 离开房间通知
      */
     void onLeaveChannelResult(int result);
```

- onRemoteTrackAvailableNotify：远端用户音视频流发生变化时的回调。

```
    /**
     *
     * 当订阅情况发生变化时，返回这个消息onSubscribeChangedNotify
     * @param userId  用户ID
     * @param videoTrack      订阅成功的视频流
     * @param audioTrack      订阅成功的音频流
     */
    void onRemoteTrackAvailableNotify(String userId, AliRtcEngine.AliRtcAudioTrack audioTr
ack, AliRtcEngine.AliRtcVideoTrack videoTrack);
```

- onActiveSpeaker：正在发言的用户。

```
  /**
    *正在发言的学生
    */
    void onActiveSpeaker(String speakers);
```

- onAudioSubscribeStateChanged：音频订阅情况变更回调。

```
    /**
     * 音频订阅情况变更回调
     * @param uid 用户Id
     * @param oldState 之前的订阅状态
     * @param newState 当前的订阅状态
     */
    void onAudioSubscribeStateChanged(String uid, AliRtcEngine.AliRtcSubscribeState oldSt
ate, AliRtcEngine.AliRtcSubscribeState newState, int elapseSinceLastState, String channel
);
```

- onVideoSubscribeStateChanged：相机流订阅情况变更回调。

```
    /**
     * 相机流订阅情况变更回调
     * @param uid 用户Id
     * @param oldState 之前的订阅状态
     * @param newState 当前的订阅状态
     */
    void onVideoSubscribeStateChanged(String uid, AliRtcEngine.AliRtcSubscribeState oldSt
ate, AliRtcEngine.AliRtcSubscribeState newState, int elapseSinceLastState, String channel
);
```

- onSubscribeStreamTypeChanged：大小流订阅情况变更回调。

```
    /**
     * 大小流订阅情况变更回调
     * @param uid 用户Id
     * @param oldStreamType 之前的订阅状态
     * @param newStreamType 当前的订阅状态
     */
    void onSubscribeStreamTypeChanged(String uid, AliRtcEngine.AliRtcVideoStreamType oldS
treamType, AliRtcEngine.AliRtcVideoStreamType newStreamType, int elapseSinceLastState, St
ring channel);
```

- onScreenShareSubscribeStateChanged：屏幕分享流订阅情况变更回调。

```
    /**
     * 屏幕分享流订阅情况变更回调
     * @param uid 用户Id
     * @param oldState 之前的订阅状态
     * @param newState 当前的订阅状态
     */
    void onScreenShareSubscribeStateChanged(String uid, AliRtcEngine.AliRtcSubscribeState
oldState, AliRtcEngine.AliRtcSubscribeState newState, int elapseSinceLastState, String ch
annel);
```

- onUserAudioMuted：用户取消音频的通知

```
    /**
     * 用户取消音频的通知
     */
    void onUserAudioMuted(String userId, boolean mute);
```

- onUserVideoMuted：用户取消视频通知。

```
    /**
     * 用户取消视频通知
     */
    void onUserVideoMuted(String userId, boolean mute);
```

- onNetworkQualityChanged：网络质量变化时的回调。

```
/**
     * 网络状态回调
     *
     * @param aliRtcNetworkQuality1 下行网络质量
     * @param aliRtcNetworkQuality  上行网络质量
     * @param s                     用户ID
     */
    void onNetworkQualityChanged(String s, AliRtcEngine.AliRtcNetworkQuality aliRtcNetwor
kQuality, AliRtcEngine.AliRtcNetworkQuality aliRtcNetworkQuality1);
```

- onUpdateRoleNotify：角色切换成功的通知。

```
    /**
     *
     * 角色切换成功通知
     */
    void onUpdateRoleNotify(AliRtcEngine.AliRTCSDK_Client_Role oldRole, AliRtcEngine.AliRT
CSDK_Client_Role newRole);
```

- onRemoteUserOnLineNotify：远端用户上线的通知。

```
    /**
     * 用户上线通知
     *
     * @param userId 用户ID
     */
    void onRemoteUserOnLineNotify(String userId);
```

- onRemoteUserOffLineNotify：远端用户下线的通知。

```
    /**
     * 用户下线通知
     * @param userId 用户ID
     */
    void onRemoteUserOffLineNotify(String userId);
```

IM功能实现接口

- sharedInstance：获取IM管理类单例。

```
/**
 * 获取单例
 */
public static AliImInteractiveEngine sharedInstance() {
    return AliImInteractiveEngineImpl.sharedInstance();
}
```

- destroy：销毁IM管理类。

```
/**
 * 销毁实例
 */
public static void destroy() {
    AliImInteractiveEngineImpl.destroyInstance();
}
```

- createUserManager：创建用户实例并登录。

```
/**
 * 创建用户实例并登录
 *
 * @param userId 用户ID
 */
public abstract void createUserManager(String userId);
```

- addMembers：加入群聊。

```
/**
 * 加入群聊
 *
 * @param userId   用户ID
 * @param nickName 昵称
 */
public abstract void addMembers(String userId, String channelId, String nickName);
```

- sendMessage：发送消息。

```
/**
 * 发送消息
 *
 * @param userId 用户ID
 * @param msg    消息内容
 */
public abstract void sendMessage(String userId, String msg, String displayName);
```

- handUp：举手。

```
/**
 * 举手
 *
 * @param userId 用户ID
 */
public abstract void handUp(String userId, String displayName);
```

- listAllMembers：拉取所有群成员。

```
/**
 * 拉取所有群成员
 */
public abstract void listAllMembers(AIMPubGroupListAllMemberListener listener);
```

- setAliImInteractiveListener：设置回调。

```
/**
 * 设置回调
 */
public abstract void setAliImInteractiveListener(AliImInteractiveListener callback);
```

IM回调接口

- joinGroupSuccess：加入群聊成功。

```
/**
 * 加入群聊成功
 */
void joinGroupSuccess(ArrayList<AIMPubGroupMember> arrayList);
```

- joinGroupFailure：加入群聊失败。

```
/**
 * 加入群聊失败
 */
void joinGroupFailure(DPSError dpsError);
```

- onLocalLogin：本地登录成功。

```
/**
 * 本地登录成功
 */
void onLocalLogin();
```

- sendMessageSuccess：发送消息成功。

```
/**
 * 发送消息成功
 */
void sendMessageSuccess(AIMPubMessage aimPubMessage);
```

- sendMessageFailure：发送消息失败。

```
/**
 * 发送消息失败
 */
void sendMessageFailure(DPSError dpsError);
```

- onAddedMessages：收到新消息的回调，可通过newMsg.type区分是自己发送的或在线、离线状态收到的消息。

```
/**
 * 收到新消息的回调，可通过newMsg.type区分是自己发送的或在线、离线状态收到的消息
 */
void onAddedMessages(final ArrayList<AIMPubNewMessage> arrayList);
```

- onGroupAddedMembers：新增群成员。

```
/**
 * 新增群成员
 */
void onGroupAddedMembers();
```

- onGroupRemovedMembers：移除群成员。

```
/**
 * 移除群成员
 */
void onGroupRemovedMembers();
```

# 2.4. 语音聊天室

## 2.4.1. 简介

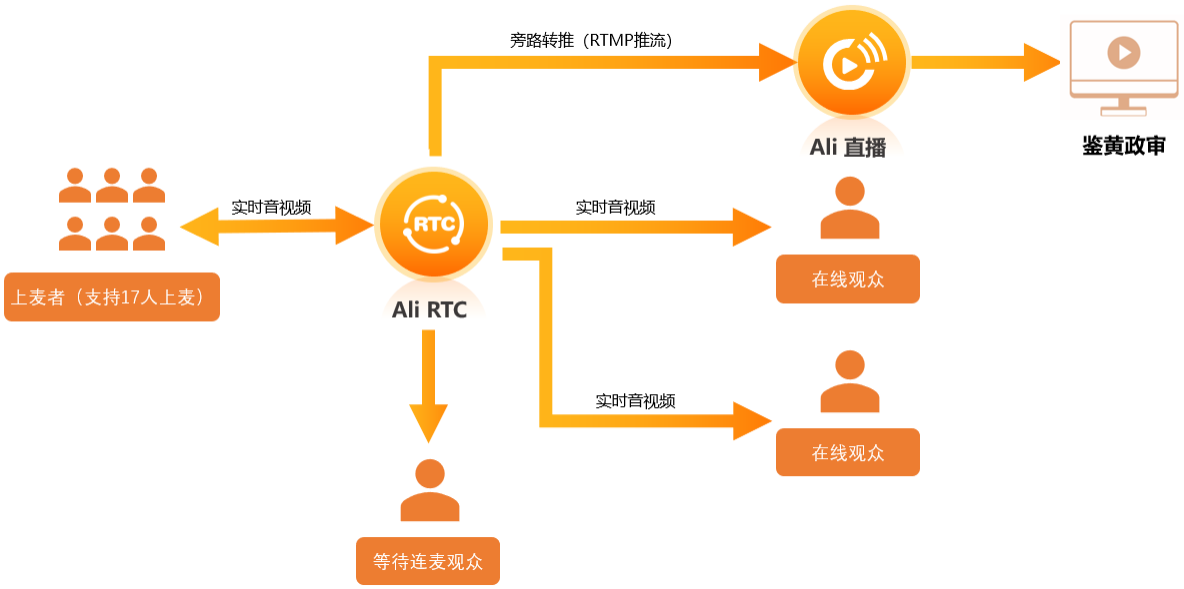
通过阅读本文，您可以快速了解语音聊天室的基本信息及实现方法。

### 使用场景

语音聊天室一般由主播和在线观众组成。房间内在线观众可以听到主播的声音，在线观众也可以通过上麦功能参与语音互动。同时在语音互动过程中互动者具备丰富的功能玩法，例如播放背景音乐、播放鼓励音效、设置混响变声等音频效果。

### 架构方案

语音聊天室架构方案如下图所示：

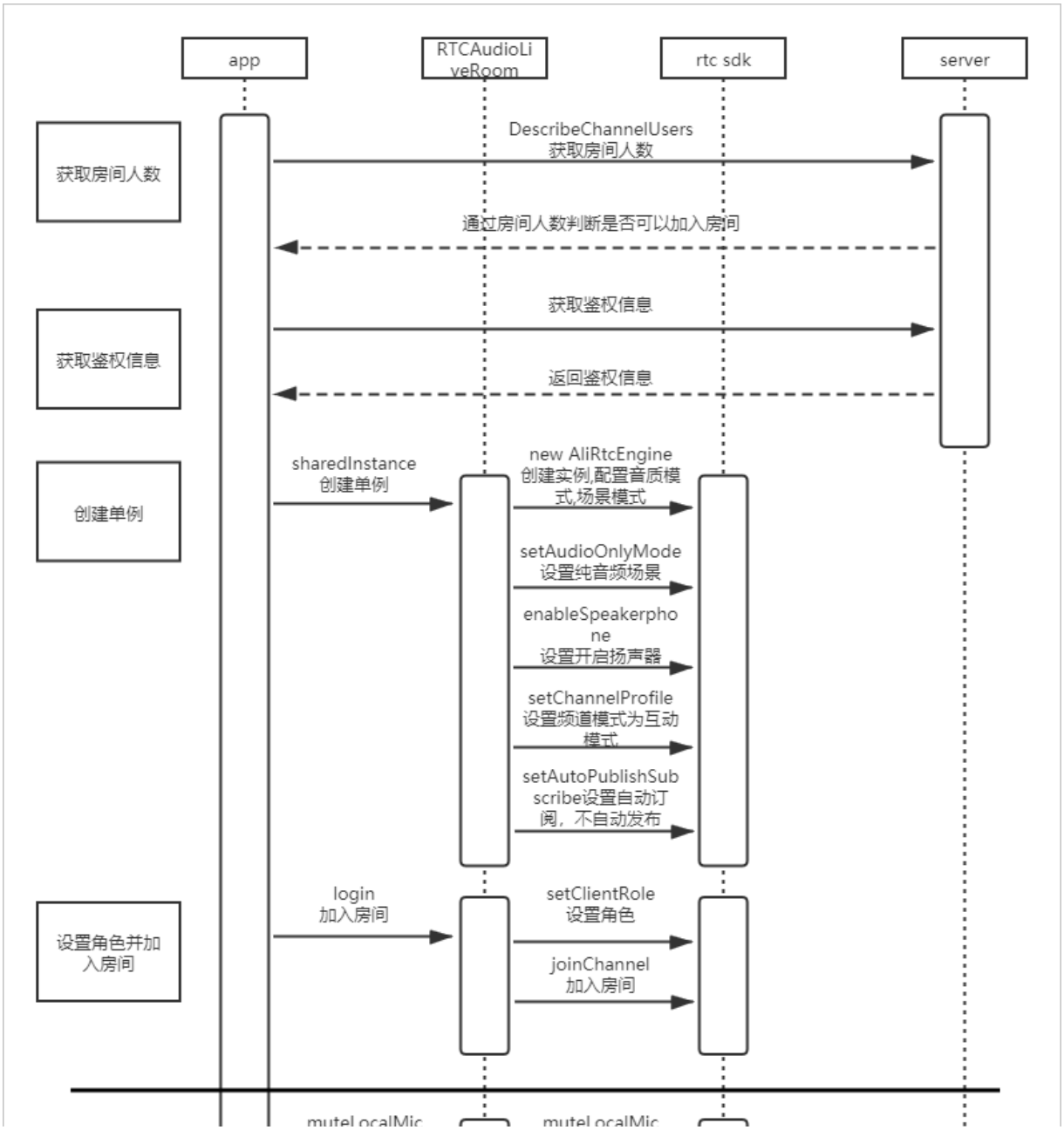


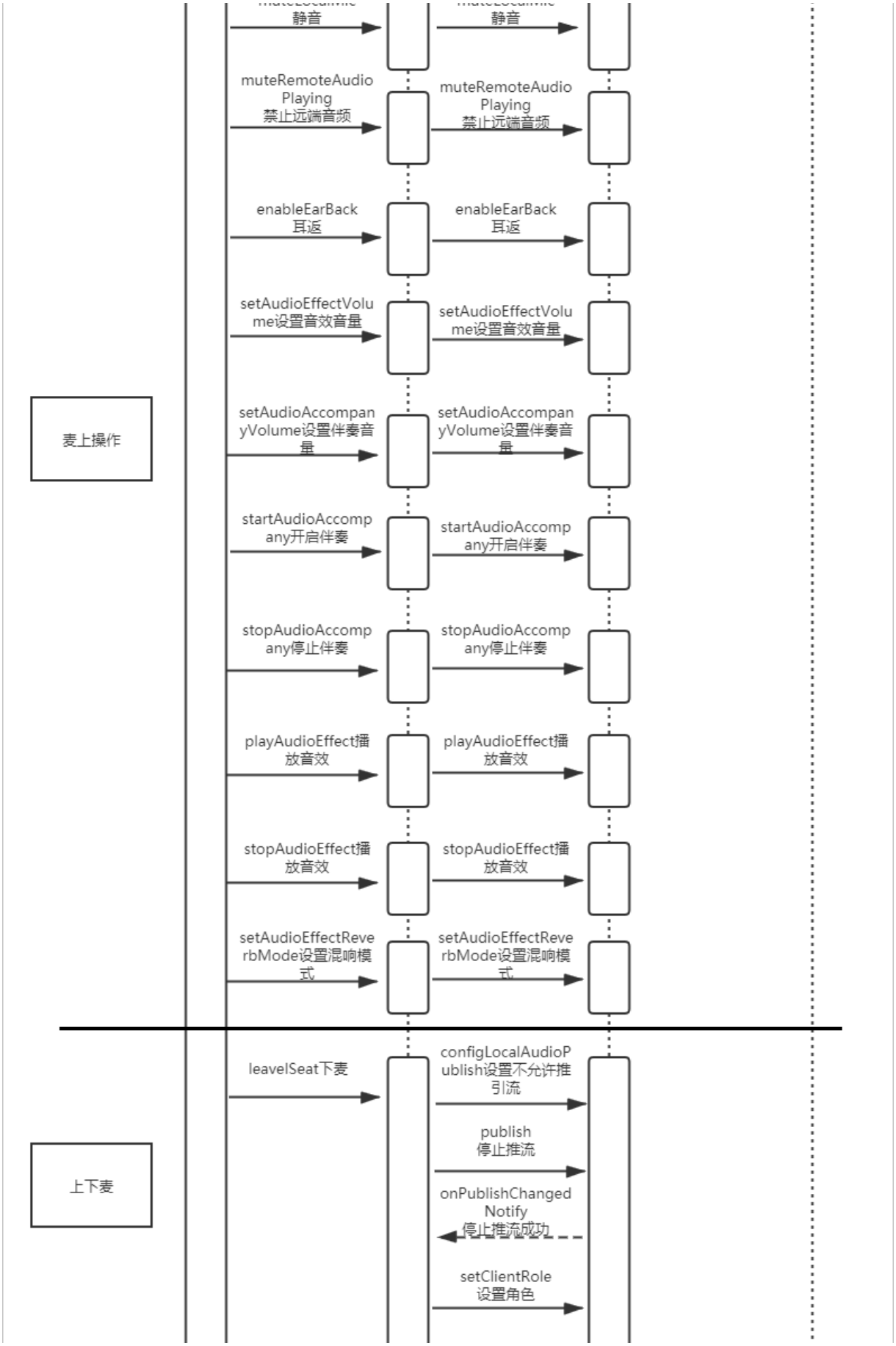

### 主要功能

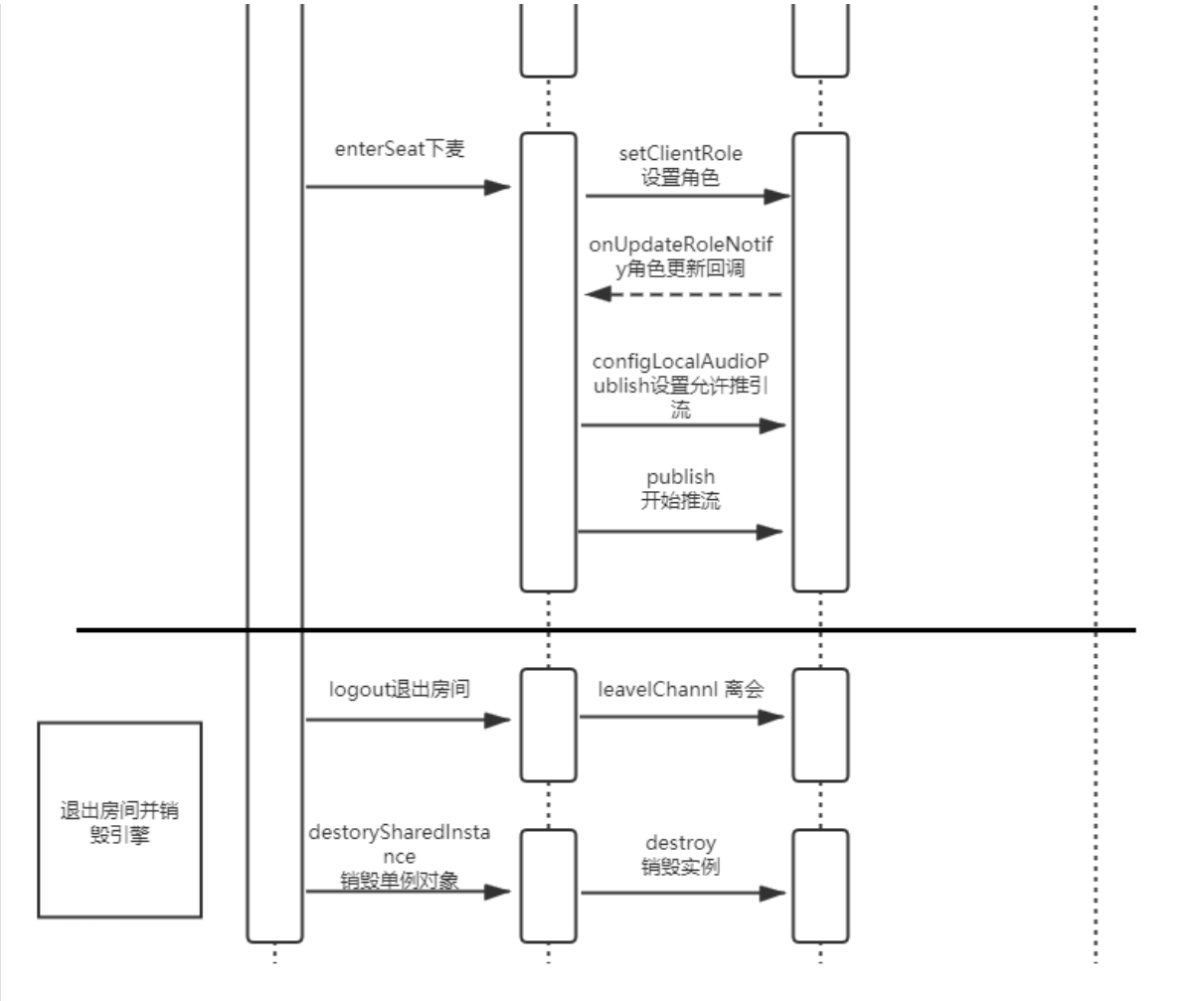
| 功能 | 描述 |
| --- | --- |
| 多人实时语音通话 | Demo支持8人同时进行语音通话，在线观众可以实时收听麦上用户的通话内容，也可以上麦参与语音互动。 |
| 伴奏音效 | 互动角色可以播放伴奏音乐，也可以播放预设好的鼓励音效。 |
| 耳返 | 互动角色可以开启耳返实时监听自己的声音效果。 |
| 变声混响 | Demo提供多种混响变声效果，提供丰富的音频互动玩法。 |

## 实现方法

语音聊天室是基于RTC和场景业务结合的开源组件，通过封装音视频通信RTC SDK接口实现场景业务功能，具体时序图如下所示：

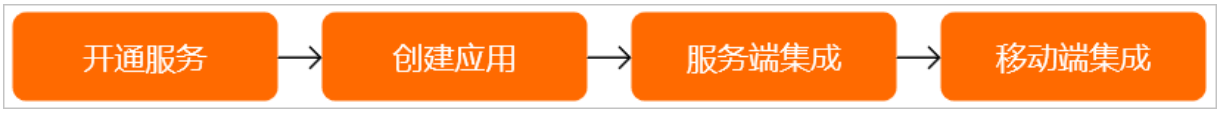



静音
静音
muteRemoteAudio Playing 禁止远端音频
muteRemoteAudio Playing 禁止远端音频
enableEarBack 耳返
enableEarBack 耳返
setAudioEffectVolume设置音效音量
setAudioEffectVolume设置音效音量
麦上操作
setAudioAccompanyVolume设置伴奏音量
setAudioAccompanyVolume设置伴奏音量
startAudioAccompany开启伴奏
startAudioAccompany开启伴奏
stopAudioAccompany停止伴奏
stopAudioAccompany停止伴奏
playAudioEffect播放音效
playAudioEffect播放音效
stopAudioEffect播放音效
stopAudioEffect播放音效
setAudioEffectReverbMode设置混响模式
setAudioEffectReverbMode设置混响模式
leaveISeat下麦
configLocalAudioPublish设置不允许推引流
publish 停止推流
onPublishChanged Notify 停止推流成功
setClientRole 设置角色
上下麦




## 实现流程

实现流程如下图所示：




| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务 | 进行服务端集成之前，您必须开通音视频通信服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |

| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 3 | 服务端集成 | 您可以通过集成服务端文档在本地快速运行语音聊天室的服务端，也可以参考服务端源码在自己的服务端进行开发。<br>**注意** 在集成服务端源码时需要初始化数据库。 |
| 4 | 移动端集成：<br>• iOS集成<br>• Android集成 | 您可以通过源码快速搭建移动端语音聊天室。 |

### Demo体验

您可以通过钉钉扫描以下二维码，下载安装语音聊天室Demo体验。

# 2.4.2. 服务端集成

通过阅读本文，您可以了解语音聊天室服务端的集成操作。

### 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 环境中已安装Java JDK 8的版本。具体操作，请参见安装JDK。

  **说明** 如果服务端为Linux环境，推荐安装Oracle JDK，不推荐使用Open JDK进行服务端集成。

- 服务端环境已安装MySQL（建议安装MySQL 5.7版本）。具体操作，请参见安装MySQL。

### 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。

ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。

iii. 在**AppID/名称**列获取AppID。

iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 获取AccessKey。

> **说明** 由于主账号AccessKey泄露会威胁您所有资源的安全，因此出于安全考虑，需要创建RAM用户（子账号）并获取RAM用户（子账号）的AccessKey，用于访问您的云资源。

i. 登录RAM控制台。

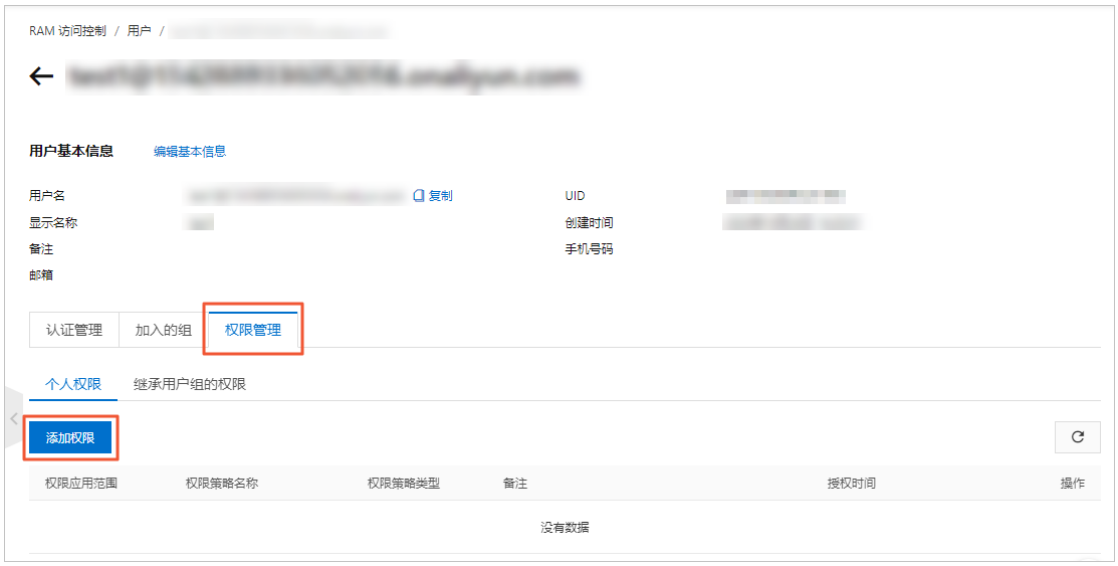
ii. 在左侧导航栏选择**身份管理 > 用户**，进入用户页面。

iii. 单击**创建用户**，填写用户账号信息，选中**Open API 调用访问**创建用户。

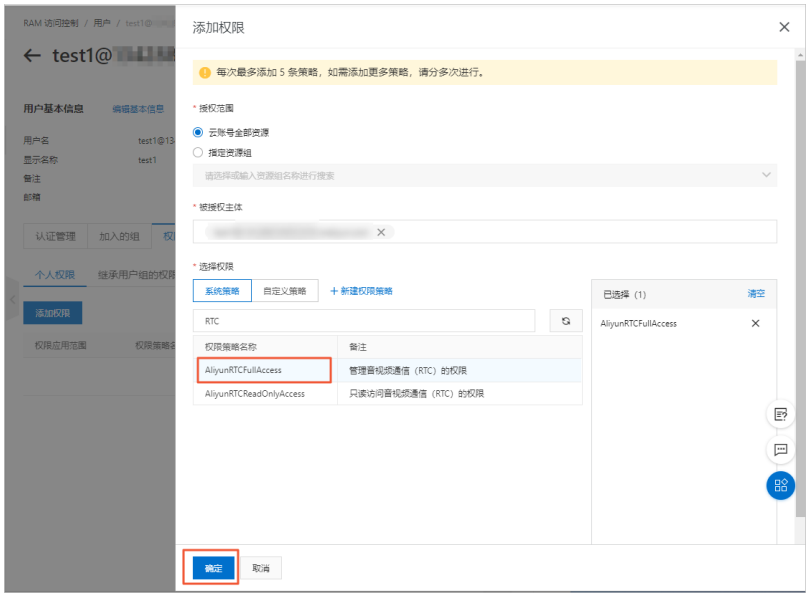

iv. 单击**确定**。

v. 重新返回用户页面。在**用户登录名称/显示名称**列下，单击目标RAM子账户，进入管理页面。

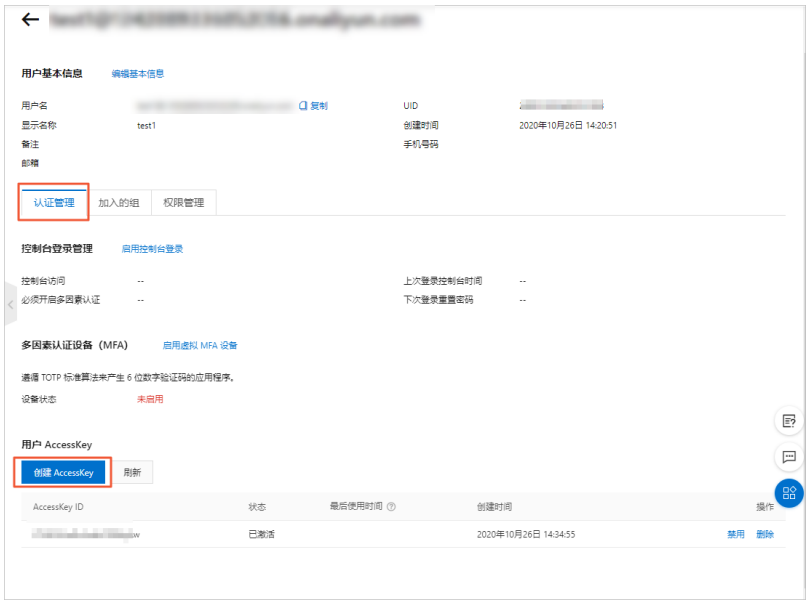
vi. 单击**权限管理**，再单击**添加权限**。




vii. 选择AliyunRTCFullAccess（管理音视频通信的权限，可输入RTC进行搜索）。单击**确定**。



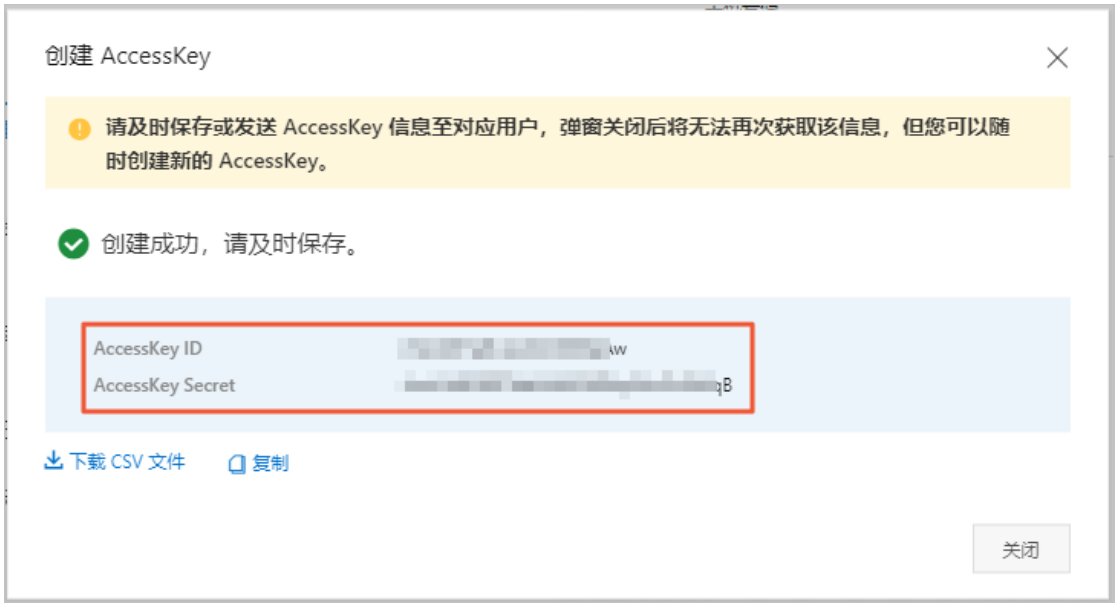
viii. 单击**认证管理**，在用户AccessKey区域单击**创建AccessKey**。




> **说明**
>
> - 首次创建时需填写手机验证码。
> - 如果AccessKey泄露或丢失，则需要创建新的AccessKey，最多可以创建2个AccessKey。

ix. 获取AccessKey ID和AccessKey Secret。

创建AccessKey完成后，会弹出**创建AccessKey**对话框，包含AccessKey ID和AccessKey Secret信息。此处建议记录一下AccessKey ID和AccessKey Secret，方便后续操作中使用。

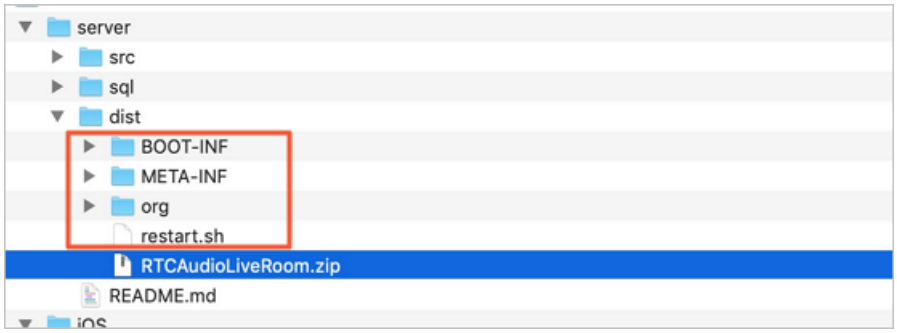



3. 下载并解压Demo，更多信息，请参见Demo源码下载。

> 说明
> - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

4. 将*server/dist/RTCAudioLiveRoom.zip*文件解压到*dist*文件夹下。

   解压成功之后如下图所示：




> 说明 解压成功后，可以看到BOOT-INF、META-INF及org文件夹，这三个文件夹需要和restart.sh保持在同一个目录下。

5. 初始化数据库。

   在MySQL中创建数据库并使用sql文件建表。

   sql文件路径：*server/sql/appserver_create_table.sql*。

   i. 启动MySQL服务。
      - Windows端：**net start MySQL服务名**
      - Mac端：**sudo Mysql服务路径 start**
      - Linux端：**service mysql start**

   ii. 新建终端，并将终端定位到sql文件目录下。

   iii. 创建语音聊天室数据库。

   ```
   mysqladmin -u root -p create chatroom
   ```

   iv. 在语音聊天室数据库中执行sql文件。

   ```
   mysql -u 用户名 -p 密码 chatroom < ./appserver_create_table.sql
   ```

6. 修改配置文件。

   打开*BOOT-INF/classes/application.properties*文件，修改配置。

> 说明 使用文本编辑器打开即可，若找不到打开方式，推荐安装VSCode等轻量级代码编辑器打开。

```
spring.datasource.url = jdbc:mysql://*****:3306/chatroom?useSSL=false&useUnicode=true&c
spring.datasource.username = *****
spring.datasource.password = *****


spring.http.multipart.maxFileSize=100Mb
spring.http.multipart.maxRequestSize=100Mb

# MyBatis
mybatis.mapper-locations=classpath:mapper/*/*.xml
mybatis.type-aliases-package=com.alivc.vod.pojo.*


server.port=*****
server.tomcat.accesslog.buffered=false
server.tomcat.accesslog.directory=/home/alivcrtc/chatroomJar
server.tomcat.accesslog.enabled=true

server.context-path=/chatroom/

#################################################log##################################
logging.level.com.live.dao=debug
logging.level.com.live.dao.user=debug
logging.level.com.alivc.vod.dao=debug
logging.file:my.log
logging.pattern.file=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{15}.%M : %
logging.pattern.console=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{15}.%M

accessKeyID=*****
accessKeySecret=*****
VOD_REGIONID = cn-shanghai


rtc.chatroom.appId = *****
rtc.chatroom.appKey = *****
rtc.gslb = https://rgslb.rtc.aliyuncs.com
```

- 设置RTC应用AppId和AppKey，可参见步骤1。

  rtc.chatroom.appId = *

  rtc.chatroom.appKey = *

- 设置AK，需要添加AliyunRTCFullAccess权限，可参见步骤2。

  accessKeyID=*

  accessKeySecret=*

- 修改数据库访问地址、用户名和密码。

  数据库访问地址：spring.datasource.url = jdbc:mysql://数据库IP地址（本地：127.0.0.1）:3306/数据库名（chatroom）?useSSL=false&useUnicode=true&characterEncoding=UTF-8

  用户名：spring.datasource.username = *

  密码：spring.datasource.password = *

  > **说明** 如果数据库和服务端都在本地运行，则数据库IP地址可以使用127.0.0.1。

7. 运行服务，执行*restart.sh*文件。如果编译过程中出现问题，请参见服务端运行常见问题。

   - Mac或Linux环境下请将终端定位至*dist*目录下，执行如下命令：

     **sh restart.sh**

     后台执行可以使用如下命令以确保退出终端时程序能够继续运行。

     **nohup sh restart.sh >./run_log.log 2>&1 &**

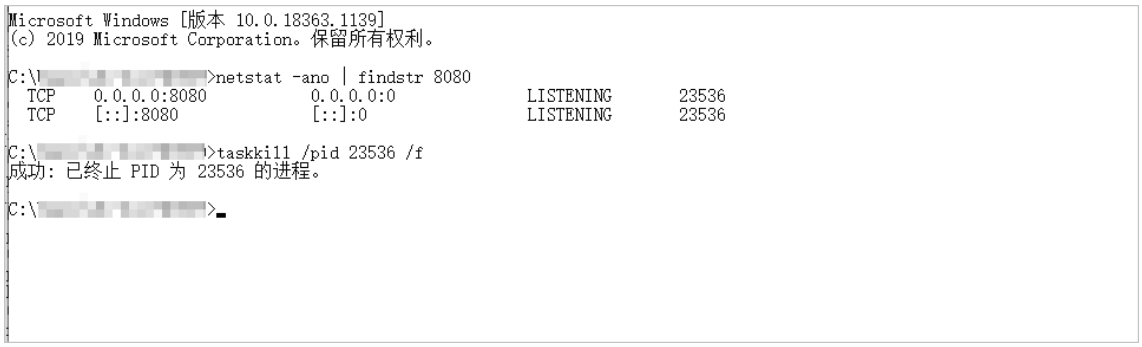
- Windows环境需要打开CMD终端定位到*server/dist*文件夹下，执行如下命令：

  **java org.springframework.boot.loader.JarLauncher**

  若提示8080端口被占用，请尝试使用netstat命令查看占用8080端口的进程pid号，并使用taskkill关闭相关进程。

  **netstat -ano | findstr 8080**

  **taskkill /pid 占用端口的进程pid号 /f**




终端成功运行后可以看到服务端启动成功的日志信息。


8. 检查服务端是否已经启动。

   在浏览器中访问*http://127.0.0.1:8080/chatroom*，如果显示如下图所示，表示服务端已经启动。


## 主要功能说明

- 生成随机用户，并返回用户信息。

  访问地址：*/user/randomUser*

  随机生成用户信息和客户端调用RTC SDK加入房间的Token信息。具体生成方式参考RTC帮助文档生成Token。

```
 if (StringUtils.isBlank(channelId)) {
        channelId = RandomStringUtils.randomNumeric(5);
 }
JSONObject rtcAuth = RtcOpenAPI.createToken(channelId, UUID.randomUUID().toString());
rtcAuth.put("userName", RandomString.getRandomName());
return rtcAuth;
```

- 成功加入房间后保存用户信息。

  访问地址：*/user/joinSuccess*

  客户端成功加入房间后，通知AppServer，保存用户信息到rds。

```
userService.insertUser(channelId, userid, userName, seatIndex);
```

- 获取上麦用户麦序。

  访问地址：*/user/getSeatList*

  获取所有上麦用户麦序信息。查询频道实时在线用户列表，对比AppServer保存的用户麦序，为新上麦的用户分配麦序，删除下麦用户的麦序。

```
        List<User> userList = userService.getUserList(channelId);
        DescribeChannelUsersResponse describeChannelUsersResponse = RtcOpenAPI.describeCh
annelUsers(appId, channelId);
        List<String> liveUserList = describeChannelUsersResponse.getInteractiveUserList()
;
        List<String> takeSeatUserIds = new ArrayList<>(liveUserList);
        List<String> leaveSeatUserIds = new ArrayList<>(userIdList);
        List<String> existSeatIndex = new ArrayList<>();
        for (User user : userList) {
            if (liveUserList.contains(user.getUserId())) {
                existSeatIndex.add(user.getSeatIndex());
            }
        }
        leaveSeatUserIds.removeAll(liveUserList);
        takeSeatUserIds.removeAll(userIdList);
        List<User> updateUserList = new ArrayList<>();
        for (int i = 0, takeSeatIndex = 0; i < 8 && takeSeatIndex < takeSeatUserIds.size(
); i++) {
            if (!existSeatIndex.contains(String.valueOf(i))) {
                User user = new User();
                user.setUserId(takeSeatUserIds.get(takeSeatIndex));
                user.setSeatIndex(String.valueOf(i));
                updateUserList.add(user);
                takeSeatIndex++;
            }
        }
        for (String leaveSeatUserId : leaveSeatUserIds) {
            User user = new User();
            user.setUserId(leaveSeatUserId);
            user.setSeatIndex(null);
            updateUserList.add(user);
        }
        userService.updateUserSeats(updateUserList);
        userList = userService.getUserList(channelId);
```

- 查询房间人数。

  调用访问地址：*/user/describeChannelUsers*

  查询频道实时在线用户列表。

```
       DefaultAcsClient client = initVodClient();
        DescribeChannelUsersRequest request = new DescribeChannelUsersRequest();
        request.setAppId(appId);
        request.setChannelId(channelId);
        DescribeChannelUsersResponse response = client.getAcsResponse(request);
```

# 2.4.3. Android集成

通过阅读本文，您可以了解语音聊天室Android端的集成操作。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

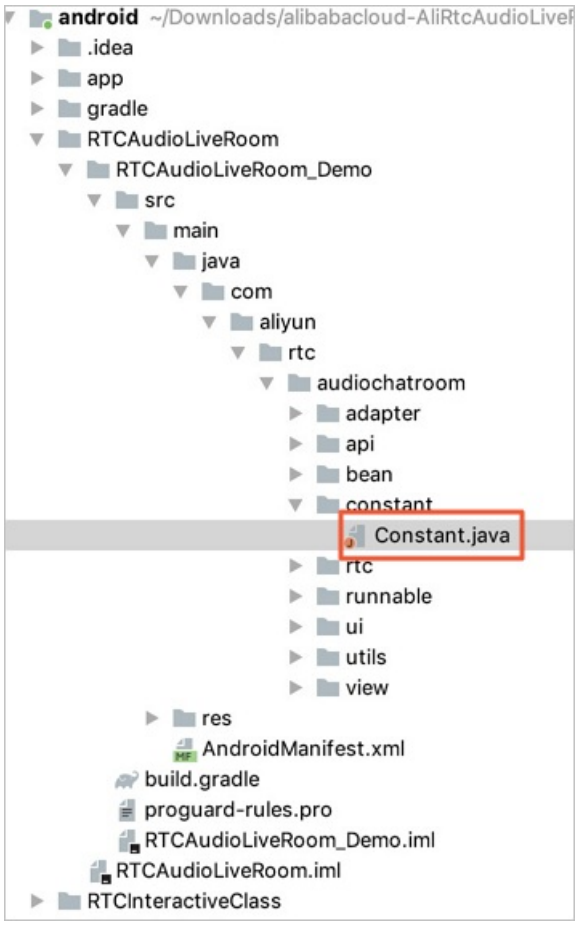
- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Android Studio 3.0或以上版本。更多信息，请参见Android Studio。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > 说明
   > - Demo源码中已通过Maven方式集成AliRTC Android SDK（1.18版本）。
   > - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 配置Demo工程。
   i. 打开Android Studio，单击**Open an Existing Project**，选择*android*文件夹。
   ii. 打开*android/RTCAudioLiveRoom/RTCAudioLiveRoom_Demo/src/main/java/com/aliyun/rtc/audiochatroom/constant/Constant.java*文件。



iii. 修改文件中的 `BASE_URL` 值。

例如本地服务端IP地址为192.0.2.1，则 `BASE_URL` 的值为 `http://192.0.2.1:8080/chatroom` ，即 `BASE_URL="http://192.0.2.1:8080/chatroom"` 。本地服务端IP地址查询，请参见查询IP地址。

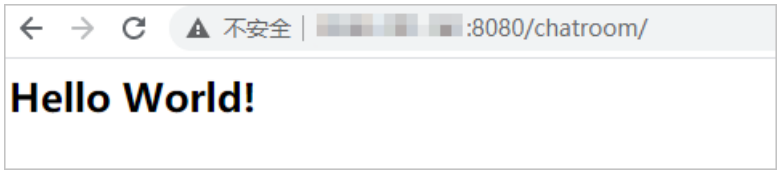
Constant.java

```
1  package com.aliyun.rtc.audiochatroom.constant;
2
3  public class Constant {
4
5      public static final String NEW_TOKEN_PARAMS_KEY_USERID = "userid";
6      public static final String PATH_ASSETS_BGM = "mp3/bgm.zip";
7      public static final String PATH_ASSETS_AUDIOEFFECT = "mp3/audioeffect.zip";
8      public static final String PATH_DIR_BGM_OUT = "bgm";
9      public static final String PATH_DIR_AUDIOEFFECT_OUT = "audioeffect";
10     //背景乐、音效默认音量
11     public static final int VALUE_AUDIO_EFFECT_VOLUME = 100;
12     public static final String NEW_TOKEN_PARAMS_KEY_USERNAME = "userName";
13     /**
14      * server端的请求域名，需要用户自己替换成自己server端的域名
15      */
16     private static final String BASE_URL = "http:/[redacted]:8080/chatroom";
17     /**
18      * 获取鉴权信息
19      */
```

> **说明**
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `BASE_URL="http://<域名>/chatroom"` 。

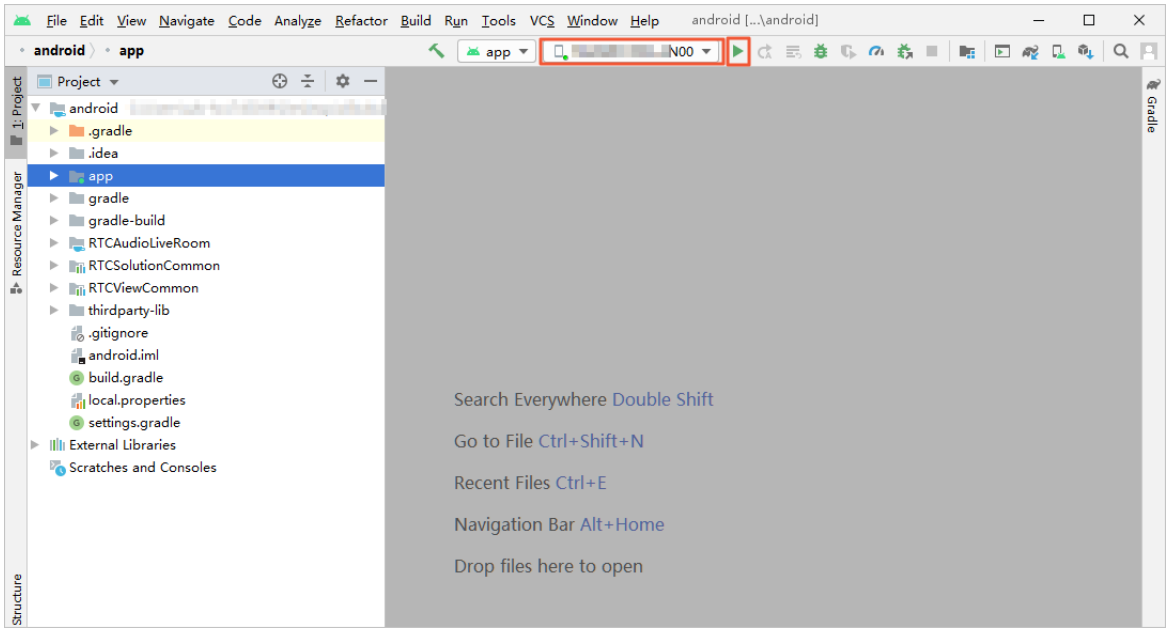
iv. 验证移动端和服务端。

分别在移动端和服务端浏览器中访问 `BASE_URL` 地址，如果显示如下图所示，表示移动端访问服务端正常。




3. 运行Demo。

将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击 ▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。




> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

4. 加入语音聊天室。
   i. 将2台或2台以上移动端设备安装Demo App。
   ii. 将设备连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间号和昵称，选择角色进入语聊房并等待他人加入。
   iv. 在其余设备上输入相同房间号和任意昵称（可同名，不做限制），选择角色加入房间并进行聊天。

## Demo目录结构说明

项目结构如下所示：

| 文件名 | 说明 |
| --- | --- |
| RTCAudioLiveRoom | 语音聊天室功能实现库。更多信息，请参见RTCAudioLiveRoom组件库目录说明。 |
| RTCSolutionCommon | 公共组建库。 |
| RTCViewCommon | 公共UI库。 |
| app | 程序入口。 |
| thirdparty-lib | 第三方库的引用。 |

RTCAudioLiveRoom组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| adapter | 列表控件adapter。 |

| 文件名 | 说明 |
| --- | --- |
| bean | 实体类。 |
| constant | 常量数据管理类，在此配置服务端请求域名和分享链接的域名。 |
| api | 网络请求。 |
| rtc | RTC。 |
| ui | 互动界面和登录界面。 |
| util | 工具类。 |
| view | 自定义view。 |

## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destorySharedInstance | 毁单例对象。 |
| login | 加入房间。 |
| logout | 退出房间。 |
| setAudioEffectVolume | 设置音效音量。 |
| setAudioAccompanyVolume | 设置伴奏音量。 |
| enterSeat | 上麦。 |
| leavelSeat | 下麦。 |
| setRTCAudioLiveRoomDelegate | 设置监听回调。 |
| muteLocalMic | 设置是否静音。 |
| muteRomteAudioPlaying | 设置是否本地静音。 |
| enableEarBack | 是否开启耳返。 |
| startAudioAccompany | 播放伴奏。 |
| stopAudioAccompany | 停止伴奏。 |
| playAudioEffect | 播放音效。 |
| stopAudioEffect | 停止音效。 |

| API | 描述 |
| --- | --- |
| setAudioEffectReverbMode | 设置音效场景（混音模式）。 |
| setAudioEffectVoiceChangerMode | 设置变声音效模式。 |

回调接口

| API | 描述 |
| --- | --- |
| onEnterSeat | 用户上麦通知。 |
| onLeaveSeat | 用户下麦通知。 |
| onOccurError | 错误信息通知。 |
| onUpdateRoleNotify | 切换用户角色回调。 |
| onPublishChangedNotify | 推流回调。 |
| onJoinChannelResult | 加入频道结果回调。 |
| onLeaveChannelResult | 离开频道结果回调。 |
| onSeatVolumeChanged | 音量大小提示。 |
| onAudioPlayingStateChanged | 伴奏播放回调。 |
| onUserAudioMuted | 用户muteAudio通知回调。 |
| onRoomDestroy | 房间被销毁回调。 |
| onNetworkQualityChanged | 网络状态回调。 |

功能实现接口

- sharedInstance：获取单例对象。

  获取BaseRTCAudioLiveRoom的实例对象，初始化RTC SDK。

```
/**
 * 获取单例
 */
public static BaseRTCAudioLiveRoom sharedInstance() {
    return RTCAudioLiveRoomImpl.sharedInstance();
}
```

- destorySharedInstance：毁单例对象。

  销毁BaseRTCAudioLiveRoom的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/**
 * 销毁实例
 */
public abstract void destorySharedInstance();
```

- login：加入房间。

  根据选中的角色类型、房间号、用户名加入RTC频道。

```
/**
     * 加入房间
     *
     * @param role            角色类型
     * @param channelId       房间号
     * @param userName        用户名
     */
    public abstract void login(String channelId, String userName, AliRtcEngine.AliRTCSDK_
Client_Role role);
```

- logout：退出房间。

  退出RTC频道。

```
    /**
     * 退出房间
     */
    public abstract void logout();
```

- setAudioEffectVolume：设置音效音量。

  同时设置播放和推流的音量。

```
    /**
     * 设置音效音量
     *
     * @param soundId 音效文件的sourceId
     * @param volume 音量 区间0-100
     *
     */
    public abstract void setAudioEffectVolume(int soundId, int volume);
```

- setAudioAccompanyVolume：设置伴奏音量。

  同时设置播放和推流的音量。

```
    /**
     * 设置伴奏音量
     * @param volume 音量 区间0-100
     */
    public abstract void setAudioAccompanyVolume(int volume);
```

- enterSeat：上麦。

  切换互动角色。

```
    /**
     * 上麦
     */
    public abstract void enterSeat();
```

- leavelSeat：下麦。

  切换观众角色。

```
    /**
     * 下麦
     */
    public abstract void leaveSeat();
```

- setRTCAudioLiveRoomDelegate：设置监听回调。

```
       /**
     * 设置监听
     *
     * @param audioLiveRoomDelegate 监听
     */
    public abstract void setRTCAudioLiveRoomDelegate(RTCAudioLiveRoomDelegate audioLiveRo
omDelegate);
```

- muteLocalMic：设置是否静音。

```
    /**
     * 是否开启静音模式
     *
     * @param mute true为静音 false不静音
     * @return 0表示设置成功，反之失败
     */
    public abstract int muteLocalMic(boolean mute);
```

- muteRomteAudioPlaying：设置是否本地静音。

  true表示本地听不到远端的声音，false表示本地可以听到远端声音。

```
    /**
     * 停止远端的所有音频流的播放。返回0为成功，其他返回错误码。
     *
     * @param enableSpeakerPhone true为开启 false关闭
     * @return 0表示设置成功，反之失败
     */
    public abstract int muteAllRemoteAudioPlaying(boolean enableSpeakerPhone);
```

- enableEarBack：是否开启耳返。

```
    /**
     * 是否耳返
     *
     * @param enableEarBack 是否开启耳返 true为开启 false关闭
     * @return 0表示设置成功，反之失败
     */
    public abstract int enableEarBack(boolean enableEarBack);
```

- startAudioAccompany：播放伴奏。

```
    /**
     * 开始伴奏
     *
     * @param fileName      伴奏文件路径，支持本地文件和网络url
     * @param onlyLocalPlay 是否仅本地播放，true表示仅仅本地播放，false表示本地播放且推流到远端
     * @param replaceMic    是否替换mic的音频流，true表示伴奏音频流替换本地mic音频流，false表示伴奏音频流和mic音频流同时推
     * @param loopCycles    循环播放次数，-1表示一直循环
     */
    public abstract void startAudioAccompany(String fileName, boolean onlyLocalPlay, boolean replaceMic, int loopCycles);
```

- stopAudioAccompany：停止伴奏。

```
    /**
     * 停止伴奏
     */
    public abstract void stopAudioAccompany();
```

- playAudioEffect：播放音效。

```
    /**
     * 播放音效
     *
     * @param soundId  音效ID
     * @param filePath 音效文件路径，支持本地文件和网络url
     * @param cycles   循环播放次数。-1表示一直循环
     * @param publish  是否将音效音频流推到远端
     */
    public abstract void playAudioEffect(int soundId, String filePath, int cycles, boolean publish);
```

- stopAudioEffect：停止音效。

```
    /**
     * 停止音效
     *
     * @param soundId 音效ID
     */
    public abstract void stopAudioEffect(int soundId);
```

- setAudioEffectReverbMode：设置音效场景（混音模式）。

```
    /**
     * 设置音效场景
     * @param aliRtcAudioEffectReverbMode 音效场景枚举
     */
    public abstract void setAudioEffectReverbMode(AliRtcEngine.AliRtcAudioEffectReverbMode aliRtcAudioEffectReverbMode);
```

- setAudioEffectVoiceChangerMode：设置变声音效模式。

```
    /**
     * 设置变声音效模式
     *
     * @param mode 模式
     * @return int 结果码 0为成功
     */
    public abstract int setAudioEffectVoiceChangerMode(AliRtcEngine.AliRtcAudioEffectVoic
eChangerMode mode);
```

回调接口

- onEnterSeat：用户上麦通知。

```
    /**
     * 上麦通知回调
     *
     * @param seatInfo 麦位信息
     */
    void onEnterSeat(SeatInfo seatInfo);
```

- onLeaveSeat：用户下麦通知。

```
    /**
     * 下麦通知回调
     *
     * @param seatInfo 麦位信息
     */
    void onLeaveSeat(SeatInfo seatInfo);
```

- onOccurError：错误信息通知。

  当错误码是以下几种情况时，需要销毁SDK实例：

  - ErrorCodeEnum.ERR_ICE_CONNECTION_HEARTBEAT_TIMEOUT
  - ErrorCodeEnum.ERR_SDK_INVALID_STATE
  - ErrorCodeEnum.ERR_SESSION_REMOVED

```
    /**
     * sdk报错
     * @param error 错误码
     */
    void onOccurError(int error);
```

- onUpdateRoleNotify：切换用户角色回调。

```
    /**
     * 角色切换成功
     *
     * @param oldRole 旧的用户角色
     * @param newRole 新的用户角色
     */
    void onUpdateRoleNotify(AliRtcEngine.AliRTCSDK_Client_Role oldRole, AliRtcEngine.AliR
TCSDK_Client_Role newRole);
```

- onPublishChangedNotify：推流回调。

```
    /**
     * 推流结果回调
     * @param result 返回码 0表示推流成功，反之失败
     * @param isPublished 是否再推流
     */
    void onPublishChangedNotify(int result, boolean isPublished);
```

- onJoinChannelResult：加入频道结果回调。

  result为0表示加入房间成功，反之失败。

```
    /**
     * 登录回调
     *
     * @param result 状态码
     * @param uid 自己的uid
     */
    void onJoinChannelResult(int result, String uid);
```

- onLeaveChannelResult：离开频道结果回调。

```
    /**
     * 退出房间回调
     * @param result 退出房间回调 0表示成功 反之失败
     */
    void onLeaveChannelResult(int result);
```

- onSeatVolumeChanged：音量大小提示。

  当某个用户说话状态改变是才会回调（从说话到不说话，或者从不说话到说话）。

```
    /**
     * 用户音量更新回调
     * @param seatIndex 麦序
     * @param isSpeaking 是否正在说话
     */
    void onSeatVolumeChanged(int seatIndex, boolean isSpeaking);
```

- onAudioPlayingStateChanged：伴奏播放回调。

  背景音乐播放状态的回调。

```
    /**
     * 播放状态更新回调
     *
     * @param audioPlayingStatus 当前播放状态
     */
    void onAudioPlayingStateChanged(AliRtcEngine.AliRtcAudioPlayingStateCode audioPlaying
Status);
```

- onUserAudioMuted：用户muteAudio通知回调。

  某个用户静音时回调该用户当前的麦序和静音状态。

```
    /**
     * 用户静音回调
     *
     * @param seatIndex 麦序
     * @param mute      是否静音
     */
    void onSeatMutedChanged(int seatIndex, boolean mute);
```

- onRoomDestroy：房间被销毁回调。

```
    /**
     * 房间被销毁回调
     */
    void onRoomDestroy();
```

- onNetworkQualityChanged：网络状态回调。

```
    /**
     * 网络状态回调
     *
     * @param aliRtcNetworkQuality1 下行网络质量
     * @param aliRtcNetworkQuality  上行网络质量
     * @param s                     String  用户ID
     */
    void onNetworkQualityChanged(String s, AliRtcEngine.AliRtcNetworkQuality aliRtcNetwor
kQuality, AliRtcEngine.AliRtcNetworkQuality aliRtcNetworkQuality1);
```

# 2.4.4. iOS集成

通过阅读本文，您可以了解语音聊天室iOS端的集成操作。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > 说明
   > - Demo源码中没有集成iOS AliRTC SDK，需要手动进行集成。
   > - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 下载并解压SDK。

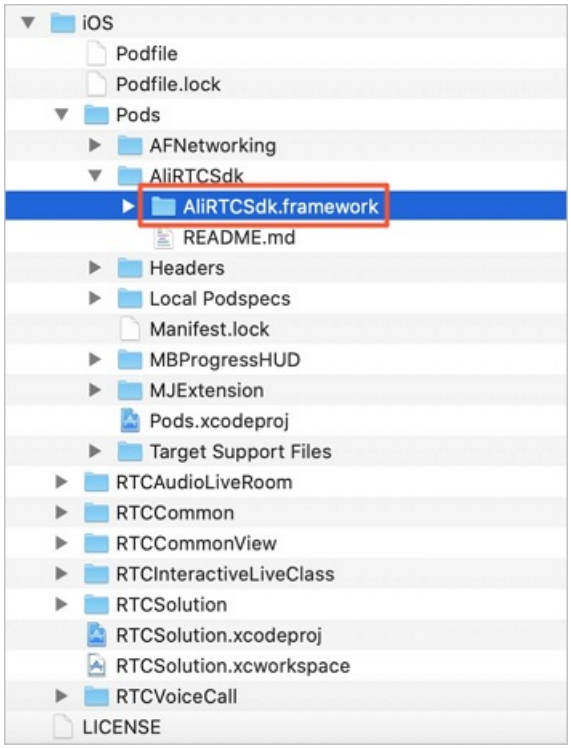
3. 打开终端，定位到Podfile文件所在的目录，执行**pod install**命令。

   安装成功如下图所示。安装后生成的Pods文件夹已在iOS目录下。

```
Installing RTCInteractiveLiveClass (0.1.0)
Installing RTCVoiceCall (0.1.0)
Generating Pods project
Integrating client project
Pod installation complete! There are 7 dependencies from the Podfile and 9 total pods installed.

[!] Automatically assigning platform `iOS` with version `9.0` on target `RTCSolution` because no platform was specified. Please specify a platform for this t
arget in your Podfile. See `https://guides.cocoapods.org/syntax/podfile.html#platform`.
```

4. 替换SDK。

   将步骤二解压的*AliRTCSdk.framework*文件，替换掉步骤三安装后生成的*Pods/AliRTCSdk/AliRTCSdk.framework*文件。

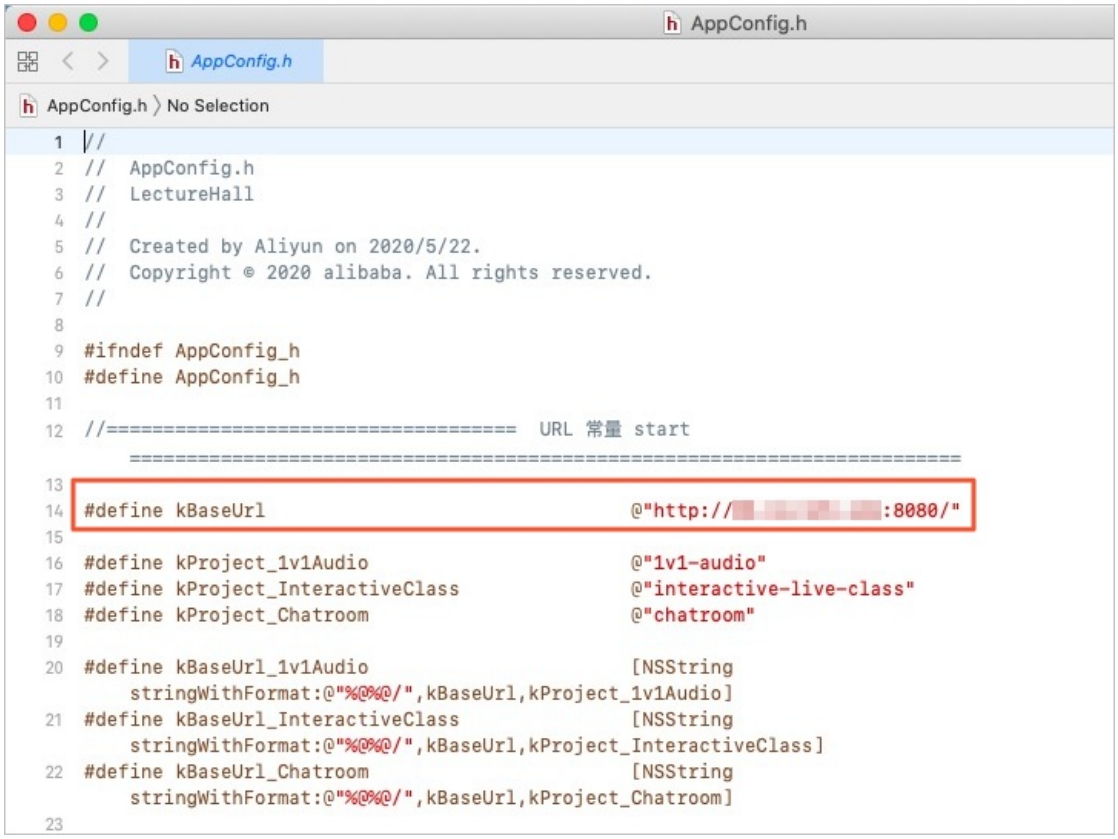



5. 配置Demo工程。

   i. 打开*iOS/RTCCommon/AppConfig.h*文件。

ii. 修改文件中的 `kBaseUrl` 值。

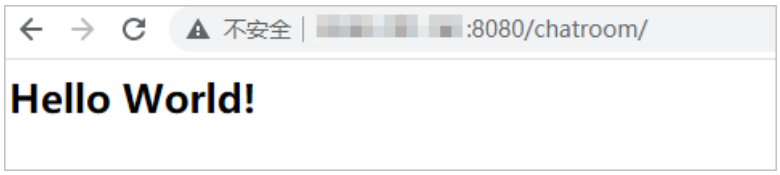
例如本地服务端IP地址为192.0.2.1，则 `kBaseUrl` 的值为 `http://192.0.2.1:8080/` 。本地服务端IP地址查询，请参见查询IP地址。




> 注意
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `kBaseUrl` 的值为 `http://<域名>/` 。

iii. 验证移动端和服务端。

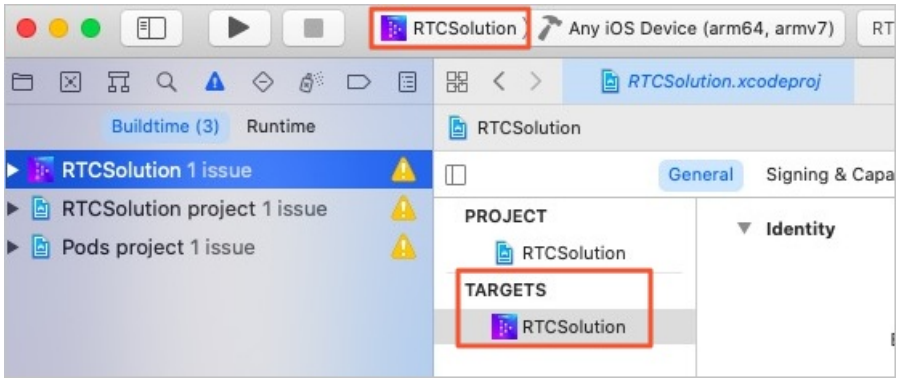
分别在移动端和服务端浏览器中访问 `http://<服务器IP> :8080/chatroom` 地址，如果显示如下图所示，表示移动端访问服务端正常。




6. 运行Demo。

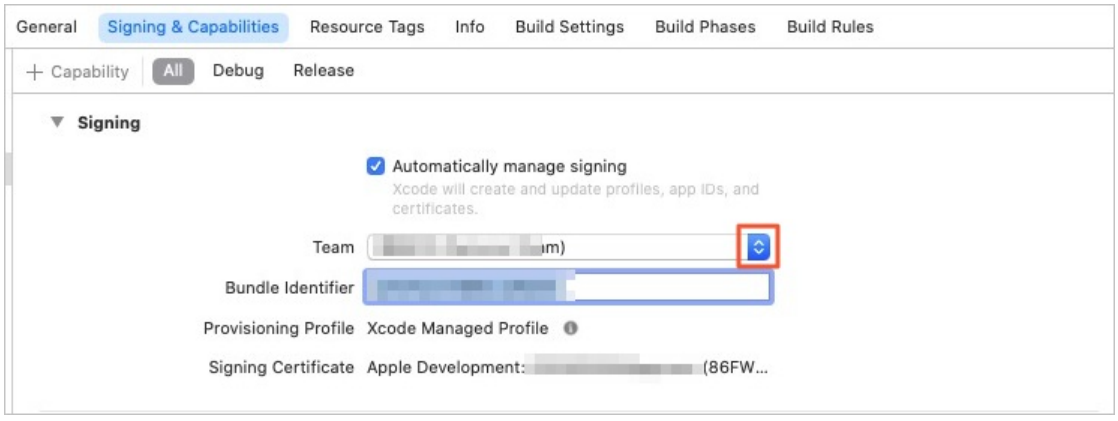
i. 使用Xcode打开*iOS*目录下的*RTCSolution.xcworkspace*工程文件。

ii. 选择运行的Target为RTCSolution，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。



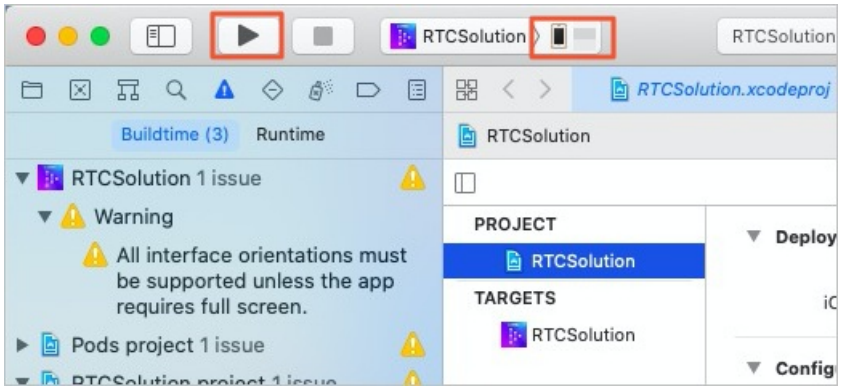

iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。


iv. 单击**Signing & Capabilities**页签，选中**Automatically manage signing**，然后单击**Team**下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择**Add an Account...**，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。



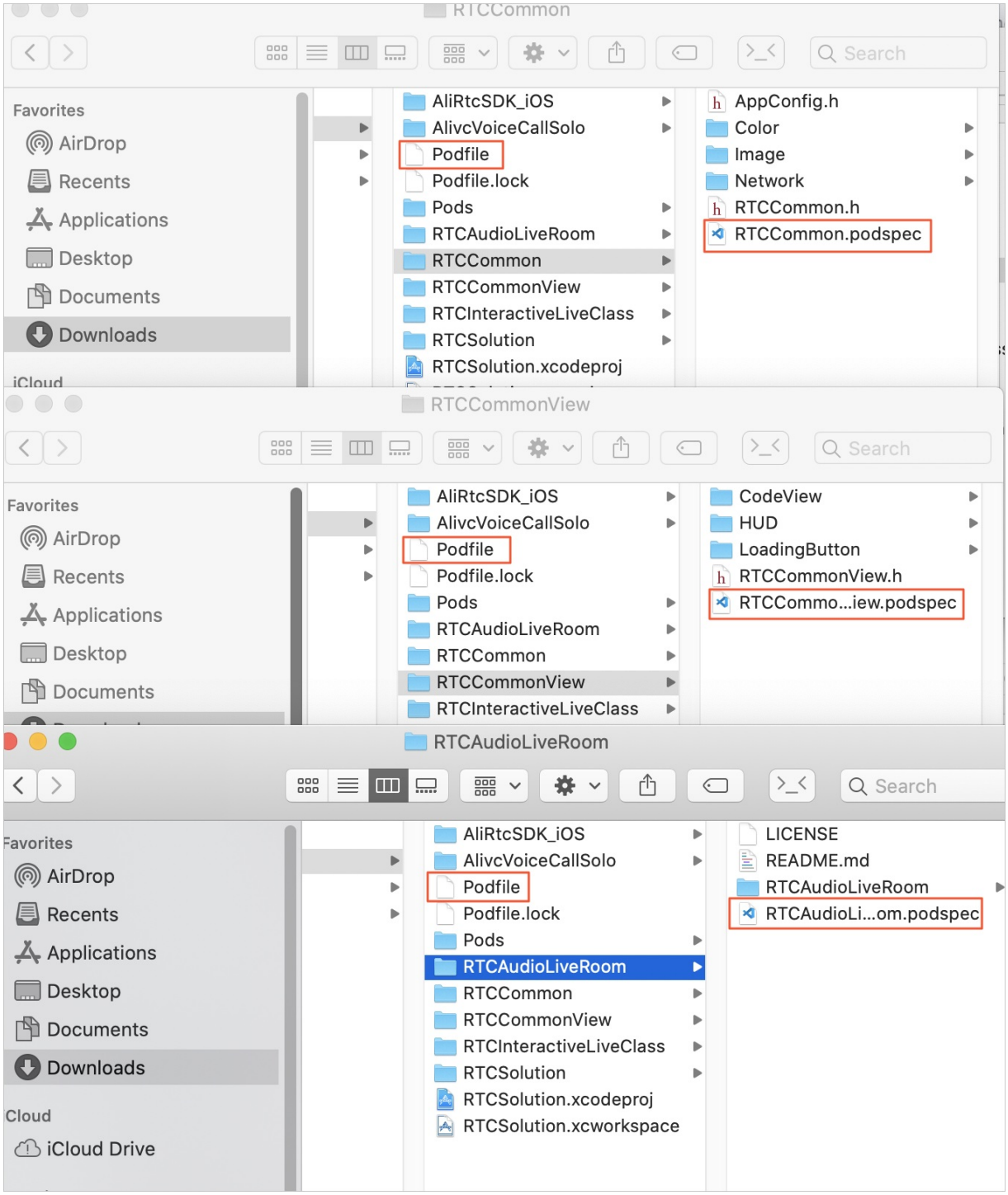

7. 加入语音聊天室。
   i. 将2台或2台以上移动端设备安装Demo App。
   ii. 将设备连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间号和昵称，选择角色进入语聊房并等待他人加入。
   iv. 在其余设备上输入相同房间号和任意昵称（可同名，不做限制），选择角色加入房间并进行聊天。

## Demo目录结构说明

RTC把开发的业务代码封装到RTCAudioLiveRoom库中，因此只需在Podfile中指定RTCAudioLiveRoom库的路径，RTCAudioLiveRoom就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
#基础组件 :path 的路径为podfile 和 RTCCommon.podspec 的相对路径
pod 'RTCCommon', :path => 'RTCCommon/'
#基础UI组件 :path 的路径为podfile 和 RTCCommonView.podspec 的相对路径
pod 'RTCCommonView', :path => 'RTCCommonView/'
#语聊房 :path 的路径为podfile 和 RTCAudioLiveRoom.podspec 的相对路径
pod 'RTCAudioLiveRoom', :path => 'RTCAudioLiveRoom/'
```

RTCAudioLiveRoom组件库目录说明，如下所示：




## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destroySharedInstance | 销毁实例对象。 |

| API | 描述 |
| --- | --- |
| login | 加入频道。 |
| logout | 退出频道。 |
| enterSeat | 上麦。 |
| leavelSeat | 下麦。 |
| renotifySeatsInfo | 重新通过回调通知座位信息。 |
| muteLocalMic | 静音或取消静音。 |
| muteAllRemoteAudioPlaying | 关闭或开启远端声音。 |
| startAudioAccompanyWithFile | 播放背景音乐。 |
| stopAudioAccompany | 停止播放背景音乐。 |
| setAudioAccompanyVolume | 设置伴奏音量。 |
| playEffectSoundtWithSoundId | 播放音效。 |
| stopAudioEffectWithSoundId | 停止播放音效。 |
| setAudioEffectPlayoutVolumeWithSoundId | 设置音效的音量。 |
| enableEarBack | 设置耳返。 |
| setAudioEffectReverbMode | 设置混响模式。 |
| setAudioEffectVoiceChangerMode | 设置音效混响模式。 |

回调接口

| API | 描述 |
| --- | --- |
| onEnterSeat | 远端用户上麦通知。 |
| onLeaveSeat | 远端用户下线通知。 |
| onRoomDestory | 房间被销毁通知。 |
| onSeatVolumeChanged | 音量变化通知。 |
| onSeatMutedChanged | 静音或取消静音变化通知。 |

功能实现接口

- 获取单例对象。

```
// 单例模式 初始化RTCAudioliveRoom
RTCAudioliveRoom *manager = [RTCAudioliveRoom sharedInstance];
//设置代理对象
manager.delegate = vc
```

- 销毁实例对象。

```
[self.manager destroySharedInstance];
```

- 加入频道。

```
/// 加入频道
/// @param channelId   频道名称
/// @param name    任意用于显示的用户名称。不是User ID
/// @param role   角色
/// @param handler   回调
 [self.manager login:@"频道名称"
                name:@"用户昵称"
                role:@"角色"
            complete:^(AliRtcAuthInfo * _Nonnull authInfo, NSInteger errorCode) {
        if (authInfo)
        {
            //加入房间成功
            // ....
            return;
        }
        //加入房间失败
    }];
```

- 退出频道。

```
[self.manager logout];
```

- 上麦。

```
[self.manager enterSeat];
```

- 下麦。

```
[self.manager leavelSeat];
```

- 重新通过回调通知座位信息。

```
[self.manager renotifySeatsInfo];
```

- 静音或取消静音。

```
//静音
[self.manager muteLocalMic:YES];
//取消静音
[self.manager muteLocalMic:NO];
```

- 关闭或开启远端声音。

```
//关闭远端声音
[self.manager muteAllRemoteAudioPlaying:YES];
//开启远端声音
[self.manager muteAllRemoteAudioPlaying:NO];
```

- 播放背景音乐。

```
/// 播放背景音乐
/// @param filePath 文件路径
/// @param publish 是否推送远端
[self.manager startAudioAccompanyWithFile:@"文件url" publish:YES];
```

- 停止播放背景音乐。

```
[self.manager stopAudioAccompany];
```

- 设置伴奏音量。

```
/// 设置背景音乐音量
/// @param volume 音量 0~100
[self.manager setAudioAccompanyVolume:100];
```

- 播放音效。

```
/// 播放音效
/// @param soundId 音效id
/// @param filePath 资源路径
/// @param publish 是否推送远端
[self.manager playEffectSoundtWithSoundId:111
                                 filePath:@"文件url"
                                 publish:YES];
```

- 停止播放音效。

```
[self.manager stopAudioEffectWithSoundId:111];
```

- 设置音效的音量。

```
[self.manager setAudioEffectPlayoutVolumeWithSoundId:111 volume:100];
```

- 设置耳返。

```
//开启耳返
[self.manager enableEarBack:YES];
//关闭耳返
[self.manager enableEarBack:NO];
```

- 设置混响模式。

```
[self.manager setAudioEffectReverbMode:AliRtcAudioEffectReverb_Off];
```

- 设置音效混响模式。

```
[self.manager setAudioEffectVoiceChangerMode:AliRtcAudioEffectvVoiceChanger_OFF];
```

回调接口

- 远端用户上麦通知。

```
/// 远端用户上麦通知
/// @param seat 麦序
- (void)onEnterSeat:(SeatInfo *)seat;
```

- 远端用户下线通知。

```
/// 远端用户下线通知
/// @param seat 麦序
- (void)onLeaveSeat:(SeatInfo *)seat;
```

- 房间被销毁通知。

```
/// 房间被销毁通知
- (void)onRoomDestory;
```

- 音量变化通知。

```
/// 音量变化通知
/// @param seatIndex 麦序
/// @param isSpeaking 是否在说话
- (void)onSeatVolumeChanged:(NSInteger)seatIndex isSpeaking:(BOOL)isSpeaking;
```

- 静音或取消静音变化通知。

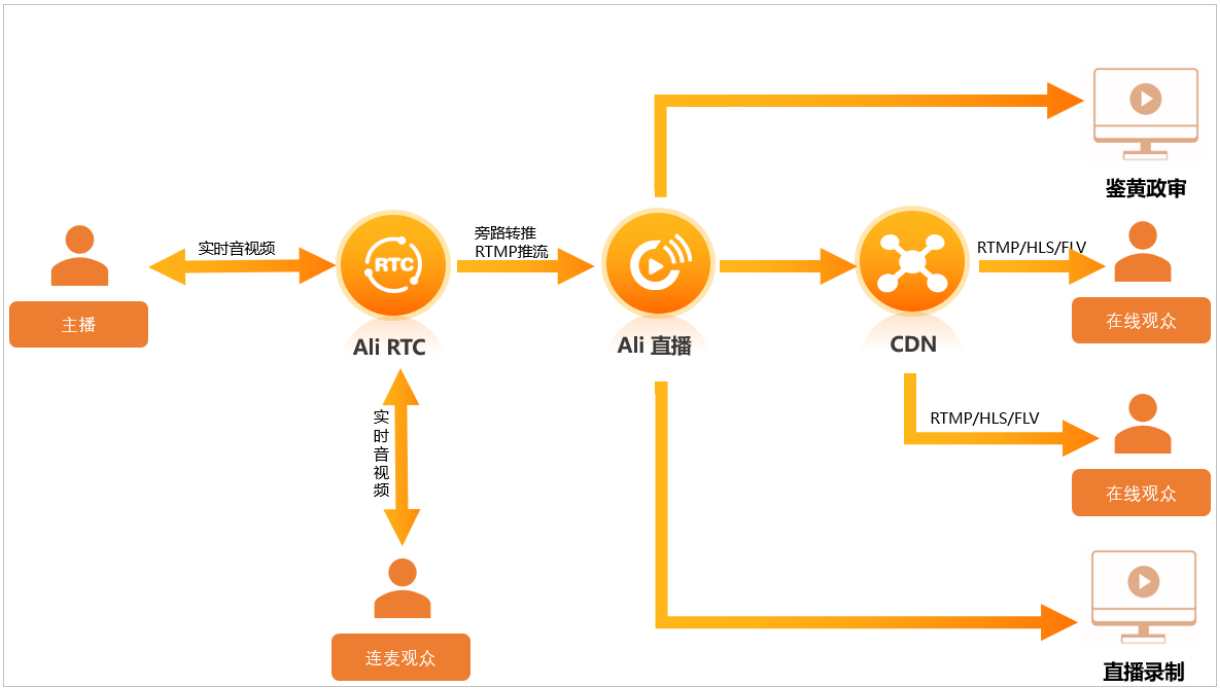
```
/// 静音/取消静音变化通知
/// @param seatIndex 麦序
/// @param mute 是否静音
- (void)onSeatMutedChanged:(NSInteger)seatIndex mute:(BOOL)mute;
```

# 2.5. 视频互动直播

## 2.5.1. 简介

视频互动直播服务支持观众与主播连麦互动（延时小于300ms），同时为主播提供伴奏、播放鼓励音效、设置混响音效等功能。通过阅读本文，您可以快速了解视频互动直播的基本信息及实现方法。

### 架构方案

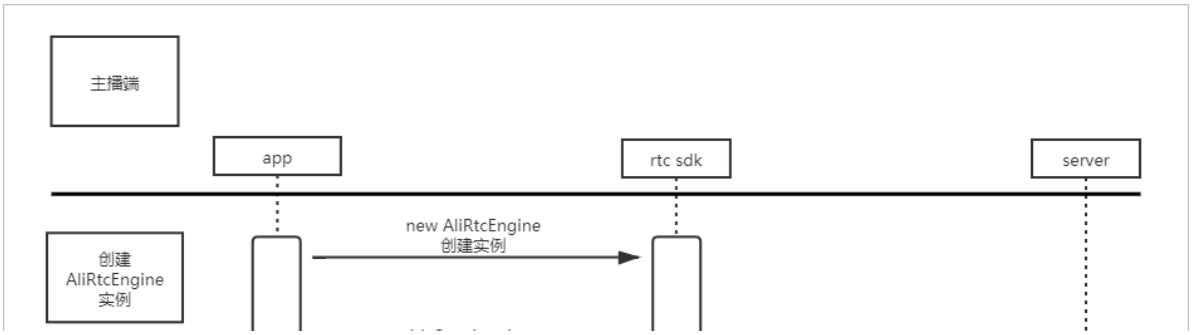


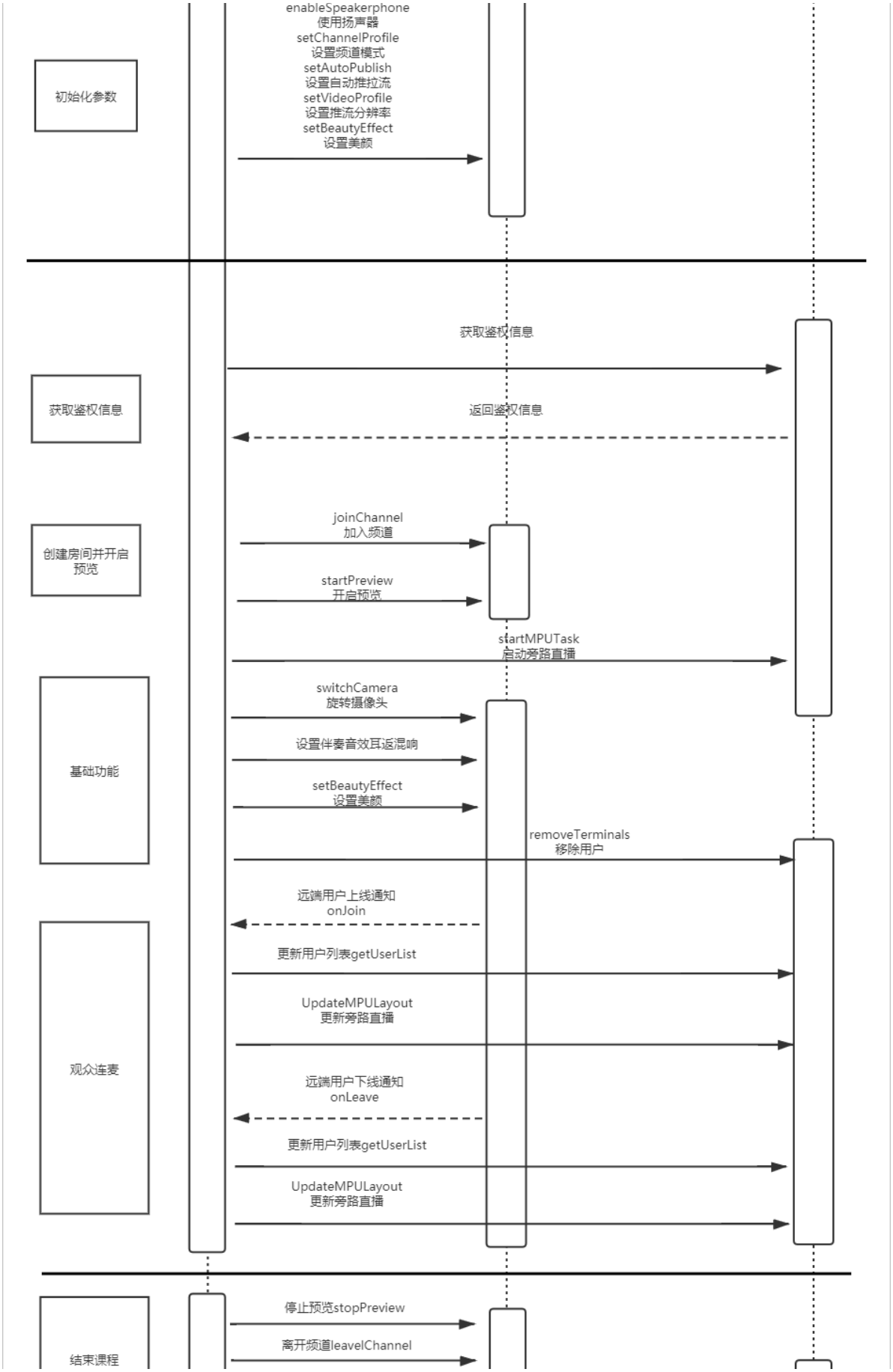

## 主要功能

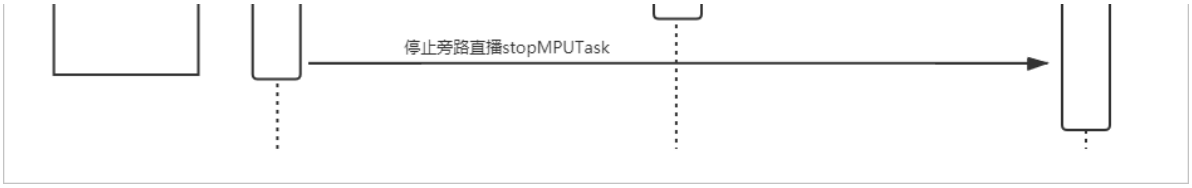
| 功能 | 描述 |
| --- | --- |
| 实时视频连麦 | 支持在线观众与主播连麦互动，频道内所有观众都可以看到连麦画面，连麦延时在300毫秒以下。 |
| 播放背景音乐 | 主播可以在直播中播放背景音乐，营造直播气氛。 |
| 播放鼓励音效 | 主播可以在直播中播放大笑、掌声、哭声等鼓励音效，烘托直播氛围。 |
| 耳返 | 主播可以开启耳返功能，实时监听自己的声音，极大提升主播的专业度。 |
| 混响音效 | 支持多种混响预设效果。 |
| 基础美颜 | 支持美白和磨皮基础美颜功能，帮助主播在直播过程中具备更好的面容。 |

## 实现方法

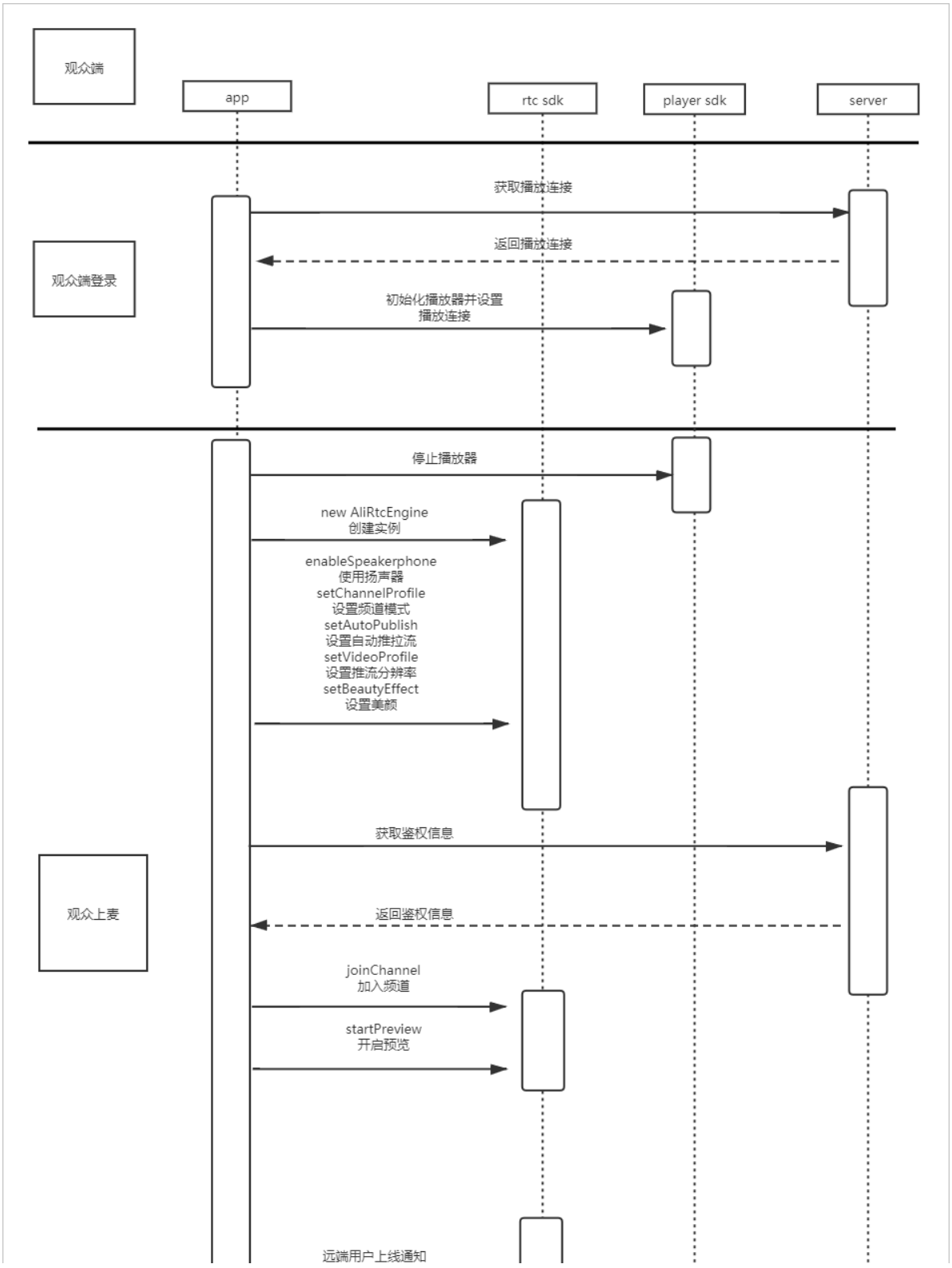
- 主播端时序图：

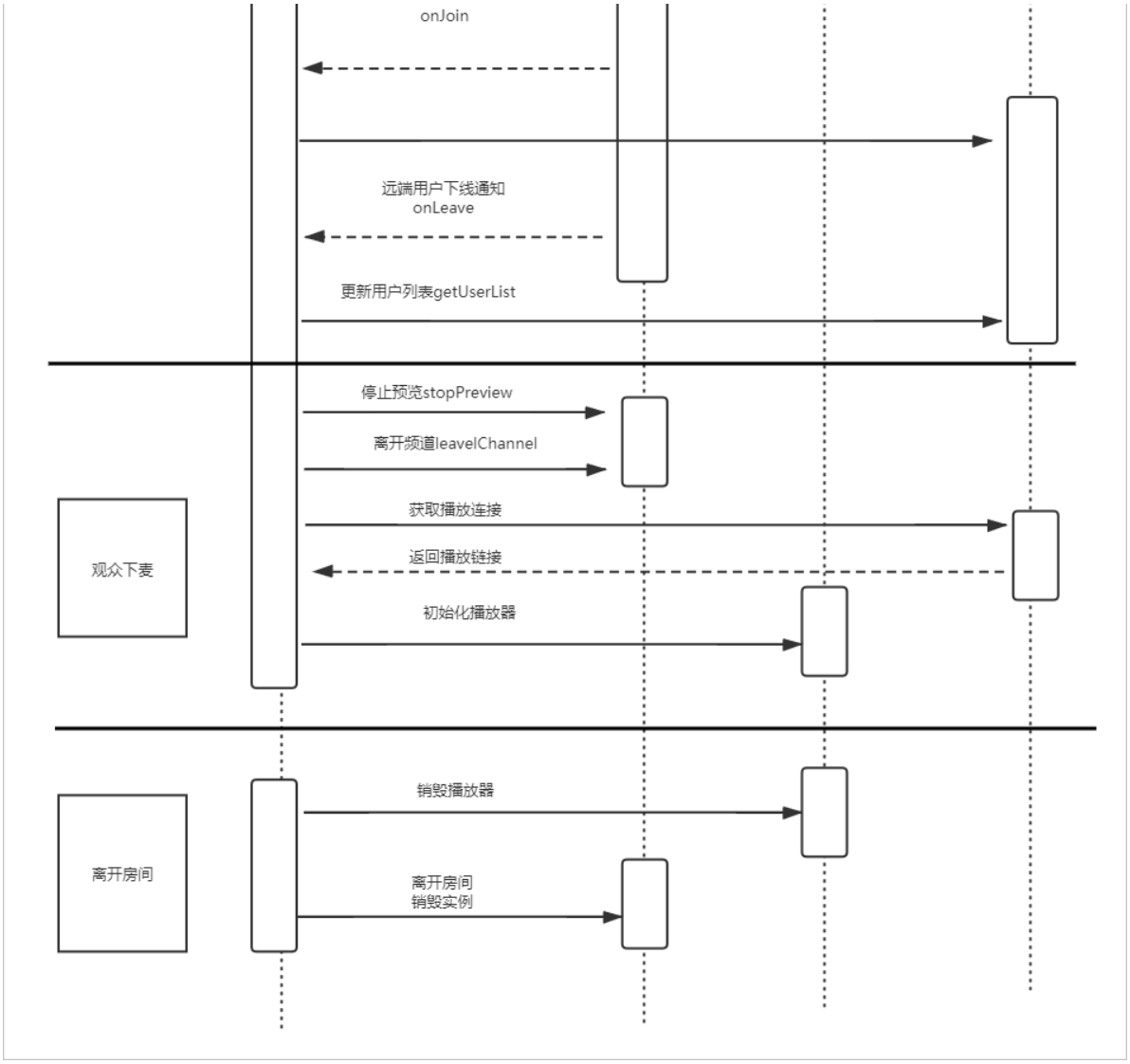



初始化参数
enableSpeakerphone
使用扬声器
setChannelProfile
设置频道模式
setAutoPublish
设置自动推拉流
setVideoProfile
设置推流分辨率
setBeautyEffect
设置美颜
获取鉴权信息
获取鉴权信息
返回鉴权信息
创建房间并开启预览
joinChannel
加入频道
startPreview
开启预览
startMPUTask
启动旁路直播
基础功能
switchCamera
旋转摄像头
设置伴奏音效耳返混响
setBeautyEffect
设置美颜
removeTerminals
移除用户
观众连麦
远端用户上线通知
onJoin
更新用户列表getUserList
UpdateMPULayout
更新旁路直播
远端用户下线通知
onLeave
更新用户列表getUserList
UpdateMPULayout
更新旁路直播
结束课程
停止预览stopPreview
离开频道leavelChannel


停止旁路直播stopMPUTask

- 观众端时序图：

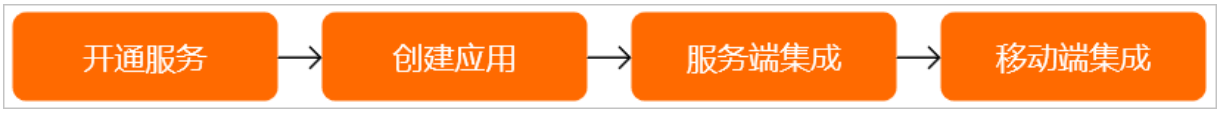

观众端
app
rtc sdk
player sdk
server
获取播放连接
返回播放连接
观众端登录
初始化播放器并设置
播放连接
停止播放器
new AliRtcEngine
创建实例
enableSpeakerphone
使用扬声器
setChannelProfile
设置频道模式
setAutoPublish
设置自动推拉流
setVideoProfile
设置推流分辨率
setBeautyEffect
设置美颜
获取鉴权信息
观众上麦
返回鉴权信息
joinChannel
加入频道
startPreview
开启预览
远端用户上线通知




观众端时序图中包含RTC和视频直播服务，RTC服务提供连麦和推流到直播中心的功能。视频直播服务提供推流、拉流地址和播放器部分功能。更多信息，请参见视频直播。

## 实现流程

实现流程如下图所示：




| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务<br>开通视频直播服务 | 进行服务端端集成之前，您必须开通音视频通信和视频直播服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |

| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 3 | 服务端集成 | 您可以通过源码快速搭建服务端视频直播互动。 |
| 4 | 移动端集成：<br>• iOS集成<br>• Android集成 | 您可以通过源码快速搭建移动端视频直播互动。 |

### Demo体验

您可以通过钉钉扫描以下二维码，下载安装视频互动直播Demo体验。

## 2.5.2. 服务端集成

通过阅读本文，您可以了解视频互动直播服务端的集成操作。

### 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 您已经开通视频直播服务。开通步骤请参见开通与购买视频直播。
- 域名已成功备案，备案域名，请前往阿里云ICP代备案管理系统。
- 环境中已安装Java JDK 8的版本。具体操作，请参见安装JDK。

  > **说明** 如果服务端为Linux环境，推荐安装Oracle JDK，不推荐使用Open JDK进行服务端集成。

- 服务端环境已安装MySQL（建议安装MySQL 5.7版本）。具体操作，请参见安装MySQL。

### 使用限制

如果启用旁路转推服务超过10个并发量时，您需要和客户经理沟通或提交工单（选择视频直播）。

### 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。

ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。

iii. 在**AppID/名称**列获取AppID。

iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 获取AccessKey。

> **说明** 由于主账号AccessKey泄露会威胁您所有资源的安全，因此出于安全考虑，需要创建一个子账号（RAM用户）并获取子账号的AccessKey，用于访问您的云资源。

i. 登录RAM控制台。

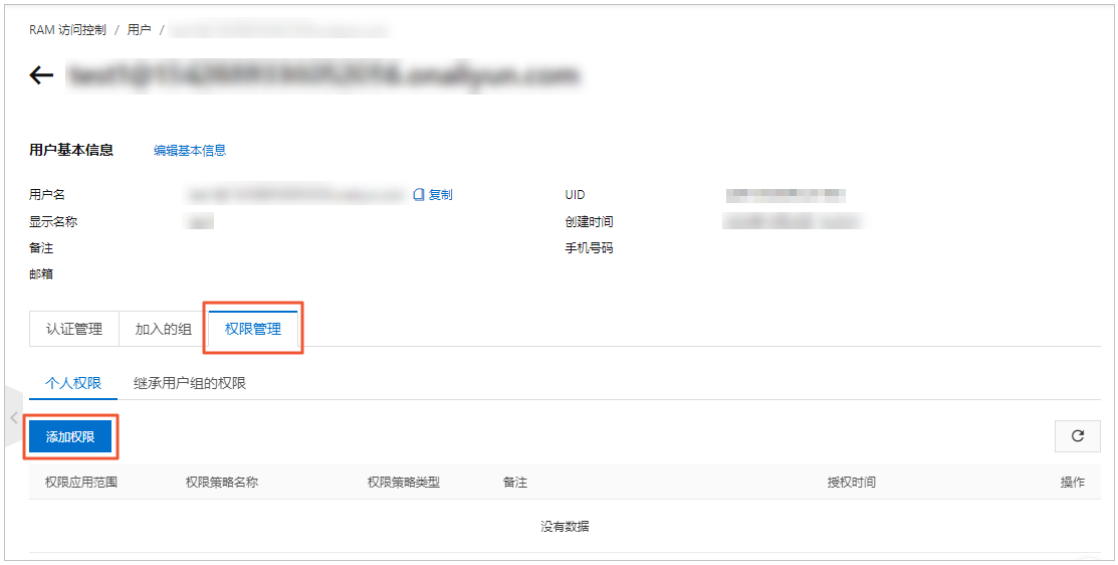
ii. 在左侧导航栏选择**身份管理 > 用户**，进入用户页面。

iii. 单击**创建用户**，填写用户账号信息，选中**Open API 调用访问**创建用户。

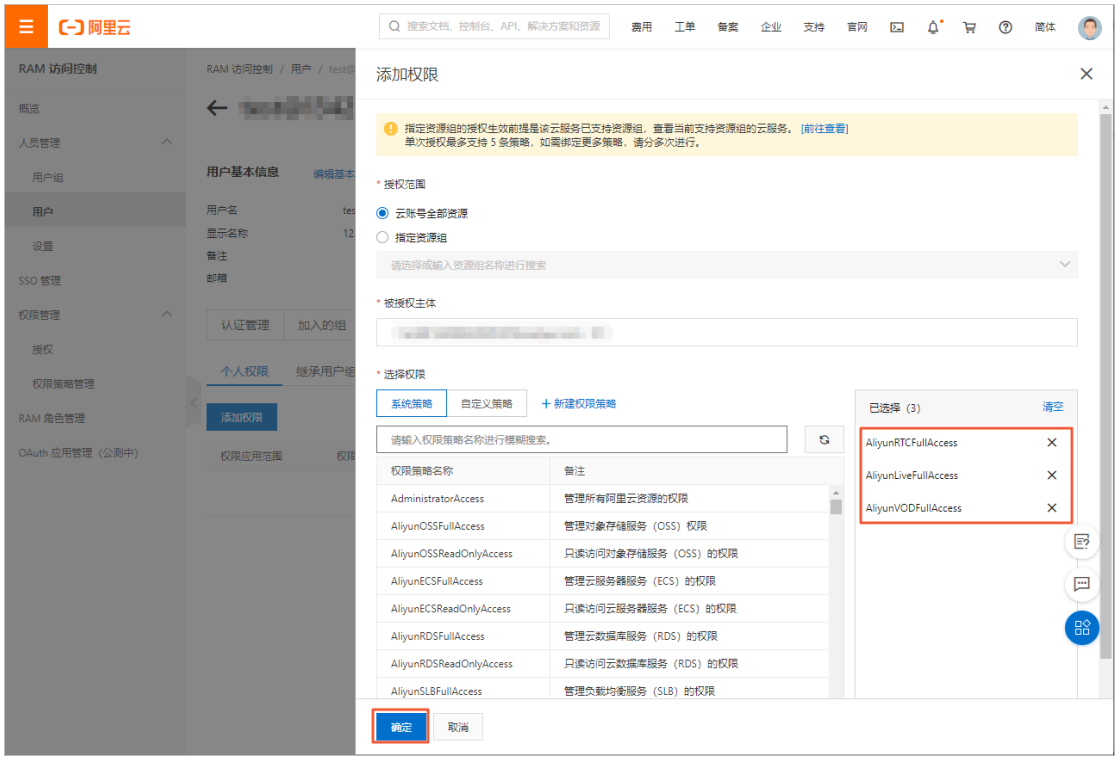

iv. 单击**确定**。

v. 重新返回用户页面。在**用户登录名称/显示名称**列下，单击目标RAM子账户，进入管理页面。

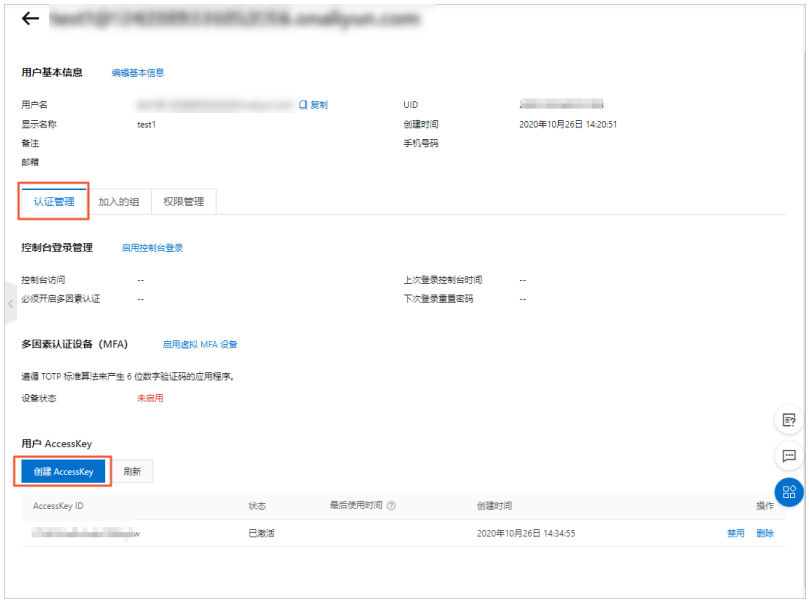
vi. 单击**权限管理**，再单击**添加权限**。




vii. 分别输入RTC、Live、VOD进行搜索，选择AliyunRTCFullAccess（管理音视频通信的权限）、AliyunLiveFullAccess（管理视频直播的权限）和AliyunVODFullAccess（管理视频点播服务的权限），单击**确定**。



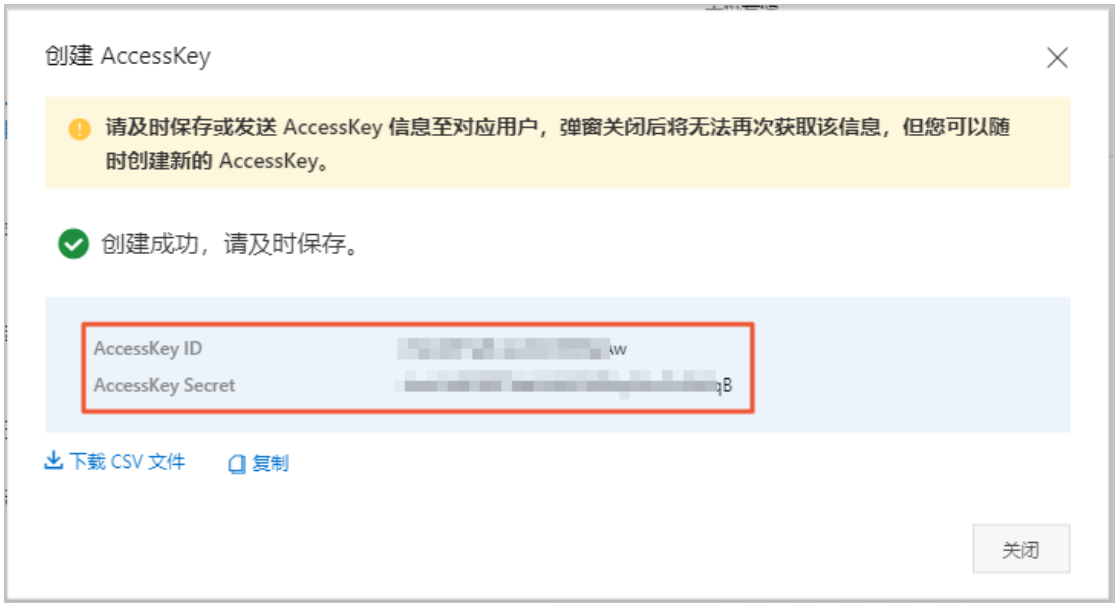
viii. 单击**认证管理**，在用户AccessKey区域单击**创建AccessKey**。



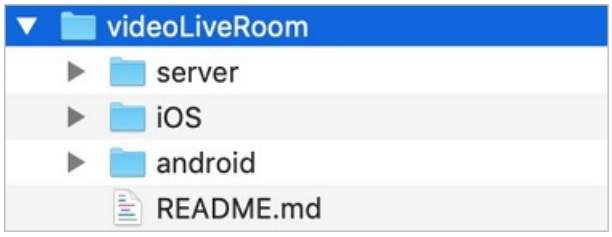

> **说明**
>
> - 首次创建时需填写手机验证码。
> - 如果AccessKey泄露或丢失，则需要创建新的AccessKey，最多可以创建2个AccessKey。

ix. 获取AccessKey ID和AccessKey Secret。

创建AccessKey完成后，会弹出**创建AccessKey**对话框，包含AccessKey ID和AccessKey Secret信息。此处建议记录一下AccessKey ID和AccessKey Secret，方便后续操作中使用。

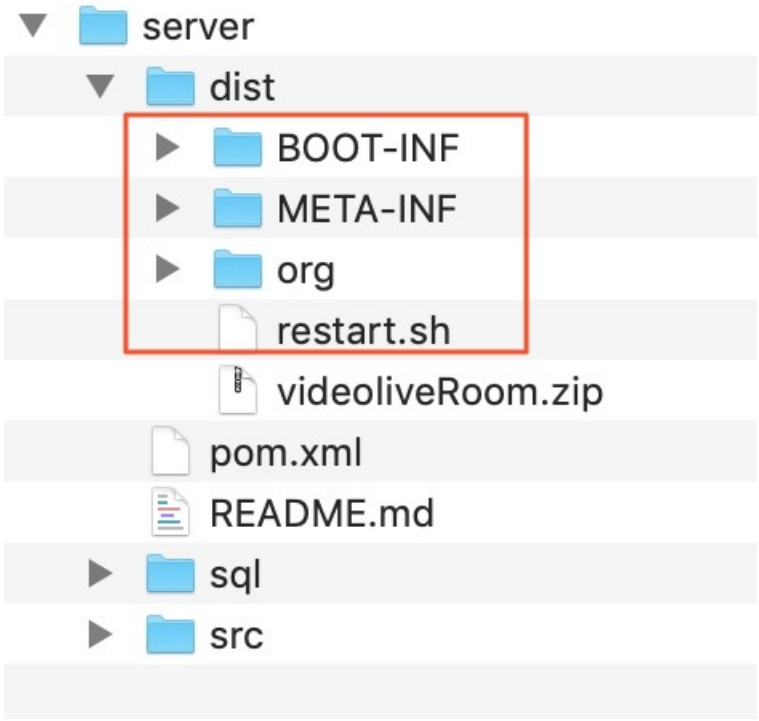



3. 获取推流地址和播放地址。详情请参见获取推流与播放地址。
4. 下载并解压Demo源码。




5. 将*server/dist/videoliveRoom.zip*文件解压到*server/dist*文件夹下。

   解压成功之后如下图所示：




> **说明** 解压成功后，可以看到BOOT-INF、META-INF及org文件夹，这三个文件夹需要和restart.sh保持在同一个目录下。

6. 初始化数据库。

   在MySQL中创建数据库并使用sql文件建表。

   sql文件路径：*server/sql/appserver_create_table.sql*。

   i. 启动MySQL服务。
      - Windows端：**net start MySQL服务名**
      - Mac端：**sudo Mysql服务路径 start**
      - Linux端：**service mysql start**

   ii. 新建终端，并将终端定位到sql文件目录下。

   iii. 创建视频互动直播数据库。

      **mysqladmin -u root -p create videoliveroom**

   iv. 在视频互动直播数据库中执行sql文件。

      **mysql -u 用户名 -p 密码 videoliveroom < ./appserver_create_table.sql**

7. 修改配置文件。

   打开*BOOT-INF/classes/application.properties*文件，修改配置。

   > **说明** 使用文本编辑器打开即可，若找不到打开方式，推荐安装VSCode等轻量级代码编辑器打开。

```
server.port=8080
server.tomcat.accesslog.buffered=false
server.tomcat.accesslog.directory=/home
server.tomcat.accesslog.enabled=true

server.context-path=/videoliveRoom/

#################################################log#############################################################
logging.level.com.live.dao=debug
logging.level.com.live.dao.user=debug
logging.level.com.alivc.vod.dao=debug
logging.file:my.log
logging.pattern.file=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{15}.%M : %msg%n
logging.pattern.console=%d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{15}.%M - %msg%n

accessKeyID=*****
accessKeySecret=*****

rtc.liveclass.appId = *****
rtc.liveclass.appKey = *****
rtc.gslb = https://rgslb.rtc.aliyuncs.com

live_play_domain = *****
live_play_domain_auth_key = *****

live_push_domain = *****
live_push_domain_auth_key = *****

live_app_stream_name = /videoliveRoom/

durationlimit=-1


layout_1=1
layout_2=2
layout_3=5

spring.datasource.url = jdbc:mysql://*****:3306/*****?useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.datasource.username = *****
spring.datasource.password = *****

spring.http.multipart.maxFileSize=100Mb
spring.http.multipart.maxRequestSize=100Mb

# MyBatis
mybatis.mapper-locations=classpath:mapper/*/*.xml
mybatis.type-aliases-package=com.alivc.vod.pojo.*
```

   - 设置RTC应用AppId和AppKey，请参见步骤1获取。

     rtc.liveclass.appId = *

     rtc.liveclass.appKey = *

   - 设置AK，需要添加AliyunRTCFullAccess权限，请可参见步骤2获取。

     accessKeyID=*

     accessKeySecret=*

   - 设置播流地址和鉴权，请参见步骤3获取。

     live_play_domain = *

     live_play_domain_auth_key = *

   - 设置推流地址和鉴权，请参见步骤3获取。

     live_push_domain = *

live_push_domain_auth_key = *

- 修改数据库访问地址、用户名和密码。

  数据库访问地址：spring.datasource.url = jdbc:mysql://数据库IP地址（本地：127.0.0.1）:3306/数据库名（videoliveroom）?useSSL=false&useUnicode=true&characterEncoding=UTF-8

  用户名：spring.datasource.username = *

  密码：spring.datasource.password = *

  > **说明** 如果数据库和服务端都在本地运行，则数据库IP地址可以使用127.0.0.1。

8. 运行服务，执行*restart.sh*文件。如果编译过程中出现问题，请参见服务端运行常见问题。
   - Mac或Linux环境下请将终端定位至*dist*目录下，执行如下命令：

     **sh restart.sh**

     后台执行可以使用如下命令以确保退出终端时程序能够继续运行。

     **nohup sh restart.sh >./run_log.log 2>&1 &**

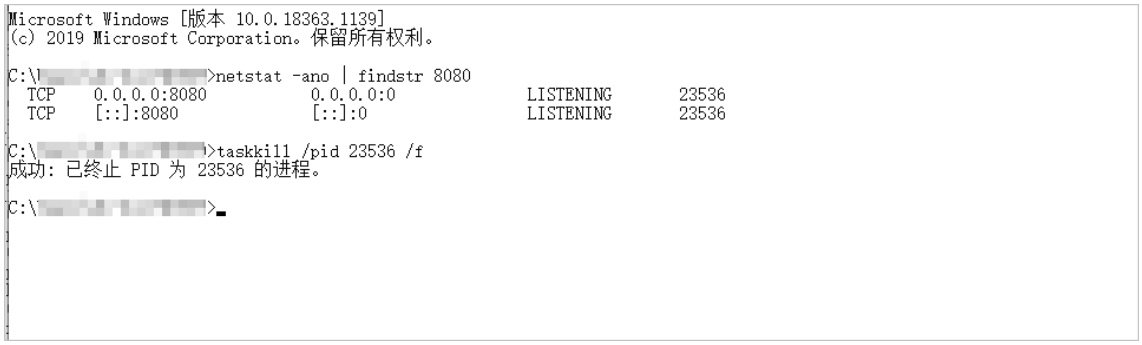
   - Windows环境需要打开CMD终端定位到*server/dist*文件夹下，执行如下命令：

     **java org.springframework.boot.loader.JarLauncher**

     若提示8080端口被占用，请尝试使用netstat命令查看占用8080端口的进程pid号，并使用taskkill关闭相关进程。

     **netstat -ano | findstr 8080**

     **taskkill /pid 占用端口的进程pid号 /f**

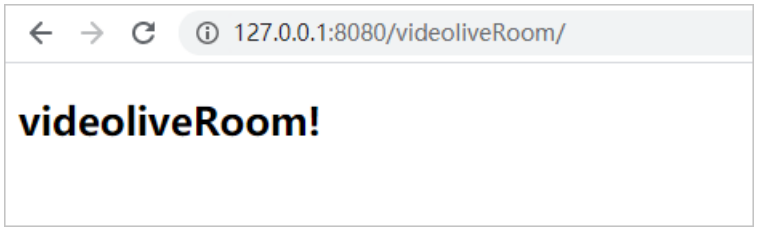



终端成功运行后可以看到服务端启动成功的日志信息。

```
tializing ExecutorService 'taskExecutor'
2020-10月-26 17:53:39.877 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapte
r.initControllerAdviceCache - Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigE
mbeddedWebApplicationContext@7ec7ffd3: startup date [Mon Oct 26 17:53:36 CST 2020]; root of context hierarchy
2020-10月-26 17:53:40.054 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappin
g.register - Mapped "{[/app/descChannelStartTime],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rt
c.controller.RtcController.descChannelStartTime(java.lang.String)
2020-10月-26 17:53:40.056 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappin
g.register - Mapped "{[/app/token],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rtc.controller.Rt
cController.getRtcToken(java.lang.String)
2020-10月-26 17:53:40.057 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappin
g.register - Mapped "{[/app/descChannelUsers],methods=[GET]}" onto public com.alivc.base.ResponseResult com.alivc.rtc.co
ntroller.RtcController.descChannelUsers(java.lang.String)
2020-10月-26 17:53:40.060 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappin
g.register - Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, jav
a.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequ
est)
2020-10月-26 17:53:40.065 INFO [main] org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMappin
g.register - Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.sprin
gframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.htt
p.HttpServletResponse)
2020-10月-26 17:53:40.111 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestH
andler]
2020-10月-26 17:53:40.112 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2020-10月-26 17:53:40.173 INFO [main] org.springframework.web.servlet.handler.SimpleUrlHandlerMapping.registerHandler -
Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequ
estHandler]
2020-10月-26 17:53:40.211 INFO [main] org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration$WelcomePageHar
dlerMapping.<init> - Adding welcome page: class path resource [resources/index.html]
2020-10月-26 17:53:40.503 INFO [main] org.springframework.jmx.export.annotation.AnnotationMBeanExporter.afterSingletons
Instantiated - Registering beans for JMX exposure on startup
2020-10月-26 17:53:40.623 INFO [main] org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.s
tart - Tomcat started on port(s): 8080 (http)
2020-10月-26 17:53:40.632 INFO [main] com.alivc.Application.logStarted - Started Application in 5.391 seconds (JVM runn
ing for 6.906)
2020-10月-26 17:53:40.633 INFO [main] com.alivc.Application.main - start success
```

9. 检查服务端是否已经启动。

   在浏览器中访问*http://127.0.0.1:8080/videoliveRoom*，如果显示如下图所示，表示服务端已经启动。




## 主要功能说明

- 获取正在直播的房间列表。

  访问地址：/getChannelList

```
List<Channel> channelList = channelService.getChannelList(lastChannelId, pageSize);
String appId = ConfigMapUtil.getValueByKey("rtc.liveclass.appId");
ResponseResult responseResult = new ResponseResult();
Iterator<Channel> channelIterator = channelList.iterator();
while (channelIterator.hasNext()) {
    Channel channel = channelIterator.next();
    String channelId = channel.getChannelId();
    DescribeChannelUsersResponse response = RtcOpenAPI.describeChannelUsers(appId, channe
lId);
    if (CollectionUtils.isEmpty(response.getUserList())) {
        channelService.deleteChannel(channelId);
        channelIterator.remove();
    }
}
responseResult.setData(channelList);
```

- 随机生成用户信息和客户端调用RTC SDK加入房间的Token信息。具体生成方式参考RTC帮助文档接入工具。

  访问地址：/getRtcAuth

```
JSONObject rtcToken = RtcOpenAPI.createToken(channelId, userId);
responseResult.setData(rtcToken);
String appId = ConfigMapUtil.getValueByKey("rtc.liveclass.appId");
```

- 开启旁路直播。

  访问地址：/startMPUTask

```
StartMPUTaskResponse response = RtcOpenAPI.startMPUTask(channelId, pushUrl, channelId, ap
pId, userId);
Channel channel = new Channel();
channel.setChannelId(channelId);
channel.setOwnerId(userId);
channel.setCoverUrl(coverUrl);
channel.setTitle(title);
channel.setCreateDateTime(LocalDateTime.now());
channelService.insertChannel(channel);
```

- 根据房间ID获取播放地址。

  访问地址：/getPlayUrl

```
String livePlayDomain = ConfigMapUtil.getValueByKey("live_play_domain");
String livePlayDomainAuthKey = ConfigMapUtil.getValueByKey("live_play_domain_auth_key");
String appStream = ConfigMapUtil.getValueByKey("live_app_stream_name");
Long timestamp = System.currentTimeMillis() / 1000;
String rand = UUID.randomUUID().toString().replace("-", "");
ResponseResult responseResult = new ResponseResult();
// 原画
String playUrlRtmp = "rtmp://" + livePlayDomain + AuthKeyUrlUtil.getAuthedPath(appStream
+ channelId, livePlayDomainAuthKey, timestamp, rand);
String playUrlFlv = "https://" + livePlayDomain + AuthKeyUrlUtil.getAuthedPath(appStream
+ channelId + ".flv", livePlayDomainAuthKey, timestamp, rand);
String playUrlM3u8 = "https://" + livePlayDomain + AuthKeyUrlUtil.getAuthedPath(appStream
+ channelId + ".m3u8", livePlayDomainAuthKey, timestamp, rand);
JSONObject playUrl = new JSONObject();
playUrl.put("rtmp", playUrlRtmp);
playUrl.put("flv", playUrlFlv);
playUrl.put("m3u8", playUrlM3u8);
```

- 房间人数变化时更新布局。

  访问地址：/updateMPULayout

```
List<String> userList = getSortedUserList(channelId);
UpdateMPULayoutRequest request = new UpdateMPULayoutRequest();
String appId = ConfigMapUtil.getValueByKey("rtc.liveclass.appId");
request.setAppId(appId);
request.setTaskId(channelId);
List<UpdateMPULayoutRequest.UserPanes> userPanesList = new LinkedList<>();
for (int i = 0; i < userList.size(); i++) {
    UpdateMPULayoutRequest.UserPanes userPanes = new UpdateMPULayoutRequest.UserPanes();
    userPanes.setUserId(userList.get(i));
    userPanes.setPaneId(i);
    userPanes.setSourceType("camera");
    userPanesList.add(userPanes);
}
request.setUserPaness(userPanesList);
String layout1Id = ConfigMapUtil.getValueByKey("layout_1");
String layout2Id = ConfigMapUtil.getValueByKey("layout_2");
String layout3Id = ConfigMapUtil.getValueByKey("layout_3");
List layoutIdArr = new ArrayList();
layoutIdArr.add(layout1Id);
layoutIdArr.add(layout2Id);
layoutIdArr.add(layout3Id);
request.setLayoutIdss(layoutIdArr);
UpdateMPULayoutResponse response = RtcOpenAPI.updateMPULayout(request);
log.info(JSON.toJSONString(response));
```

- 主播关闭直播后关闭旁路转推任务。

  访问地址：/stopMPUTask

```
StopMPUTaskResponse response = RtcOpenAPI.stopMPUTaskRequest(request);
RtcOpenAPI.deleteChannel(appId, channelId);
channelService.endChannel(channelId, LocalDateTime.now());
return responseResult;
```

- 查询正在上麦的用户列表。

  访问地址：/getUserList

```
ResponseResult responseResult = new ResponseResult();
List<String> userList;
userList = getSortedUserList(channelId);
```

- 将指定终端用户从频道踢出。

  调用访问地址：/removeTerminals

```
DescribeChannelUsersResponse response = RtcOpenAPI.describeChannelUsers(appId, channelId)
;
List<String> userList = response.getUserList();
userList.remove(operatorId);
RtcOpenAPI.removeTerminals(appId, channelId, userList);
```

# 2.5.3. Android集成

通过阅读本文，您可以了解视频互动直播Android端的集成操作。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

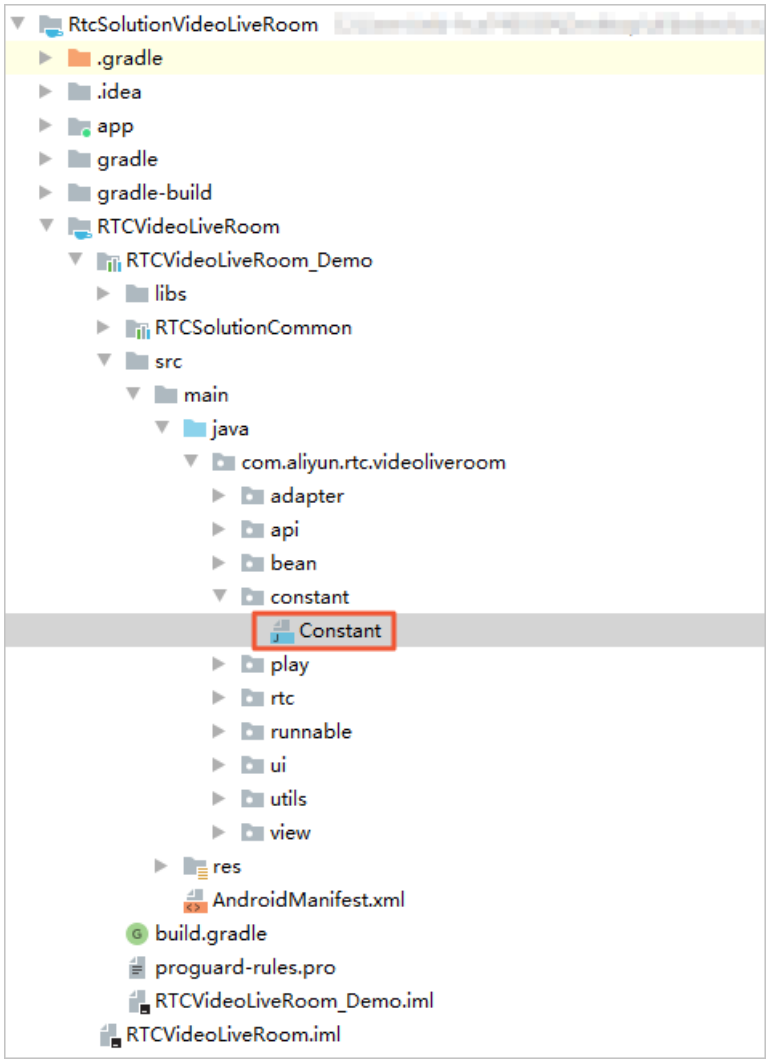
- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Android Studio 3.0或以上版本。更多信息，请参见Android Studio。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

   > **说明**
   > - Demo源码中已集成AliRTC Android SDK。
   > - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 配置Demo工程。
   i. 打开Android Studio，单击**Open an Existing Project**，选择*android*目录下的*RtcSolutionVideoLiveRoom*文件夹。
   ii. 打开*RtcSolutionVideoLiveRoom/RTCVideoLiveRoom/RTCVideoLiveRoom_Demo/src/main/java/com/aliyun/rtc/videoliveroom/constant/Constant.java*文件。




iii. 修改文件中的 `API_URL` 值。

例如本地服务端IP地址为192.0.2.1，则 `API_URL` 的值为 `http://192.0.2.1:8080/videoliveRoom` ，即 `API_URL="http://192.0.2.1:8080/videoliveRoom"` 。本地服务端IP地址查询，请参见查询IP地址。

```
package com.aliyun.rtc.videoliveroom.constant;


public class Constant {

    public static final String NEW_TOKEN_PARAMS_KEY_USERID = "userId";
    public static final String PATH_ASSETS_BGM = "mp3/bgm.zip";
    public static final String PATH_ASSETS_AUDIOEFFECT = "mp3/audioeffect.zip";
    public static final String PATH_DIR_BGM_OUT = "bgm";
    public static final String PATH_DIR_AUDIOEFFECT_OUT = "audioeffect";
    //背景乐、音效默认音量
    public static final int VALUE_AUDIO_EFFECT_VOLUME = 100;
    public static final String RTC_SP_KEY_USER_INFO = "userInfo";
    //保存随机图片的地址
    public static final String PICTURE_URL = "picture_url";
    /**
     * server端的请求域名，需要用户自己替换成自己server端的域名
     */
    private static final String API_URL = "http://  :8080/videoliveRoom";
    /**
     * 获取鉴权信息
     */
    private static final String URL_CHANNEK_LIST = API_URL + "/getChannelList";
    //加入房间成功
    private static final String URL_RTC_AUTH = API_URL + "/getRtcAuth";
```

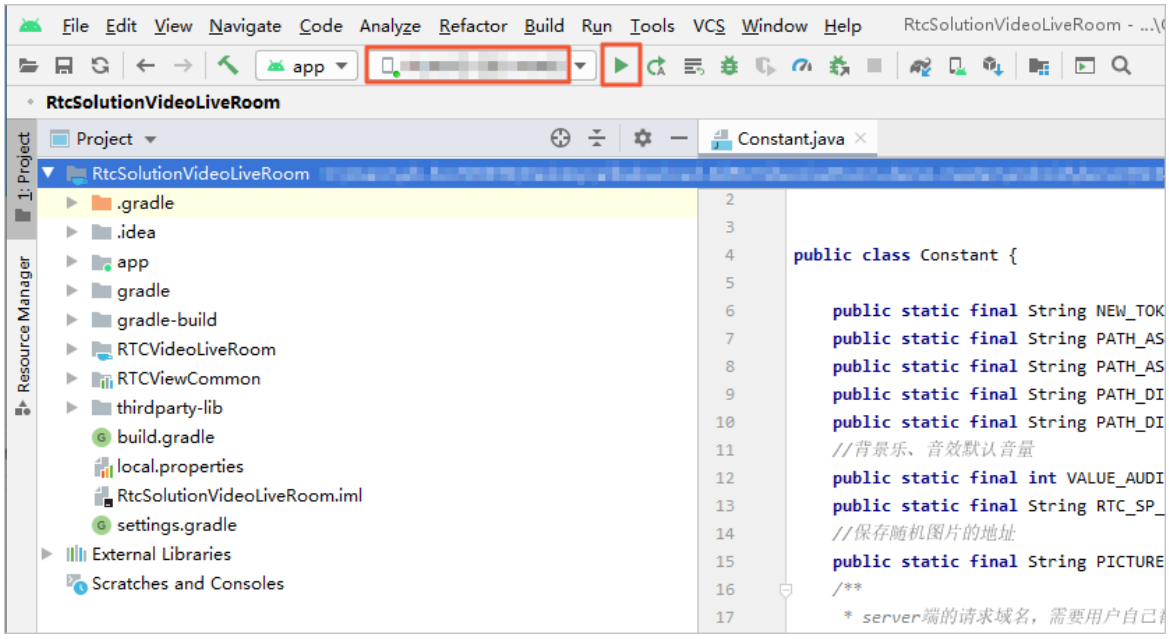
> **说明**
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `API_URL="http://<域名>/videoliveRoom"` 。

iv. 验证移动端和服务端。

分别在移动端和服务端浏览器中访问 `API_URL` 地址，如果显示如下图所示，表示移动端访问服务端正常。


3. 运行Demo。

将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。




> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

4. 开启视频互动直播。
   i. 将2台或2台以上移动设备安装Demo App。
   ii. 将设备都连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间标题开始直播。
   iv. 其余设备可在视频直播区看到该房间后，进入直播间观看。

## Demo目录结构说明

项目结构如下所示：

| 文件名 | 说明 |
| --- | --- |
| RTCVideoLiveRoom | 视频互动直播功能实现库。更多信息，请参见RTCVideoLiveRoom组件库目录说明。 |
| RTCSolutionCommon | 公共组建库。 |
| RTCViewCommon | 公共UI库。 |
| app | 程序入口。 |
| thirdparty-lib | 第三方库的引用。 |

RTCVideoLiveRoom组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| adapter | 列表控件adapter。 |

| 文件名 | 说明 |
| --- | --- |
| bean | 实体类。 |
| constant | 常量数据管理类，在此配置服务端请求域名和分享链接的域名。 |
| api | 网络请求。 |
| rtc | RTC。 |
| ui | 互动界面和登录界面。 |
| util | 工具类。 |
| view | 自定义view。 |

## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destorySharedInstance | 毁单例对象。 |
| joinRoom | 加入直播。 |
| leaveRoom | 退出直播。 |
| setAudioEffectVolume | 设置音效音量。 |
| setAudioAccompanyVolume | 设置伴奏音量。 |
| enterSeat | 上麦。 |
| leavelSeat | 下麦。 |
| enableEarBack | 设置耳返。 |
| startAudioAccompany | 播放伴奏。 |
| stopAudioAccompany | 停止伴奏。 |
| playAudioEffect | 播放音效。 |
| stopAudioEffect | 停止音效。 |
| setAudioEffectReverbMode | 设置音效场景（混音模式）。 |
| setDelegate | 设置监听。 |
| switchCamera | 翻转摄像头。 |

| API | 描述 |
| --- | --- |
| setBeautyEffect | 设置美颜效果。 |
| getWhiteLevel | 获取美白等级。 |
| getSmoothLevel | 获取磨皮等级。 |
| createRoom | 创建房间。 |
| destroyRoom | 销毁房间。 |
| kickout | 踢人。 |
| startCameraPreView | 本地预览。 |
| stopCameraPreView | 停止本地预览。 |
| startPlay | 播放远端画面。 |

回调接口

| API | 描述 |
| --- | --- |
| onEnterSeat | 用户上麦通知。 |
| onLeaveSeat | 用户下麦通知。 |
| onOccurError | 错误信息通知。 |
| onJoinChannelResult | 加入频道结果回调。 |
| onLeaveChannelResult | 离开频道结果回调。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onAudioPlayingStateChanged | 伴奏播放回调。 |
| onNetworkQualityChanged | 网络状态回调。 |
| onKickedOut | 被踢出房间的回调。 |

功能实现接口

- sharedInstance：获取单例对象。

  获取BaseRTCAudioLiveRoom的实例对象，初始化RTC SDK。

```
/**
 * 获取单例
 */
public static BaseRTCAudioLiveRoom sharedInstance() {
    return RTCAudioLiveRoomImpl.sharedInstance();
}
```

- destorySharedInstance：毁单例对象。

  销毁BaseRTCAudioLiveRoom的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
    /**
     * 销毁实例
     */
    public abstract void destorySharedInstance();
```

- joinRoom：加入房间。

  观众端调用，通过房间号等信息获取播放链接并播放旁路流。

```
    /**
     * 加入直播
     *
     * @param channelId 房间号
     * @param view  画面载体
     * @param userId uid
     * @param userName 昵称
     */
    public abstract void joinRoom(String channelId, String userId, String userName, Surfa
ceView view);
```

- leaveRoom：离开房间。

  观众端调用，离开房间。

```
     /**
     * 退出房间
     */
    public abstract void leaveRoom();
```

- setAudioEffectVolume：设置音效音量。

  同时设置播放和推流的音量。

```
    /**
     * 设置音效音量
     *
     * @param soundId 音效文件的sourceId
     * @param volume 音量 区间0-100
     *
     */
    public abstract void setAudioEffectVolume(int soundId, int volume);
```

- setAudioAccompanyVolume：设置伴奏音量。

  同时设置播放和推流的音量。

```
    /**
     * 设置伴奏音量
     * @param volume 音量 区间0-100
     */
    public abstract void setAudioAccompanyVolume(int volume);
```

- enterSeat：上麦。

切换互动角色。

```
    /**
     * 上麦
     */
    public abstract void enterSeat();
```

- leavelSeat：下麦。

  切换观众角色。

```
    /**
     * 下麦
     */
    public abstract void leaveSeat();
```

- enableEarBack：是否开启耳返。

```
    /**
     * 是否耳返
     *
     * @param enableEarBack 是否开启耳返 true为开启 false关闭
     * @return 0表示设置成功，反之失败
     */
    public abstract int enableEarBack(boolean enableEarBack);
```

- startAudioAccompany：播放伴奏。

```
    /**
     * 开始伴奏
     *
     * @param fileName      伴奏文件路径，支持本地文件和网络url
     * @param onlyLocalPlay 是否仅本地播放，true表示仅仅本地播放，false表示本地播放且推流到远端
     * @param replaceMic    是否替换mic的音频流，true表示伴奏音频流替换本地mic音频流，false表示伴奏音频流和mic音频流同时推
     * @param loopCycles    循环播放次数，-1表示一直循环
     */
    public abstract void startAudioAccompany(String fileName, boolean onlyLocalPlay, boolean replaceMic, int loopCycles);
```

- stopAudioAccompany：停止伴奏。

```
    /**
     * 停止伴奏
     */
    public abstract void stopAudioAccompany();
```

- playAudioEffect：播放音效。

```
    /**
     * 播放音效
     *
     * @param soundId  音效ID
     * @param filePath 音效文件路径，支持本地文件和网络url
     * @param cycles   循环播放次数。-1表示一直循环
     * @param publish  是否将音效音频流推到远端
     */
    public abstract void playAudioEffect(int soundId, String filePath, int cycles, boolea
n publish);
```

- stopAudioEffect：停止音效。

```
    /**
     * 停止音效
     *
     * @param soundId 音效ID
     */
    public abstract void stopAudioEffect(int soundId);
```

- setAudioEffectReverbMode：设置音效场景（混音模式）。

```
    /**
     * 设置音效场景
     * @param aliRtcAudioEffectReverbMode 音效场景枚举
     */
    public abstract void setAudioEffectReverbMode(AliRtcEngine.AliRtcAudioEffectReverbMod
e aliRtcAudioEffectReverbMode);
```

- setDelegate：设置监听。

```
    /**
     * 设置rtc监听
     *
     * @param audioLiveRoomDelegate 监听
     */
    public abstract void setDelegate(RTCVideoLiveRoomDelegate audioLiveRoomDelegate);
```

- switchCamera：翻转摄像头。

```
    /**
     * 翻转摄像头
     */
    public abstract void switchCamera();
```

- setBeautyEffect：设置美颜效果。

```
    /**
     * 设置美颜等级
     */
    public abstract void setBeautyEffect(float whiteLevel, float smoothLevel);
```

- getWhiteLevel：获取美白等级。

```
/**
 * 获取当前美白等级
 */
public abstract float getWhiteLevel();
```

- getSmoothLevel：获取磨皮等级。

```
/**
 * 获取当前磨皮等级
 */
public abstract float getSmoothLevel();
```

- createRoom：创建房间。

  主播调用，创建RTC频道并加入。

```
/**
 * 创建房间
 * @param channelId 房间号
 * @param userId uid
 * @param userName  用户名
 */
public abstract void createRoom(String channelId, String userId, String userName);
```

- destroyRoom：销毁房间。

  主播调用，销毁RTC频道并退出。

```
/**
 * 销毁房间
 */
public abstract void destroyRoom();
```

- kickout：踢人。

  主播调用，将连麦观众都踢下线。

```
/**
 * 踢人（主播才可以踢人）
 */
public abstract void kickOut();
```

- startCameraPreView：本地预览。

```
/**
 * 预览本地摄像头画面
 * @param viewGroup 播放画面的载体
 */
public abstract void startCameraPreView(ViewGroup viewGroup);
```

- stopCameraPreView：停止本地预览。

```
/**
 * 停止预览本地摄像头画面
 */
public abstract void stopCameraPreview();
```

- startPlay：播放远端画面。

```
/**
 * 播放远端流
 * @param uid 用户id
 * @param viewGroup 播放画面的载体
 */
public abstract void startPlay(String uid, ViewGroup viewGroup);
```

回调接口

- onEnterSeat：用户上麦通知。

```
/**
 * 上麦通知回调
 *
 * @param seatInfo 麦位信息
 */
void onEnterSeat(SeatInfo seatInfo);
```

- onLeaveSeat：用户下麦通知。

```
/**
 * 下麦通知回调
 *
 * @param seatInfo 麦位信息
 */
void onLeaveSeat(SeatInfo seatInfo);
```

- onOccurError：错误信息通知。

```
/**
 * sdk报错,需要销毁实例
 */
void onOccurError(int error);
```

- onJoinChannelResult：创建房间的回调。

```
/**
 * 创建房间的回调
 * @param result 0为成功 反之失败
 */
void onJoinChannelResult(int result);
```

- onLeaveChannelResult：退出房间的回调。

```
/**
 * 退出房间回调
 */
void onLeaveChannelResult(int result);
```

- onAudioPlayingStateChanged：伴奏播放回调。

  背景音乐播放状态的回调。

```
    /**
     * 播放状态更新回调
     *
     * @param audioPlayingStatus 当前播放状态
     */
    void onAudioPlayingStateChanged(AliRtcEngine.AliRtcAudioPlayingStateCode audioPlaying
Status);
```

- onRoomDestroy：房间被销毁回调。

```
    /**
     * 房间被销毁回调
     */
    void onRoomDestroy();
```

- onNetworkQualityChanged：网络质量变化时回调。

```
    /**
     * 网络状态回调
     *
     * @param aliRtcNetworkQuality1 下行网络质量
     * @param aliRtcNetworkQuality  上行网络质量
     * @param s                     String  用户ID
     */
    void onNetworkQualityChanged(String s, AliRtcEngine.AliRtcNetworkQuality aliRtcNetwor
kQuality, AliRtcEngine.AliRtcNetworkQuality aliRtcNetworkQuality1);
```

- onKickedOut：被踢出房间的回调。

```
    /**
     * 被踢出房间
     */
    void onKickOuted();
```

# 2.5.4. iOS集成

通过阅读本文，您可以了解视频互动直播iOS端的集成操作。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- 服务端已集成并开启。具体操作，请参见服务端集成。
- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

## 操作步骤

1. 下载并解压Demo，更多信息，请参见Demo源码下载。

> 说明
>
> - Demo源码中已经集成AliRTC SDK。
> - 源码压缩文件内分为Server端、Android端、iOS端三个文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

2. 配置Demo工程。
   i. 打开*iOS/RTCSolution/RTCCommon/AppConfig.h*文件。
   ii. 修改文件中的 `kBaseUrl` 和 `kProject_VideoRoom` 值。

   例如本地服务端IP地址为192.0.2.1，则 `kBaseUrl` 的值为 `http://192.0.2.1:8080/` 。本地服务端IP地址查询，请参见查询IP地址。

   修改 `kProject_VideoRoom` 的值为 `videoliveRoom` 。

```
//
//  AppConfig.h
//  LectureHall
//
//  Created by Aliyun on 2020/5/22.
//  Copyright © 2020 alibaba. All rights reserved.
//

#ifndef AppConfig_h
#define AppConfig_h

//====================================  URL 常量 start ================================================================

#define kBaseUrl                                @"http://[redacted]/"

#define kProject_VideoRoom                      @"videoliveRoom"

#define kBaseUrl_VideoRoom                      [NSString stringWithFormat:@"%@%@/",kBaseUrl,kProject_VideoRoom]


//====================================  URL 常量 end ==================================================================

#endif /* AppConfig_h */
```

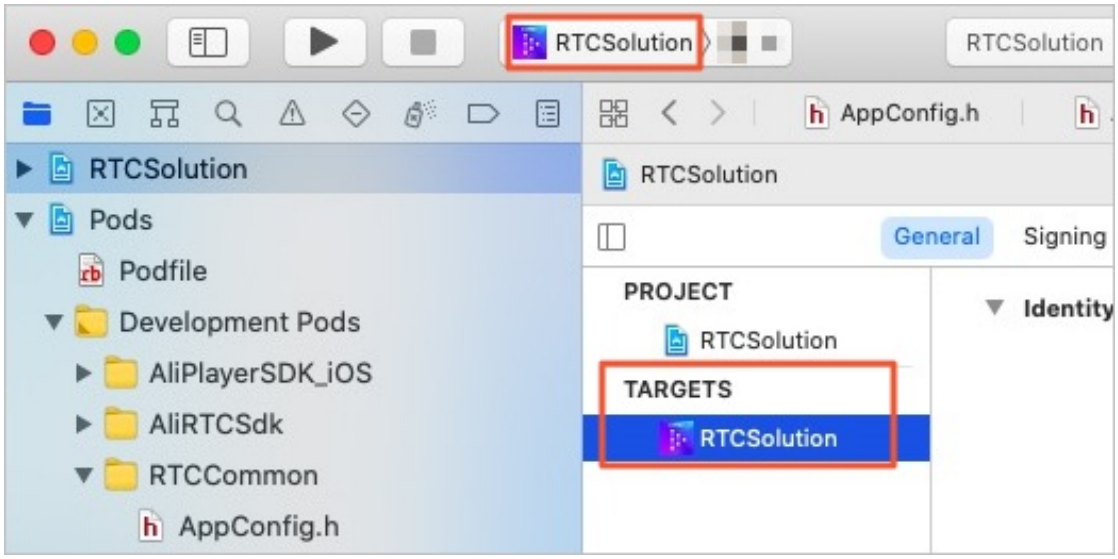
> 说明
>
> - 服务端IP地址禁止使用127.0.0.1。
> - 移动端和服务端处于同一局域网中。
> - 如果需要部署到正式环境，请绑定域名即 `AppServer` 的值为 `http://<域名>/1v1-audio` 。

   iii. 验证移动端和服务端。

   分别在移动端和服务端浏览器中访问 `http://<服务器IP> :8080/videoliveRoom` 地址，如果显示如下图所示，表示移动端访问服务端正常。

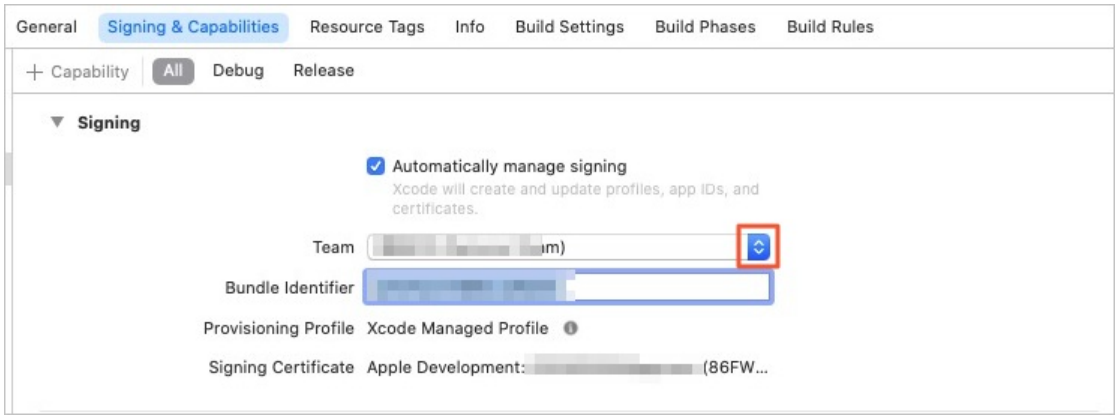

3. 运行Demo。
   i. 使用Xcode打开*iOS/RTCSolution*目录下的*RTCSolution.xcworkspace*文件。

ii. 选择运行的Target为RTCSolution，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。



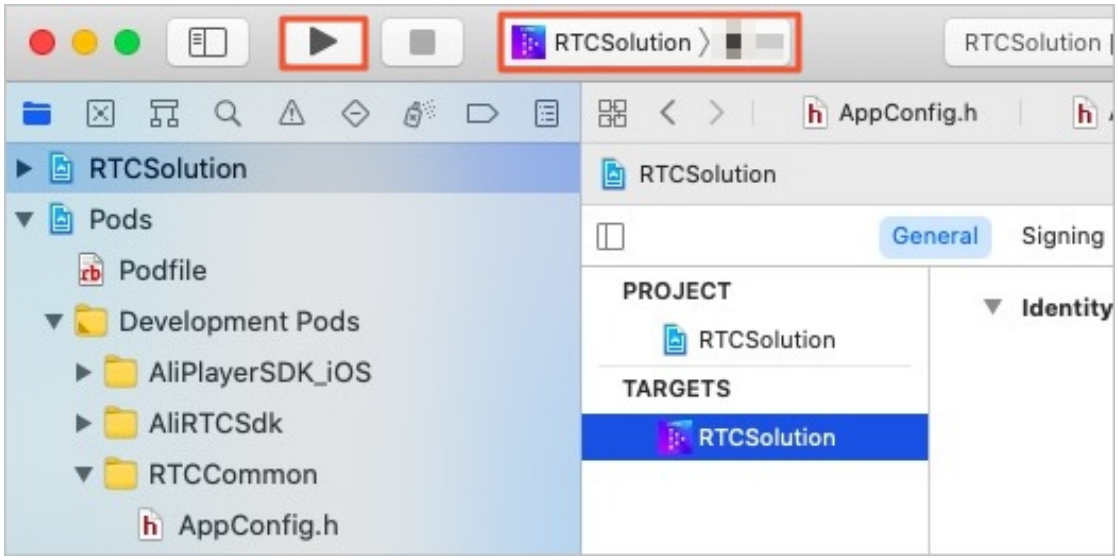

iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。


iv. 单击**Signing & Capabilities**页签，选中**Automatically manage signing**，然后单击**Team**下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择**Add an Account...**，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击▶，编译并运行。如果编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。




4. 开启视频互动直播。
   i. 将2台或2台以上移动设备安装Demo App。
   ii. 将设备都连接到同一局域网下，保证可以连接到Server端。
   iii. 在第一台设备上输入任意房间标题开始直播。
   iv. 其余设备可在视频直播区看到该房间后，进入直播间观看。

## Demo目录结构说明

RTC把开发的业务代码封装到RTCVideoLiveRoom库中，因此只需在Podfile中指定RTCVideoLiveRoom库的路径，RTCVideoLiveRoom就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
pod 'RTCVideoLiveRoom', :path => 'RTCVideoLiveRoom'
   ##   pod 'RTCVideoLiveRoom'   说明项目依赖RTCVideoLiveRoom库
   ##   path => 'RTCVideoLiveRoom'   指明RTCVideoLiveRoom库的位置（podfile文件路径）
```

RTCVideoLiveRoom组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| RTCVideoLiveRoom.podspec | 组件的描述文件。 |
| RTCVideoLiveRoom.bundle | 存放资源的bundle。 |
| RTC | RTC相关功能的封装（建议直接复用）。 |
| UI | UI相关文件。 |

## API说明

功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destorySharedInstance | 毁单例对象。 |
| joinRoom | 加入直播。 |
| leaveRoom | 退出直播。 |
| setAudioEffectVolume | 设置音效音量。 |
| setAudioAccompanyVolume | 设置伴奏音量。 |
| enterSeat | 上麦。 |
| leavelSeat | 下麦。 |
| enableEarBack | 设置耳返。 |
| startAudioAccompany | 播放伴奏。 |
| stopAudioAccompany | 停止伴奏。 |
| playAudioEffect | 播放音效。 |
| stopAudioEffect | 停止音效。 |
| setAudioEffectReverbMode | 设置音效场景（混音模式）。 |
| setDelegate | 设置监听。 |
| switchCamera | 翻转摄像头。 |
| setWhiteLevel | 设置美白效果。 |
| setSmoothLevel | 设置磨皮效果。 |
| getWhiteLevel | 获取美白等级。 |
| getSmoothLevel | 获取磨皮等级。 |
| createRoom | 创建房间。 |
| destroyRoom | 销毁房间。 |
| kickout | 踢人。 |
| startCameraPreView | 本地预览。 |
| stopCameraPreView | 停止本地预览。 |
| startPlay | 播放远端画面。 |

回调接口

| API | 描述 |
| --- | --- |
| onEnterSeat | 用户上麦通知。 |
| onLeaveSeat | 用户下麦通知。 |
| onOccurError | 错误信息通知。 |
| onJoinChannelResult | 加入频道结果回调。 |
| onLeaveChannelResult | 离开频道结果回调。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onAudioPlayingStateChanged | 伴奏播放回调。 |
| onNetworkQualityChanged | 网络状态回调。 |
| onKickedOut | 被踢出房间的回调。 |

功能实现接口

- 获取单例对象

  接口名称：sharedInstance。

  获取BaseRTCAudioLiveRoom的实例对象，初始化RTC SDK。

```
/// @brief 获取单例
/// @return RTCAudioliveRoomManager 单例对象
+ (RTCVideoliveRoom *) sharedInstance;
```

- 毁单例对象

  接口名称：destorySharedInstance。

  销毁BaseRTCAudioLiveRoom的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
/// 销毁RTCSDK
- (void)destroySharedInstance;
```

- 加入房间

  接口名称：joinRoom。

  观众端调用，通过房间号等信息获取播放链接并播放旁路流。

```
/// 加入房间
/// @param channelId  频道
/// @param userId 观众id
/// @param userName 观众昵称
/// @param preview 预览view
/// @param handler 回调
- (void)joinRoom:(NSString *)channelId
          userId:(NSString *)userId
        userName:(NSString *)userName
         preview:(UIView *)preview
        complete:(void(^)(NSInteger result))handler;
```

- 离开房间

  接口名称：leaveRoom。

  观众端调用，离开房间。

```
/// 观众调用 退出直播
/// @param handler 回调
- (void)leaveRoom:(void(^)(NSInteger result))handler;
```

- 设置音效音量

  接口名称：setAudioEffectVolume。

  同时设置播放和推流的音量。

```
/// 设置音效的音量
/// @param soundId 音效id
/// @param volume 音量 0~100
- (int)setAudioEffectVolumeWithSoundId:(NSInteger)soundId
                                volume:(NSInteger)volume;
```

- 设置伴奏音量

  接口名称：setAudioAccompanyVolume。

  同时设置播放和推流的音量。

```
/// 设置背景音乐音量
/// @param volume 音量 0~100
- (int)setAudioAccompanyVolume:(NSInteger)volume;
```

- 上麦

  接口名称：enterSeat。

  切换互动角色。

```
/// 上麦
/// @param handler 回调
- (void)enterSeat:(void(^)(NSInteger result))handler;
```

- 下麦

  接口名称：leavelSeat。

  切换观众角色。

```
/// 下麦
/// @param handler 回调
- (void)leaveSeat:(void(^)(NSInteger result))handler;
```

- 是否开启耳返

  接口名称：enableEarBack。

```
/// 是否开启耳返
/// @param enable YES/NO
- (int)enableEarBack:(BOOL)enable;
```

- 播放伴奏

  接口名称：startAudioAccompany。

```
/// 播放背景音乐
/// @param filePath 文件路径
/// @param publish 是否推送远端
- (int)startAudioAccompanyWithFile:(NSString *)filePath
                           publish:(BOOL)publish;
```

- 停止伴奏

  接口名称：stopAudioAccompany。

```
/// 停止播放背景音乐
- (int)stopAudioAccompany;
```

- 播放音效

  接口名称：playAudioEffect。

```
/// 播放音效
/// @param soundId 音效id
/// @param filePath 资源路径
/// @param publish 是否推送远端
- (int)playEffectSoundtWithSoundId:(NSInteger)soundId
                          filePath:(NSString *)filePath
                           publish:(BOOL)publish;
```

- 停止音效

  接口名称：stopAudioEffect。

```
/// 停止播放音效
/// @param soundId 音效id
- (int)stopAudioEffectWithSoundId:(NSInteger)soundId;
```

- 设置音效场景（混音模式）

  接口名称：setAudioEffectReverbMode。

```
/// 设置音效混响模式
/// @param mode 混响模式
- (int)setAudioEffectReverbMode:(AliRtcAudioEffectReverbMode)mode;
```

- 设置监听

接口名称：setDelegate。

```
 - (void)setDelegate:(id<RTCVideoliveRoomDelegate> _Nullable)delegate;
```

- 翻转摄像头

  接口名称：switchCamera。

```
/// 切换摄像头
- (void)switchCamera;
```

- 设置美白效果

  接口名称：setWhiteLevel。

```
- (void)setWhiteLevel:(NSInteger)whiteLevel;
```

- 设置磨皮效果

  接口名称：setSmoothLevel。

```
- (void)setSmoothLevel:(NSInteger)smoothLevel;
```

- 获取美白等级

  接口名称：getWhiteLevel。

```
-(NSInteger)whiteLevel;
```

- 获取磨皮等级

  接口名称：getSmoothLevel。

```
- (NSInteger)smoothLevel;
```

- 创建房间

  接口名称：createRoom。

  主播调用，创建RTC频道并加入。

```
/// 加入频道
/// @param channelId    频道名称
/// @param userName    任意用于显示的用户名称。不是User ID
/// @param userId    角色
/// @param handler    回调
- (void)createRoom:(NSString *)channelId
          userName:(NSString *)userName
            userId:(NSString *)userId
          complete:(void(^)(AliRtcAuthInfo *authInfo,
                            NSInteger errorCode))handler;
```

- 销毁房间

  接口名称：destroyRoom。

  主播调用，销毁RTC频道并退出。

```
/// 主播调用 销毁房间
/// @param handler 回调
- (void)destroyRoom:(void(^)(NSInteger result))handler;
```

- 踢人

  接口名称：kickout。

  主播调用，将连麦观众都踢下线。

```
/// 踢出房间的其他用户
/// @param handler 回调
- (void)kickout:(void(^)(NSInteger result))handler;
```

- 本地预览

  接口名称：startCameraPreView。

```
/// 开启本地预览
/// @param preview 预览View
- (void)startCameraPreView:(UIView *)preview;
```

- 停止本地预览

  接口名称：stopCameraPreView。

```
/// 停止本地预览
- (void)stopCameraPreview;
```

- 播放远端画面

  接口名称：startPlay。

```
/// 播放远端画面
/// @param preview     预览的view
/// @param userId  对方的userId
- (void)startPlay:(UIView *)preview
           userId:(NSString *)userId;
```

回调接口

- 用户上麦通知

  回调名称：onEnterSeat。

```
/// 远端用户上麦通知
/// @param userId 麦序
- (void)onEnterSeat:(NSString *)userId;
```

- 用户下麦通知

  回调名称：onLeaveSeat

```
/// 远端用户下线通知
/// @param userId 麦序
- (void)onLeaveSeat:(NSString *)userId;
```

- 错误信息通知

  回调名称：onOccurError。

```
- (void)onOccurError:(int)error;
```

- 创建房间的回调

  回调名称：onJoinChannelResult。

```
- (void)onJoinChannelResult:(int)result
                   authInfo:(AliRtcAuthInfo *)authInfo;
```

- 退出房间的回调

  回调名称：onLeaveChannelResult。

```
- (void)onLeaveChannelResult:(int)result;
```

- 伴奏播放回调

  回调名称：onAudioPlayingStateChanged。

  背景音乐播放状态的回调。

```
- (void)onAudioPlayingStateChanged:(AliRtcAudioPlayingStateCode)playState
                         errorCode:(AliRtcAudioPlayingErrorCode)errorCode;
```

- 房间被销毁回调

  回调名称：onRoomDestroy。

```
/// 房间被销毁通知
- (void)onRoomdestroy;
```

- 网络质量变化时回调

  回调名称：onNetworkQualityChanged。

```
- (void)onNetworkQualityChanged:(NSString *)uid
               upNetworkQuality:(AliRtcNetworkQuality)upQuality
             downNetworkQuality:(AliRtcNetworkQuality)downQuality;
```

- 被踢出房间的回调

  回调名称：onKickedOut。

```
/// 被踢出房间
- (void)onkickedOut;
```

# 2.6. 超级小班课

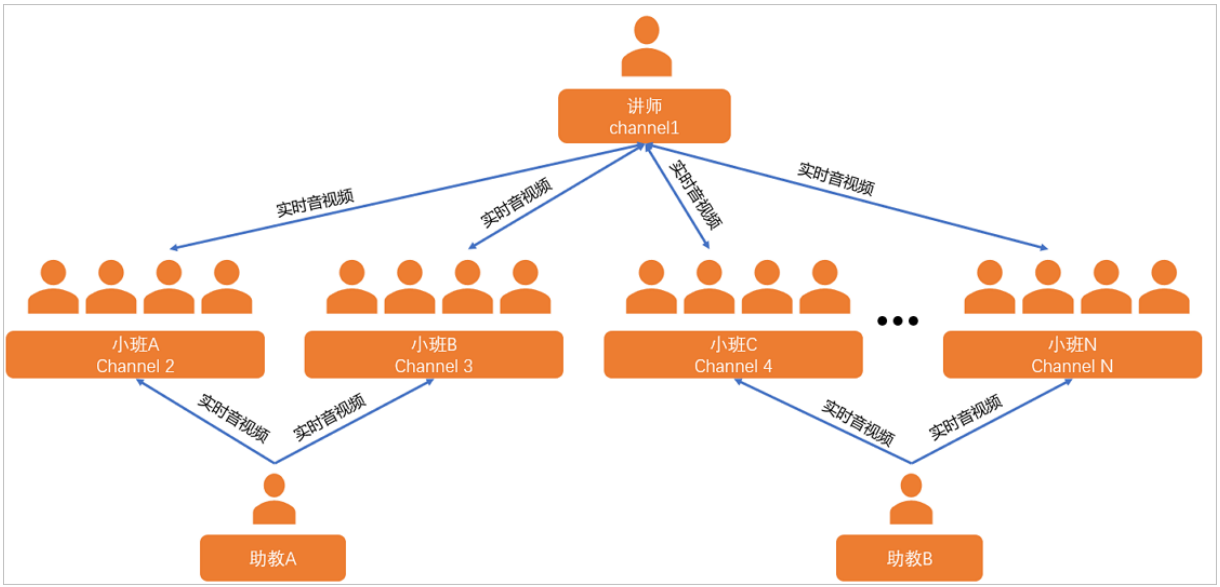
## 2.6.1. 简介

通过阅读本文，您可以快速了解超级小班课的基本信息及实现方法。

### 使用场景

超级小班课是将千名学生以小组形式分成若干小班（推荐4~6名），同时由一名主讲名师和多名助教进行辅导。所有学生均可以实时观看主讲名师授课画面，并可以与名师进行连麦互动。多名助教实时关注小班内学生动态，维护小班课堂秩序，并可连麦小班内学生进行助教辅导。该场景不仅可以让优秀的名师辅导更多的学生，更可以保证学生之间的互动性，让学生的学习效率大大增加。超级小班课适用于K12、少儿语培等教育场景。

## 架构方案

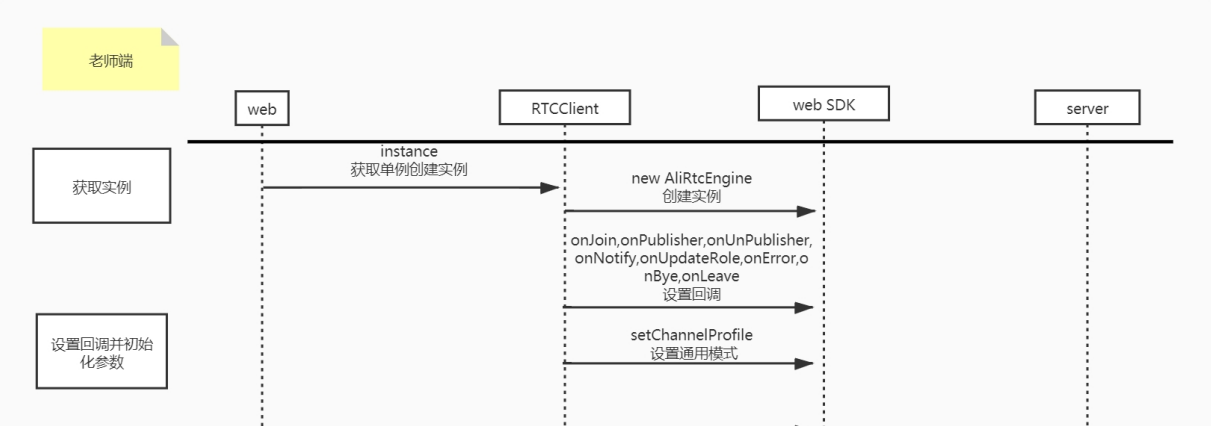


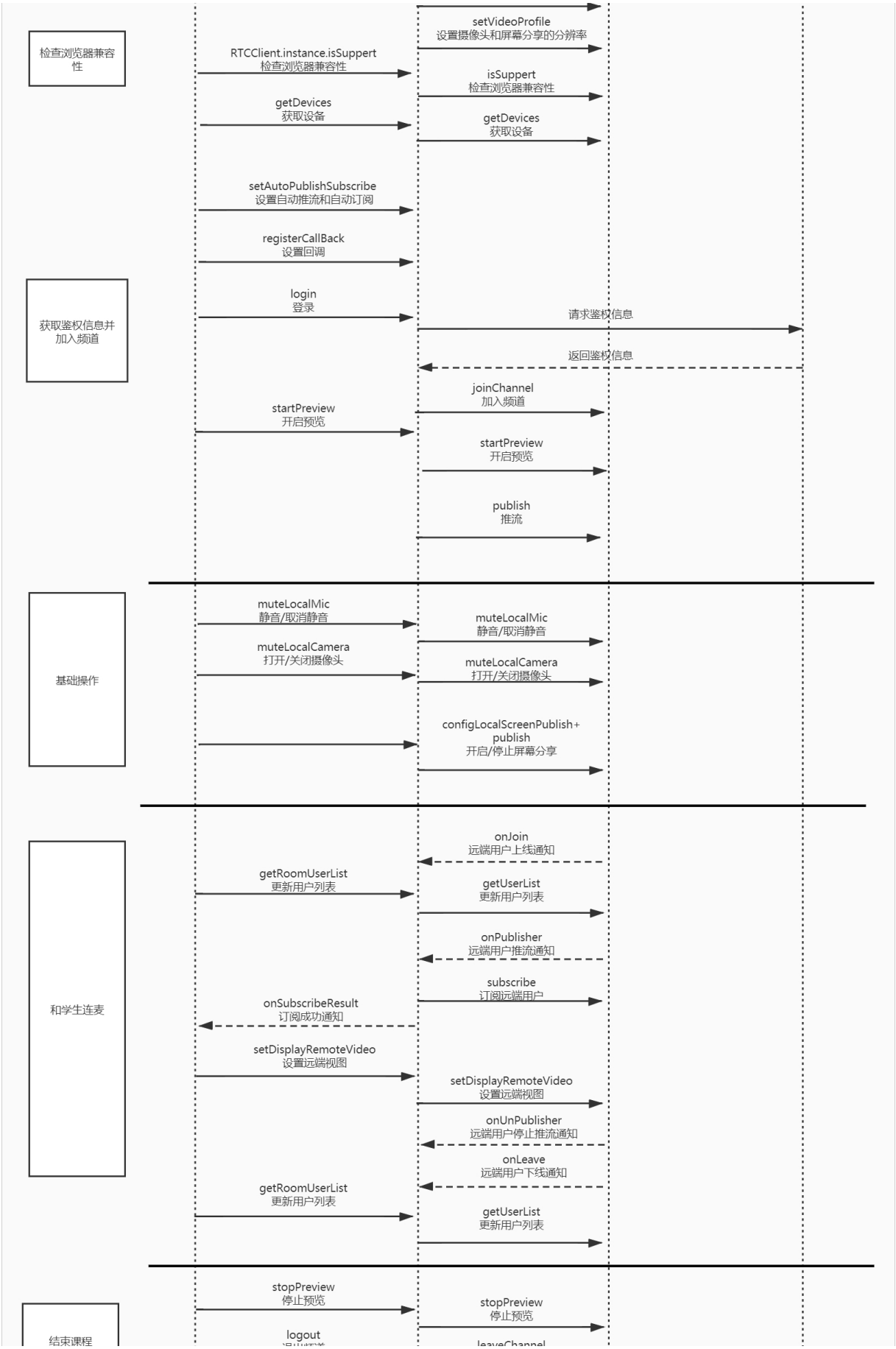

## 主要功能

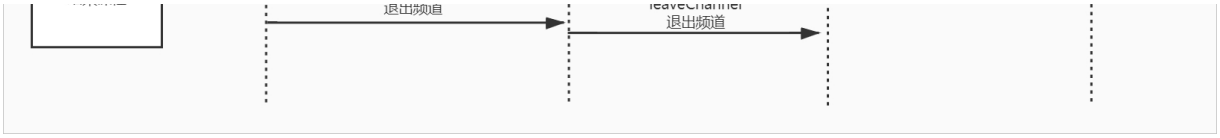
| 功能 | 描述 |
| --- | --- |
| 实时视频连麦 | 所有小班内的学生均可以接收到老师的音视频，学生也可以与老师进行音视频连麦，其它小班内的学生可以正常接收连麦画面和音频。 |
| 加入多频道 | 学生同时加入小班频道和公共教师频道，助教可订阅三个频道并接收学生音视频。 |
| 屏幕共享 | 教师可将电脑屏幕画面共享给其它学生。 |

## 实现方法

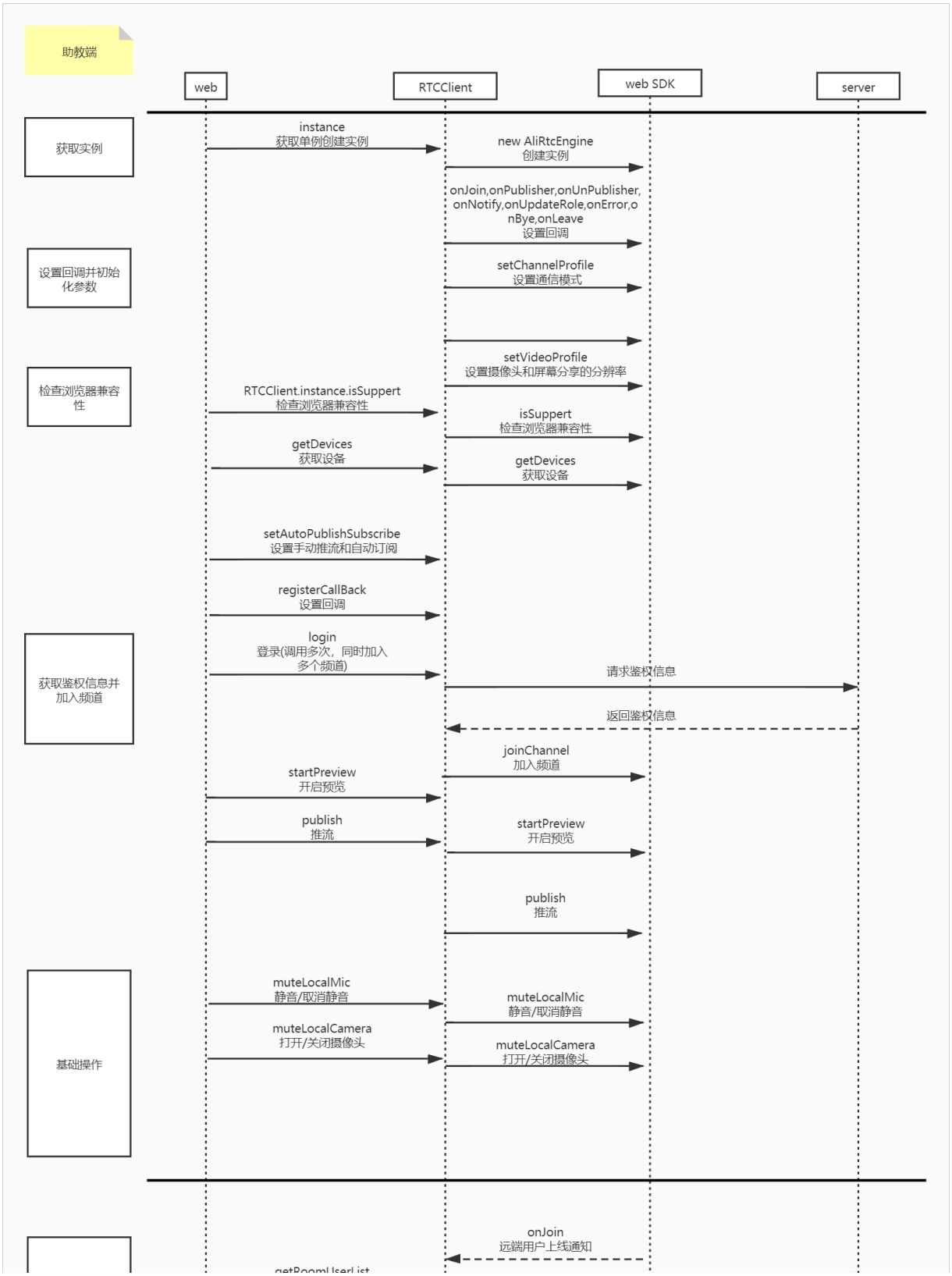
教师端（Web）

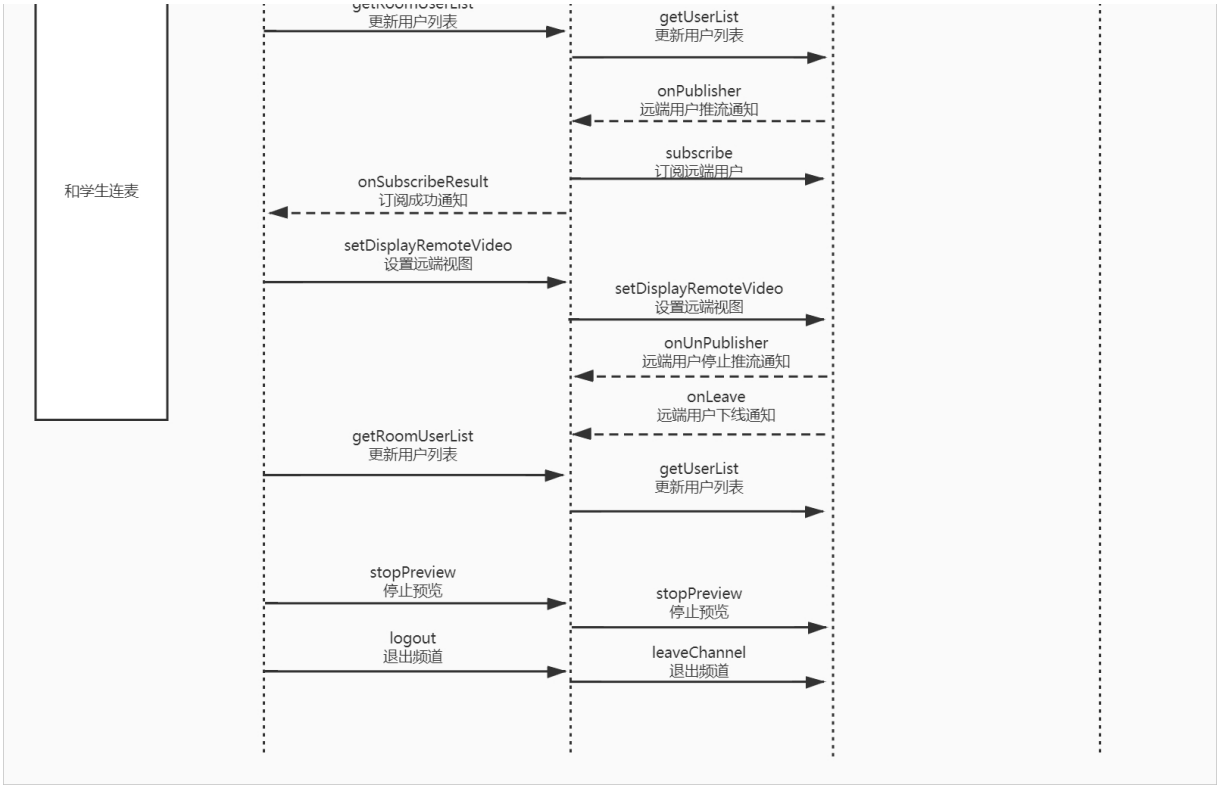



检查浏览器兼容性
setVideoProfile
设置摄像头和屏幕分享的分辨率
RTCClient.instance.isSuppert
检查浏览器兼容性
isSuppert
检查浏览器兼容性
getDevices
获取设备
getDevices
获取设备
setAutoPublishSubscribe
设置自动推流和自动订阅
registerCallBack
设置回调
获取鉴权信息并加入频道
login
登录
请求鉴权信息
返回鉴权信息
joinChannel
加入频道
startPreview
开启预览
startPreview
开启预览
publish
推流
基础操作
muteLocalMic
静音/取消静音
muteLocalMic
静音/取消静音
muteLocalCamera
打开/关闭摄像头
muteLocalCamera
打开/关闭摄像头
configLocalScreenPublish+
publish
开启/停止屏幕分享
和学生连麦
onJoin
远端用户上线通知
getRoomUserList
更新用户列表
getUserList
更新用户列表
onPublisher
远端用户推流通知
subscribe
订阅远端用户
onSubscribeResult
订阅成功通知
setDisplayRemoteVideo
设置远端视图
setDisplayRemoteVideo
设置远端视图
onUnPublisher
远端用户停止推流通知
onLeave
远端用户下线通知
getRoomUserList
更新用户列表
getUserList
更新用户列表
结束课程
stopPreview
停止预览
stopPreview
停止预览
logout
leaveChannel

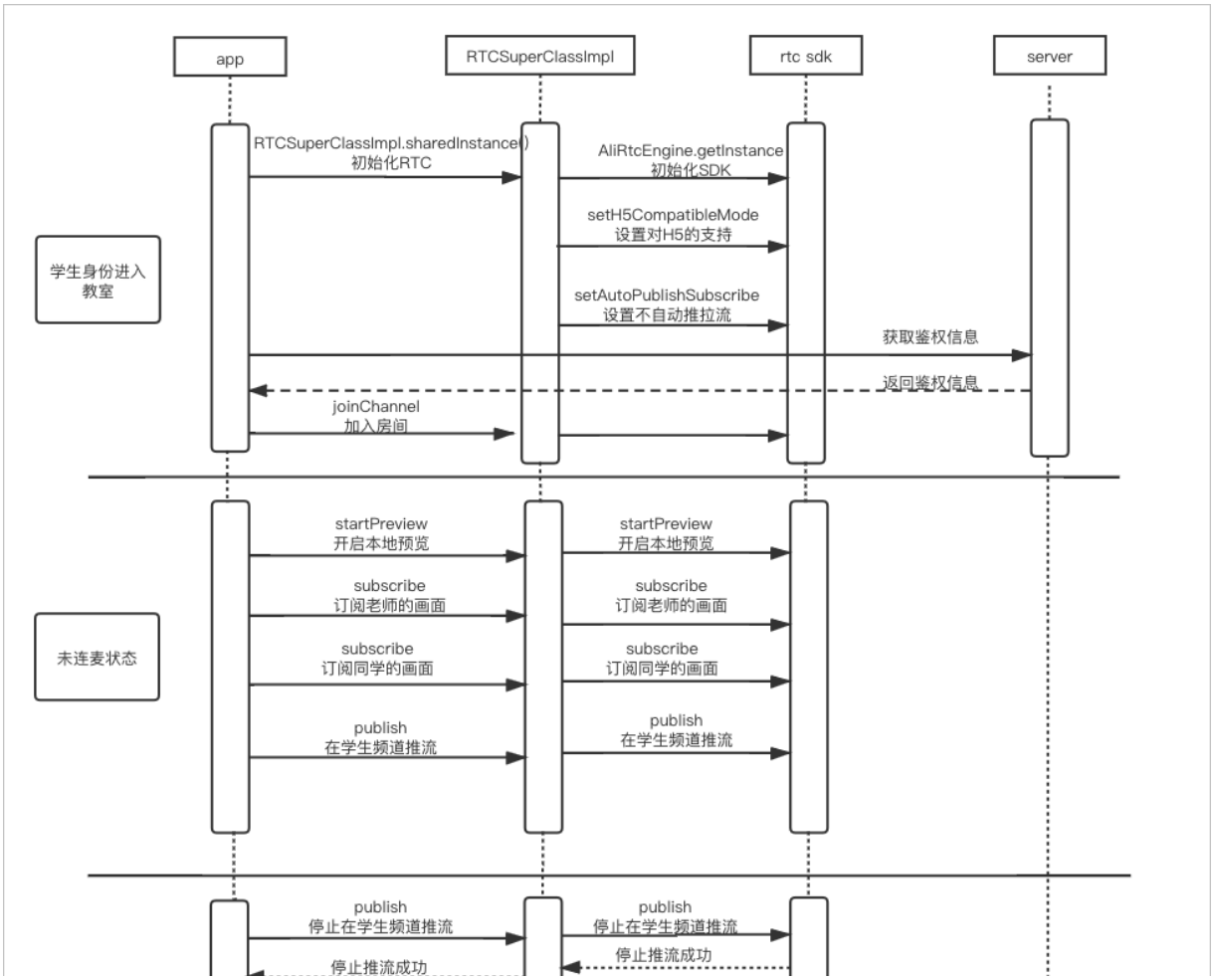

退出频道
退出频道

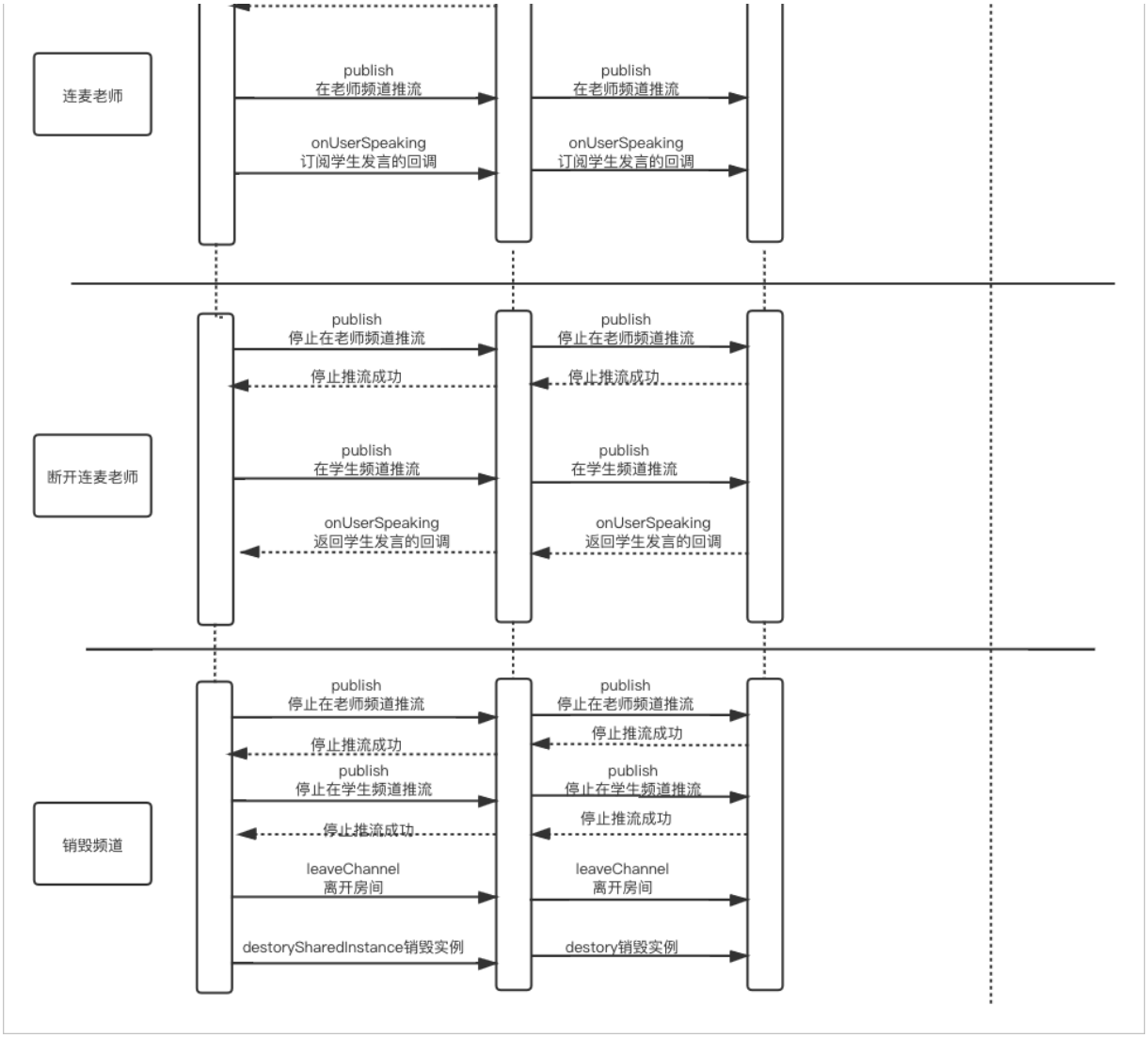
助教端（Web）


助教端
web
RTCClient
web SDK
server
获取实例
instance
获取单例创建实例
new AliRtcEngine
创建实例
onJoin,onPublisher,onUnPublisher,
onNotify,onUpdateRole,onError,o
nBye,onLeave
设置回调
设置回调并初始
化参数
setChannelProfile
设置通信模式
setVideoProfile
设置摄像头和屏幕分享的分辨率
检查浏览器兼容
性
RTCClient.instance.isSuppert
检查浏览器兼容性
isSuppert
检查浏览器兼容性
getDevices
获取设备
getDevices
获取设备
setAutoPublishSubscribe
设置手动推流和自动订阅
registerCallBack
设置回调
获取鉴权信息并
加入频道
login
登录(调用多次，同时加入
多个频道)
请求鉴权信息
返回鉴权信息
joinChannel
加入频道
startPreview
开启预览
publish
推流
startPreview
开启预览
publish
推流
基础操作
muteLocalMic
静音/取消静音
muteLocalMic
静音/取消静音
muteLocalCamera
打开/关闭摄像头
muteLocalCamera
打开/关闭摄像头
onJoin
远端用户上线通知


和学生连麦
更新用户列表
getUserList
更新用户列表
onPublisher
远端用户推流通知
subscribe
订阅远端用户
onSubscribeResult
订阅成功通知
setDisplayRemoteVideo
设置远端视图
setDisplayRemoteVideo
设置远端视图
onUnPublisher
远端用户停止推流通知
onLeave
远端用户下线通知
getRoomUserList
更新用户列表
getUserList
更新用户列表
stopPreview
停止预览
stopPreview
停止预览
logout
退出频道
leaveChannel
退出频道

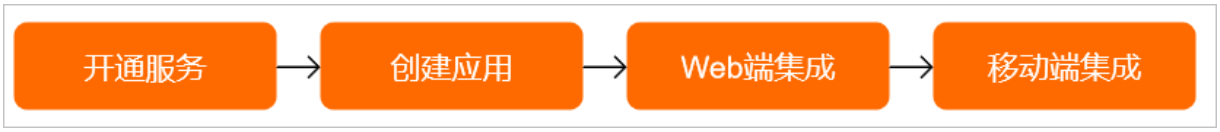
学生端（Android）


app
RTCSuperClassImpl
rtc sdk
server
学生身份进入教室
RTCSuperClassImpl.sharedInstance()
初始化RTC
AliRtcEngine.getInstance
初始化SDK
setH5CompatibleMode
设置对H5的支持
setAutoPublishSubscribe
设置不自动推拉流
获取鉴权信息
返回鉴权信息
joinChannel
加入房间
未连麦状态
startPreview
开启本地预览
startPreview
开启本地预览
subscribe
订阅老师的画面
subscribe
订阅老师的画面
subscribe
订阅同学的画面
subscribe
订阅同学的画面
publish
在学生频道推流
publish
在学生频道推流
publish
停止在学生频道推流
publish
停止在学生频道推流
停止推流成功
停止推流成功




## 实现流程

实现流程如下图所示：




| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 1 | 开通音视频通信服务 | 进行Web端集成之前，您必须开通音视频通信服务。音视频通信默认采取后付费的模式，您可以在阿里云账户充值任意金额进行测试。 |
| 2 | 创建应用 | 根据实际情况使用现有的应用或创建新的应用，同时获取对应的AppID和AppKey。 |

| 步骤 | 操作 | 描述 |
| --- | --- | --- |
| 3 | Web集成 | 目前教师端仅支持Web端，您可以通过源码快速搭建Web端超级小班课。 |
| 4 | 移动端集成：<br>• iOS集成<br>• Android集成 | 您可以通过源码快速搭建移动端超级小班课。 |

### Demo体验

您可以通过钉钉扫描以下二维码，下载安装超级小班课Demo体验。

## 2.6.2. Web集成

通过阅读本文，您可以了解超级小班课Web端的集成操作。

### 环境要求

Web端具体环境要求，更多信息，请参见使用限制。

### 前提条件

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 您已经开通音视频通信服务。具体操作，请参见开通服务。
- 环境中已安装Node.js 6.0或以上版本。具体操作，请参见安装Node.js。
- 如果设备为Mac，需要为浏览器打开屏幕录制权限。

### 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。
   iv. 单击**操作**列的**查询AppKey**，获取AppKey。

> **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

> **说明**
> - 源码压缩文件内分为Web端、Android端、iOS端三个文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

3. 配置Demo工程。

   根据步骤1中获取的AppID和AppKey修改*web/src/core/data/config.js*文件中`appID`和`appKey`的值。

```
export default{
    appId:"",
    appKey:""
}
```

> **说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

4. 运行Demo。
   i. 打开后台终端并进入到*web*文件夹下*package.json*文件所在目录。
   ii. 安装项目依赖。

   ```
   npm install
   ```

   iii. 集成RTC Web SDK 。

   ```
   npm install aliyun-webrtc-sdk -S
   ```

iv. 运行Demo。

**npm run serve**

运行成功后，浏览器会默认打开Demo应用示例。如果没有打开，请在浏览器中访问：

- Windows：*https://localhost:888*。
- Mac：*https://localhost:1024*。

如果访问页面出现警告：**您的连接不是私密连接**。单击**高级**，继续访问。


v. （可选）压缩静态资源文件，打包发布。

**npm run build**

## Demo源码解析

- 目录结构说明

```
├── dist                                #打包文件
├── public                              #静态资源
├── src                                 #项目文件目录
│   ├── assets                          #静态资源
│   ├── components                      #公共组件
│   ├── core                            #js文件
│   │   ├── data
│   │   │   ├── config.js               #相关配置参数
│   │   ├── util
│   │   │   ├── utils.js                #一些公共方法
│   │   ├── rtc-engine.js               #单例
│   │   ├── rtc-clinet.js               #RTC实例文件
│   ├── plugins
│   ├── router                          #路由
│   ├── views                           #页面
│   │   ├── login
│   │   │   ├── login.vue               #登录页面
│   │   ├── student
│   │   │   ├── student.vue             #学生页面
│   │   ├── assistant
│   │   │   ├── assistant.vue           #助教页面
│   │   ├── teacher
│   │   │   ├── teacher.vue             #教师页面
│   ├── vuex
│   ├── App.vue                         #根组件
│   ├── main.js                         #入口文件
├── vue.config.js                       #vue配置文件
```

- 主要功能说明
  - 检查浏览器是否支持。

    ```
    RtcEngine.instance.isSupport().then(re => {}).catch(err=>{});
    ```

  - 获取设备信息。

    ```
    RtcEngine.instance.getDevices().then(re => {})
    ```

  - 指定摄像头。

    ```
    RtcEngine.instance.currentCamera(deviceId)
    ```

  - 指定麦克风。

    ```
    RtcEngine.instance.currentAudioCapture(deviceId)
    ```

  - 开启预览。

    ```
    /**
     * 预览
     * @parame {HtmlVideoElement} video 播放预览画面的video标签
     */
    RtcEngine.instance.startPreview(video).then(re=>{})
    ```

- 停止预览。

```
RtcEngine.instance.stopPreview(video).then(re=>{})
```

- 设置是否自动推流自动订阅。

  需要在加入频道之前设置，此接口是针对频道设置的。

```
/**
   * 设置是否自动推流和自动订阅 默认自动推流自动订阅
   * @param {*} channel 频道
   * @param { boolean } autoPub true表示自动推流
   * @param { boolean } autoSub true表示自动订阅
   */
RtcEngine.instance.setAutoPublishSubscribe(channel, autoPub, autoSub)
```

- 注册回调。

  需要在加入频道之前设置，此接口是针对频道设置的。

```
/**
   * 注册回调
   * @param {*} channel 频道
   * @param {*} callback
   */
RtcEngine.instance.registerCallBack(channel, callback)
```

- 加入频道。

```
/**
   * 加入房间
   * @param {*} channel 频道
   * @param {*} userName
   */
RtcEngine.instance.login(channel, userName).then(re=>{})
```

- 开始推流。

```
/**
   * 开始推流
   * @param {*} channel 频道
   */
RtcEngine.instance.startPublish(channel).then(re=>{})
```

- 停止推流。

```
/**
   * 停止推流
   * @param {*} channel 频道
   */
RtcEngine.instance.stopPublish(channel).then(re=>{})
```

- 设置是否停止发布本地音频。

```
 /**
   * 设置是否停止发布本地音频
   * @param {*} channel  频道
   * @param {*} enable
   */
RtcEngine.instance.muteLocalMic(channel, enable)
```

- 设置是否停止发布本地视频。

```
 /**
   * 设置是否停止发布本地视频
   * @param {*} channel  频道
   * @param {*} enable
   */
RtcEngine.instance.muteLocalCamera(channel, enable)
```

- 切换开启和停止屏幕流。

```
/**
   * 切换开启和停止屏幕流
   * @param {*} channel  频道
   * @param {*} enable
   */
RtcEngine.instance.switchPublishScreen(channel, enable)
```

- 订阅音视频。

```
/**
   * 设置远端渲染 默认订阅音频 + 视频（小流）
   * @param {*} channel  频道
   * @param {*} userId
   */
RtcEngine.instance.subscribe(channel, userId).then(re=>{})
```

- 取消订阅。

```
/**
   * 取消订阅
   * @param {*} channel  频道
   * @param {*} userId
   */
RtcEngine.instance.unSubScribe(channel, userId).then(re=>{})
```

- 设置远端渲染。

```
/**
   * 设置远端渲染
   * @param {*} channel  频道
   * @param {*} userId
   * @param {*} video
   * @param {*} streamType
   */
RtcEngine.instance.setDisplayRemoteVideo(channel, userId, video, streamType)
```

- 获取频道用户列表。

```
  /**
   * 获取判断用户列表 频道
   * @param {*} channel
   * @return { array | boolean }
   */
RtcEngine.instance.getUserList(channel)
```

- 获取用户信息。

```
  /**
   * 获取判断用户列表
   * @param {*} channel 频道
   * @return { array | boolean }
   */
RtcEngine.instance.getUserInfo(channel, userId)
```

- 离开频道。

```
 /**
   * 离开频道
   */
RtcEngine.instance.logout().then(re=>{})
```

# 2.6.3. Android集成

通过阅读本文，您可以了解Android端超级小班课的集成操作。

## 环境要求

Android端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- Web端服务已集成并开启。具体操作，请参见Web集成。
- 环境中已安装Android Studio 3.0或以上版本。更多信息，请参见Android Studio。

## 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。
   iv. 单击**操作**列的**查询AppKey**，获取AppKey。

   > **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

> **说明**
>
> - 源码压缩文件内分为Android、iOS、Electron、Server四端文件。
> - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

3. 配置Demo工程。

   根据步骤1中获取的AppID和AppKey修改*RTCSuperClassRoom/RTCSuperClassRoom_Demo/src/main/java/com/aliyun/rtc/superclassroom/utils/MockAliRtcAuthInfo.java*文件中 `appID` 和 `appKey` 的值。

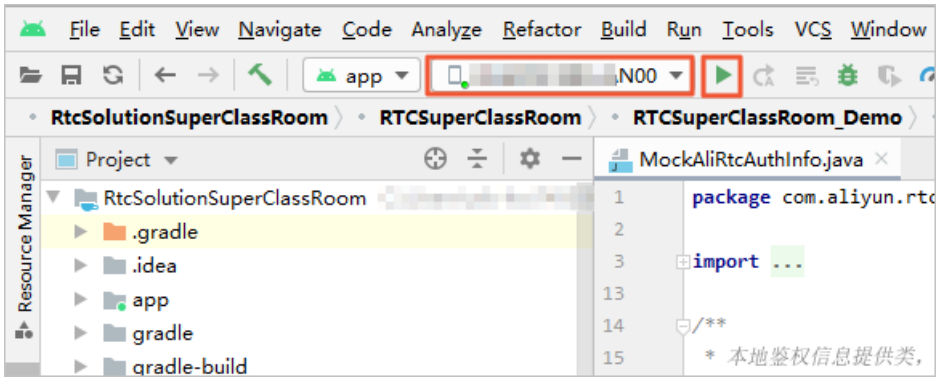

> **说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

4. 运行Demo。

   i. 打开Android Studio，单击**Open an Existing Project**，选择*android*目录下的*RtcSolutionSuperClassRoom*文件夹。

   ii. 单击，同步工程。

iii. 将Android设备与电脑有线连接，并在Android Studio中选择相对应的设备（暂不支持模拟器运行），单击，编译并运行。如果编译过程中出现问题或无法正常通话，请参见Android端运行常见问题。




> **说明** 将Android设备和电脑有线连接时，需要在Android设备的设置中开启开发者模式和USB调试模式，同时在Android设备上选择同意调试。

5. 开启超级小班课。
   i. 教师由Web端输入姓名进入教室，为助教和学生发送教室码。
   ii. 助教由Web端输入姓名和教室码进入教室，选择需要管理的小组进行管理。
   iii. 学生由Web端或移动端输入姓名和教室码进行教室听课学习，还可以进行连麦互动。

## Demo目录结构说明

项目结构如下所示：

| 文件名 | 说明 |
| --- | --- |
| RTCSuperClassRoom | 超级小班课功能实现库。更多信息，请参见RTCSuperClassRoom组件库目录说明。 |
| RTCSolutionCommon | 公共组建库。 |
| RTCViewCommo | 公共UI库。 |
| app | 程序入口。 |
| thirdparty-lib | 第三方库的引用。 |

RTCSuperClassRoom组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| activity | activity类。 |
| adapter | 列表控件adapter。 |
| api | 网络请求。 |

| 文件名 | 说明 |
| --- | --- |
| bean | 实体类。 |
| constant | 常量数据管理类，在此配置服务端请求域名和分享链接的域名。 |
| rtc | RTC。 |
| ui | 互动界面和登录界面。 |
| util | 工具类。 |
| view | 自定义view。 |

## API说明

### 功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destorySharedInstance | 毁单例对象。 |
| login | 加入房间。 |
| logout | 退出房间。 |
| muteLocalCamera | 切换是否停止发布本地视频。 |
| muteLocalMic | 切换是否停止发布本地音频。 |
| switchCamera | 切换摄像头。 |
| getUserInfo | 获取用户信息。 |
| registerCallBack | 注册回调。 |
| startPreview | 开始预览。 |
| stopPreview | 停止预览。 |
| startPublish | 开始推流。 |
| stopPublish | 停止推流。 |
| setRemoteViewConfig | 设置远端画面。 |
| getRemoteUserList | 获取远端用户列表。 |
| isInCall | 是否入会。 |
| isPreview | 是否预览。 |

回调接口

| API | 描述 |
| --- | --- |
| onJoin | 用户上线通知。 |
| onLeave | 用户下线通知。 |
| onOccurError | SDK报错。 |
| onOccurWarning | SDK警告。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onSDKError | SDK报错，需要销毁实例。 |
| onJoinChannelResult | 加入房间通知。 |
| onLeaveChannelResult | 离开房间通知。 |
| onNetworkQualityChanged | 网络状态回调。 |
| onUserVideoMuted | 用户muteVideo通知。 |
| onUserAudioMuted | 用户muteAudio通知。 |
| onSubscribeResult | 订阅成功回调。 |
| onUserSpeaking | 判断用户是否在说话。 |
| onRemoteTraceAvaliable | 当订阅情况发生变化时回调。 |

功能实现接口

- sharedInstance：获取单例对象。

  获取RTCSuperClassImpl的实例对象，初始化RTC SDK。

```
/**
     * 获取单例
     */
    public static BaseRTCSuperClass sharedInstance() {
        return RTCSuperClassImpl.sharedInstance();
    }
```

- destorySharedInstance：毁单例对象。

  销毁RTCSuperClassImpl的实例对象，销毁后需要再调用sharedInstance接口再次初始化实例。

```
    /**
     * 销毁实例
     */
    public abstract void destorySharedInstance();
```

- login：加入房间。

  根据输入的房间号、用户名加入RTC频道。

```
/**
     * 登录
     *
     * @param channelId 房间号
     * @param userName 昵称
     */
    public abstract void login(String channelId,  String userName);
```

- logout：退出房间。

  退出RTC频道。

```
    /**
     * 登出
     */
    public abstract void logout();
```

- muteLocalCamera：切换是否停止发布本地视频。

```
    /**
     * 切换是否停止发布本地视频
     * @param channelId  频道号
     * @param isMute 是否停止发布
     */
    public abstract void muteLocalCamera(String channelId,boolean isMute);
```

- muteLocalMic：切换是否停止发布本地音频。

```
/**
     * 切换是否停止发布本地音频
     *  @param channelId  频道号
     *  @param isMute 是否停止发布
     */
    public abstract void muteLocalMic(String channelId,boolean isMute);
```

- switchCamera：设置摄像头接口。

```
    /**
     * 设置摄像头
     */
    public abstract void switchCamera();
```

- getUserInfo：获取用户信息。

```
 /**
     * 获取用户信息
     * @param channelId  频道号
     * @param userId 用户id
     */
    public abstract AliRtcRemoteUserInfo getUserInfo(String channelId, String userId);
```

- registerCallBack：注册回调。

```
 /**
     * 注册回调
     */
    public abstract void registerCallBack(RTCSuperClassCallback rtcSuperClassCallback);
```

- startPreview：开始预览。

```
/**
     * 开始预览
     * @param viewGroup  显示画面的view
     */
    public abstract void startPreview(ViewGroup viewGroup);
```

- stopPreview：停止预览。

```
    /**
     * 停止预览
     */
    public abstract void stopPreview();
```

- startPublish：开始推流。

```
    /**
     * 开始推流
     * @param channelId  频道号
     */
    public abstract void startPublish(String channelId);
```

- stopPublish：停止推流。

```
   /**
     * 停止推流
     * @param channelId  频道号
     */
    public abstract void stopPublish(String channelId);
```

- setRemoteViewConfig：设置远端画面。

```
/**
     * 设置远端画面
     * @param channelId  频道号
     * @param canvas canvas对象
     * @param track track对象
     */
    public abstract void setRemoteViewConfig(String channelId, AliRtcEngine.AliVideoCanva
s canvas, String uid, AliRtcEngine.AliRtcVideoTrack track);
```

- getRemoteUserList：获取远端用户列表。

```
 /**
     * 获取远端用户列表
     * @param channelId  频道号
     */
    public abstract List<RTCUserInfo> getRemoteUserList(String channelId);
```

- isInCall：是否入会。

```
/**
    * 是否入会
    * @param channelId  频道号
    */
   public abstract boolean isInCall(String channelId);
```

- isPreview：是否预览。

```
/**
    * 是否预览
    * @param channelId  频道号
    */
   public abstract boolean isPreview(String channelId);
```

回调接口

- onJoin：用户上线通知。

```
/**
    * 用户上线通知
    *
    * @param userId 用户id
    */
   void onJoin(String channelId,String userId);
```

- onLeave：用户下线。

```
/**
    * 用户下线通知
    * @param channelId  频道号
    * @param userId 用户id
    */
   void onLeave(String channelId,String userId);
```

- onOccurError：SDK报错通知。

```
   /**
    * sdk报错
    *
    * @param channelId  频道号
    */
   void onOccurError(String channelId,int error);
```

- onOccurWarning：SDK警告。

```
   /**
    * sdk警告
    *
    * @param channelId  频道号
    */
   void onOccurWarning(String channelId,int error);
```

- onRoomDestroy：房间被销毁的回调。

```
    /**
     * 房间被销毁的回调
     * @param channelId  频道号
     */
    void onRoomDestroy(String channelId);
```

- onSDKError：SDK报错，需要销毁实例。

```
    /**
     * sdk报错,需要销毁实例
     * @param channelId  频道号
     */
    void onSDKError(String channelId,int error);
```

- onJoinChannelResult：加入房间通知。

```
    /**
     * 加入房间通知
     * @param channelId  频道号
     * @param result 0为成功 反之失败
     */
    void onJoinChannelResult(String channelId,int result);
```

- onLeaveChannelResult：离开房间通知。

```
    /**
     * 离开房间通知
     * @param channelId  频道号
     */
    void onLeaveChannelResult(String channelId,int result);
```

- onNetworkQualityChanged：网络状态。

```
 /**
      * 网络状态回调
      *
      * @param channelId  频道号
      * @param aliRtcNetworkQuality1 下行网络质量
      * @param aliRtcNetworkQuality  上行网络质量
      * @param userId                    String  用户ID
      */
    void onNetworkQualityChanged(String channelId,String userId, AliRtcEngine.AliRtcNetwo
rkQuality aliRtcNetworkQuality, AliRtcEngine.AliRtcNetworkQuality aliRtcNetworkQuality1);
```

- onUserVideoMuted：用户muteVideo通知。

```
/**
     * 用户muteVideo通知
     * @param channelId  频道号
     */
    void onUserVideoMuted(String channelId,String userId, boolean mute);
```

- onUserAudioMuted：用户muteAudio通知。

```
/**
     * 用户muteAudio通知
     * @param channelId  频道号
     */
    void onUserAudioMuted(String channelId,String userId, boolean mute);
```

- onSubscribeResult：订阅成功。

```
/**
     *
     * 订阅成功
     * @param channelId  频道号
     * @param userId  用户ID
     * @param result  0表示订阅成功，非0表示失败
     * @param videoTrack      订阅成功的视频流
     * @param audioTrack      订阅成功的音频流
     */
    void onSubscribeResult(String channelId,String userId, int result, AliRtcEngine.AliRt
cVideoTrack videoTrack, AliRtcEngine.AliRtcAudioTrack audioTrack);
```

- onUserSpeaking：判断用户是否在说话。

```
/**
     *
     * 用户是否在说话
     * @param channelId  频道号
     * @param userId  用户ID
     * @param isSpeaking 是否正在说话
     */
    void onUserSpeaking(String channelId,String userId, boolean isSpeaking);
```

- onRemoteTraceAvaliable：当订阅情况发生变化时的回调。

```
/**
     *
     * 当订阅情况发生变化时，返回这个消息 onSubscribeChangedNotify
     * @param channelId  频道号
     * @param userId  用户ID
     * @param videoTrack      订阅成功的视频流
     * @param audioTrack      订阅成功的音频流
     */
    void onRemoteTraceAvaliable(String channelId,String userId, AliRtcEngine.AliRtcAudioT
rack audioTrack, AliRtcEngine.AliRtcVideoTrack videoTrack);
```

# 2.6.4. iOS集成

通过阅读本文，您可以了解超级小班课iOS端的集成操作。

## 环境要求

iOS端具体环境要求，更多信息，请参见使用限制。

## 前提条件

- Web端服务已集成并开启。具体操作，请参见Web集成。

- 环境中已安装Xcode 9.0或以上版本，更多信息，请参见Xcode。
- 您需要持有Apple开发证书或个人账号。

## 操作步骤

1. 获取AppID和AppKey。此处建议记录一下AppID和AppKey，方便后续操作中使用。
   i. 登录RTC控制台。
   ii. 在左侧导航栏单击**应用管理**，进入应用管理页面。
   iii. 在**AppID/名称**列获取AppID。
   iv. 单击**操作**列的**查询AppKey**，获取AppKey。

   > **说明** 如果应用列表中没有您需要的应用，可以单击**创建应用**，创建新的应用。具体操作，请参见创建应用。

2. 下载并解压Demo，更多信息，请参见Demo源码下载。

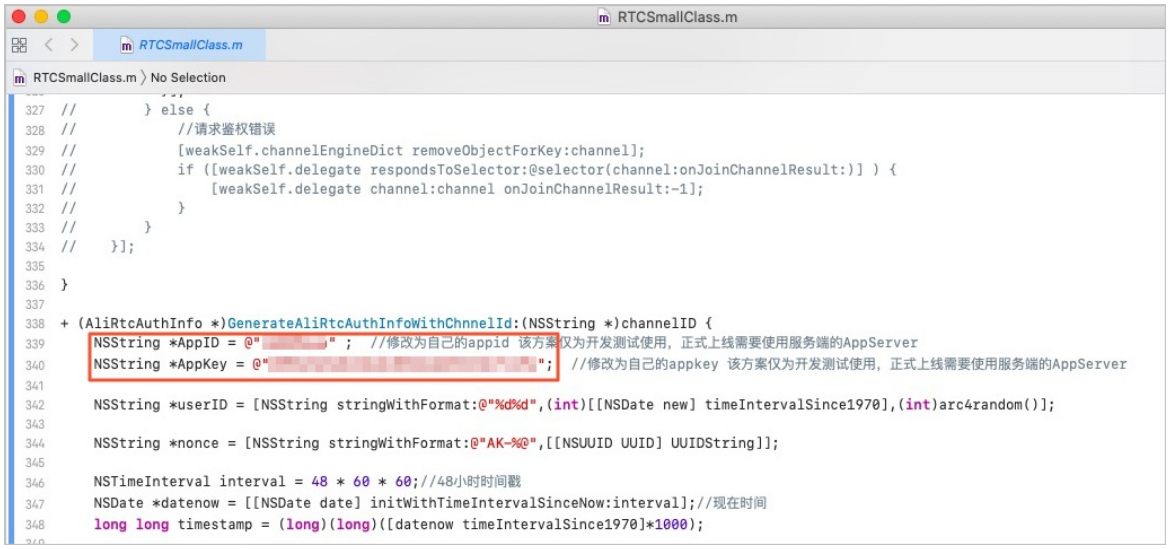
   > **说明**
   > - Demo源码中未集成AliRTC SDK，需要手动进行集成。
   > - 源码压缩文件内分为Web端、Android端、iOS端三个文件。
   > - 如果GitHub代码库下载缓慢，可安装加速插件等方式加速下载。

3. 打开终端，定位到iOS文件夹下Podfile文件所在的目录，执行**pod install**命令。

   安装后生成的Pods文件夹已在iOS目录下。

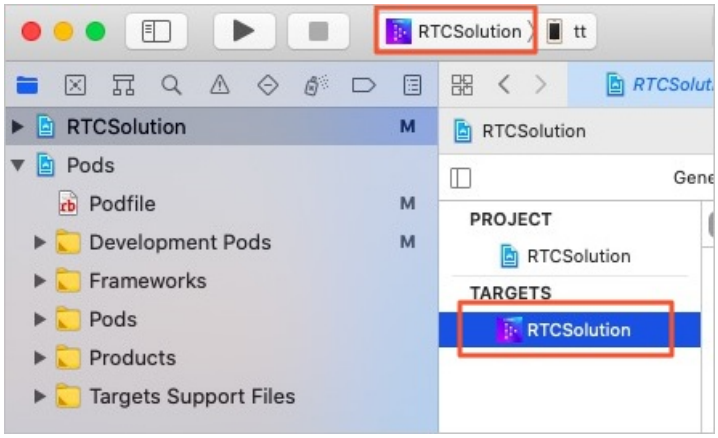
4. 配置Demo工程。

   根据步骤1中获取的AppID和AppKey修改*iOS/RTCSmallClass/RTCSmallClass/RTC/RTCSmallClass.m*文件中 `AppId` 和 `AppKey` 的值。



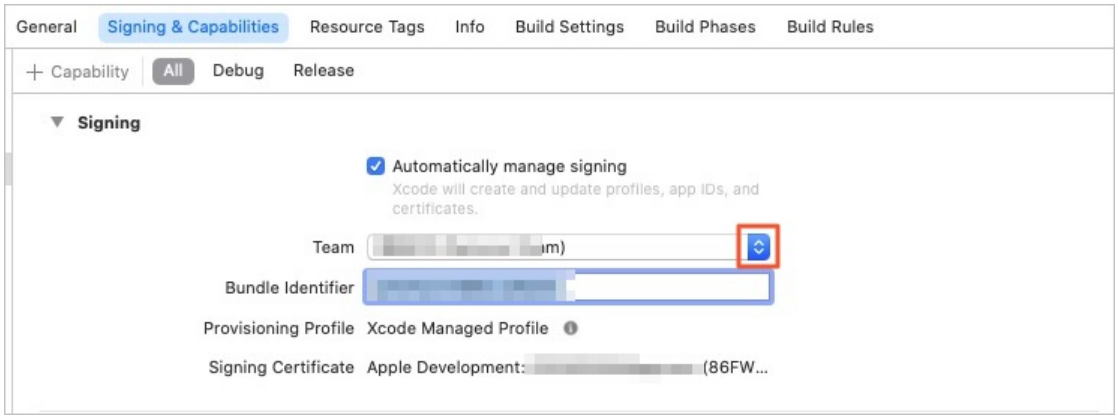
> **说明** 此处配置的AppID和AppKey很容易被反编译破解，如果被破解，攻击者可以盗用您的阿里云流量，因此AppID和AppKey仅适用于Demo演示及功能调试。在正式环境中您可以将Token计算代码集成到服务器中，并提供面向App的接口，在需要Token时由App向业务服务器发起请求获取动态Token。更多信息，请参见生成Token。

5. 运行Demo。
   i. 使用Xcode打开*iOS*目录下的*RTCSolution.xcworkspace*文件。
   ii. 选择运行的Target为RTCSolution，然后将iOS设备与电脑有线连接，并在Xcode中选择相对应的设备（暂不支持模拟器运行）。



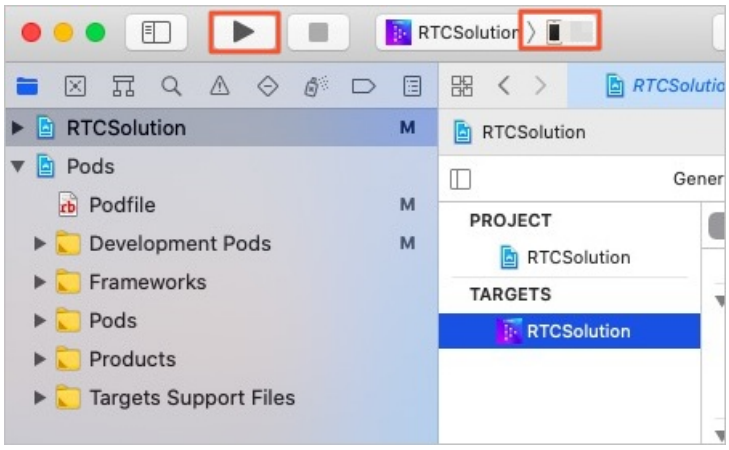

   iii. 单击**General**页签，修改**Bundle Identifier**，建议将Bundle Identifier改成com.<公司名>.<项目名>，避免由于Bundle已被注册从而运行失败。

iv. 单击Signing & Capabilities页签，选中Automatically manage signing，然后单击Team下拉框，根据实际情况选择Team。




> **说明** 如果之前没有添加过账号，可以选择Add an Account...，根据提示添加账号，然后在此处选择新添加的账号。

v. 单击 ▶，编译并运行。如果在编译过程中出现问题或无法正常通话，请参见iOS端运行常见问题。




6. 开启超级小班课。
   i. 教师由Web端输入姓名进入教室，为助教和学生发送教室码。
   ii. 助教由Web端输入姓名和教室码进入教室，选择需要管理的小组进行管理。
   iii. 学生由Web端或移动端输入姓名和教室码进行教室听课学习，还可以进行连麦互动。

## Demo目录结构说明

RTC把业务代码封装到RTCSmallClass的库中，因此只需在Podfile中指定RTCSmallClass库的路径，RTCSmallClass就可以以本地第三方库的形式移植到其他项目中。如下所示：

```
pod 'RTCSmallClass', :path => '../RTCSmallClass'
   ##   pod 'RTCSmallClass'  表示项目依赖RTCSmallClass库
   ##   path => './RTCSmallClass'  表示RTCSmallClass库的位置(相对于Podfile文件)
```

RTCSmallClass组件库目录说明，如下所示：

| 文件名 | 说明 |
| --- | --- |
| RTCSmallClass.podspec | 组件的描述文件。 |
| RTCSmallClass.bundle | 存放资源的bundle。 |
| SmallClassLogin | 登录页。 |
| SmallClassMainController | 通话页。 |
| SmallClassGroupMemberController | 小组成员页。 |

## API说明

### 功能实现接口

| API | 描述 |
| --- | --- |
| sharedInstance | 获取单例对象。 |
| destorySharedInstance | 销毁实例对象。 |
| login | 加入房间。 |
| logout | 退出房间。 |
| muteLocalCamera | 切换是否停止发布本地视频麦。 |
| muteLocalMic | 切换是否停止发布本地音频。 |
| switchCamera | 切换摄像头。 |
| getUserInfo | 获取用户信息。 |
| startPreview | 开始预览。 |
| stopPreview | 停止预览。 |
| startPublish | 开始推流。 |
| stopPublish | 停止推流。 |
| getRemoteUserList | 获取远端用户列表。 |

### 回调接口

| API | 描述 |
| --- | --- |
| onJoin | 用户上线通知。 |
| onLeave | 用户下线通知。 |
| onOccurError | SDK报错。 |

| API | 描述 |
| --- | --- |
| onOccurWarning | SDK警告。 |
| onRoomDestroy | 房间被销毁的回调。 |
| onSDKError | SDK报错，需要销毁实例。 |
| onJoinChannelResult | 加入房间通知。 |
| onLeaveChannelResult | 离开房间通知。 |
| onNetworkQualityChanged | 网络状态回调。 |
| onUserVideoMuted | 用户muteVideo通知。 |
| onUserAudioMuted | 用户muteAudio通知。 |
| onSubscribeResult | 订阅成功回调。 |
| onUserSpeaking | 判断用户是否在说话。 |
| onRemoteTraceAvaliable | 当订阅情况发生变化时的回调。 |

功能实现接口

- sharedInstance：获取单例对象。

```
/**
     * 获取单例
     */
   [RTCSmallClass sharedInstance];
```

- destorySharedInstance：毁单例对象。

```
/// 销毁RTCSDK
- (void)destroySharedInstance;
```

- login：加入房间。

```
/// 加入频道
/// @param channel 频道
/// @param userName 用户昵称
- (void)login:(NSString *)channel userName:(NSString *)userName;
```

- logout：退出房间。

```
/// 离开所有频道
- (void)logout;
```

- muteLocalCamera：切换是否停止发布本地视频。

```
/// 禁用摄像头
/// @param mute 是否禁用
/// @param channel 频道
- (int)muteLocalCamera:(BOOL)mute channel:(NSString *)channel;
```

- muteLocalMic：切换是否停止发布本地音频。

```
/// 静音
/// @param mute 是否静音
/// @param channel 频道
- (int)muteLocalMic:(BOOL)mute channel:(NSString *)channel;
```

- switchCamera：设置摄像头。

```
/// 旋转摄像头
- (int)switchCamera;
```

- getUserInfo：获取用户信息。

```
 /// 获取用户信息
/// @param uid 用户id
/// @param channel 频道
- (NSDictionary *)getUserInfo:(NSString *)uid channel:(NSString *)channel;
```

- startPreview：开始预览。

```
/// 开始本地预览
/// @param preview 预览的view
- (void)startPreview:(UIView *)preview;
```

- stopPreview：停止预览。

```
/// 停止本地预览
- (void)stopPreview;
```

- startPublish：开始推流。

```
  /// 开始推流
/// @param channel 频道
- (void)startPublish:(NSString *)channel;
```

- stopPublish：停止推流。

```

/// 停止推流
/// @param channel 频道
- (void)stopPublish:(NSString *)channel;
```

- getRemoteUserList：获取远端用户列表。

```
/// 获取某个频道远端用户列表
/// @param channel 频道
- (NSArray *)getRemoteUserList:(NSString *)channel;
```

回调接口

- onJoin：用户上线通知。

```
  /// 远端用户加入频道
/// @param channel 频道
/// @param uid 用户id
- (void)channel:(NSString *)channel onJoin:(NSString *)uid;
```

- onLeave：用户下线通知。

```
/// 远端用户离开频道
/// @param channel 频道
/// @param uid 用户id
- (void)channel:(NSString *)channel onLeave:(NSString *)uid;
```

- onOccurError：SDK报错。

```
/// 错误码
/// @param channel 频道
/// @param error 错误码
- (void)channel:(NSString *)channel onOccurError:(int)error;
```

- onOccurWarning：SDK警告。

```
/// 警告
/// @param channel 频道
/// @param warn 警告码
- (void)channel:(NSString *)channel onOccurWarning:(int)warn;
```

- onRoomDestroy：房间被销毁的回调。

```
   /// 房间被销毁通知
- (void)onRoomdestroy;
```

- onSDKError：SDK报错，需要销毁实例。

```
/// sdk严重错误 需要销毁引擎
/// @param channel 频道
/// @param error 错误码
- (void)channel:(NSString *)channel onSDKError:(int)error;
```

- onJoinChannelResult：加入房间通知。

```
  /// 加入频道
/// @param channel 频道
/// @param result 结果
- (void)channel:(NSString *)channel onJoinChannelResult:(int)result;
```

- onLeaveChannelResult：离开房间通知。

```
/// 离开频道
/// @param channel 频道
/// @param result 结果
- (void)channel:(NSString *)channel onLeaveChannelResult:(int)result ;
```

- onNetworkQualityChanged：网络状态。

```
/// 网络变化判断
/// @param uid 用户id
/// @param upQuality 上行网络质量
/// @param downQuality 下行网络质量
- (void)onNetworkQualityChanged:(NSString *)uid
                upNetworkQuality:(AliRtcNetworkQuality)upQuality
              downNetworkQuality:(AliRtcNetworkQuality)downQuality;
```

- onUserVideoMuted：用户muteVideo通知。

```
/// 用户是否禁用摄像头
/// @param channel 频道
/// @param uid 用户id
/// @param isMute 是否禁用
- (void)channel:(NSString *)channel onUserVideoMuted:(NSString *)uid videoMuted:(BOOL)isM
ute;
```

- onUserAudioMuted：用户muteAudio通知。

```
/// 用户是否静音
/// @param channel 频道
/// @param uid 用户id
/// @param isMute 用户是否静音
- (void)channel:(NSString *)channel onUserAudioMuted:(NSString *)uid audioMuted:(BOOL)isM
ute;
```

- onSubscribeResult：订阅成功。

```
/// 订阅成功
/// @param channel 频道
/// @param uid 用户id
/// @param result 结果
- (void)channel:(NSString *)channel onSubscribe:(NSString *)uid result:(BOOL)result;
```

- onUserSpeaking：判断用户是否在说话。

```
/// 用户说话状态改变
/// @param channel 频道
/// @param uid 用户id
/// @param speaking 是否正在说话
/// @param name 用户昵称
- (void)channel:(NSString *)channel onUserSpeaking:(NSString *)uid speaking:(BOOL) speaki
ng displayName:(NSString *)name;
```

- onRemoteTraceAvaliable：当订阅情况发生变化时的回调。

```
/// 远端流变化
/// @param channel 频道
/// @param uid 用户id
/// @param abaliable 是否可用
- (void)channel:(NSString *)channel onRemoteTrace:(NSString *)uid avaliable:(BOOL)abaliab
le;
```