蚂蚁科技

音视频通话
使用指南

文档版本：20231226

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

未经蚂蚁集团事先书面许可，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。

商标声明

蚂蚁集团 ANT GROUP及其他蚂蚁集团相关的商标均为蚂蚁集团所有。本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

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。蚂蚁集团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，并在蚂蚁集团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蚂蚁集团授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

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 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 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 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 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` *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.产品简介 07
2.快速开始 10
3.接入客户端 11
3.1. 接入 Android 11
3.1.1. 快速开始 11
3.1.2. Android 进阶功能 12
3.1.3. 代码示例 14
3.2. 接入 iOS 15
3.2.1. 添加 iOS SDK 15
3.2.2. 使用 iOS SDK 16
3.2.3. iOS 进阶功能 21
3.2.4. 代码示例 29
3.2.5. 常见问题 29
3.3. 接入 Web 30
3.4. 接入 Linux 33
3.4.1. 快速开始 33
3.4.2. 使用 Linux SDK 34
3.4.3. Demo 示例 39
3.4.4. 常见问题 40
3.5. 接入支付宝小程序 41
3.5.1. 快速开始 41
3.5.2. 集成插件 42
3.5.3. 组件功能 49
3.5.4. 在 uniapp 中使用音视频插件 53
3.6. 接入微信小程序 55
3.6.1. 快速开始 55

3.6.2. 版本更新说明 58
4.安全加签 59
5.使用控制台 62
5.1. 通话应用管理 62
5.1.1. 创建通话应用 62
5.1.2. 管理通话应用 62
5.2. 录制文件管理 68
5.3. 签名校验 69
5.4. 用量统计 71
6.API 参考 74
6.1. Android API 74
6.1.1. 权限及隐私说明 74
6.1.2. 重要参数 74
6.1.3. 公共接口 83
6.1.4. 回调函数 96
6.1.5. 错误码 112
6.2. iOS API 113
6.2.1. 重要参数 113
6.2.2. 公共接口 115
6.2.3. 回调函数 127
6.3. Web API 128
6.3.1. Web SDK 发布说明 128
6.3.2. 主调接口 136
6.3.3. 回调接口 173
6.3.4. 错误码 181
6.4. Linux API 187
6.4.1. 概述 188
6.4.2. 接口定义 191

6.4.3. 重要参数 197
6.4.4. 主调函数 200
6.4.5. 回调函数 200
6.4.6. 错误码 202
6.5. 支付宝小程序 API 203
6.5.1. 概述 203
6.5.2. 重要参数 204
6.5.3. 状态码和错误码 208
6.6. 微信小程序 API 210
6.6.1. 概述 210
6.6.2. 主调接口 211
6.6.3. 回调接口 214
6.6.4. 状态码 218
7.附录 219

1.产品简介

音视频通话组件（Mobile Real-Time Communication，简称 MRTC）是 mPaaS 提供的音频、视频通话组件。该组件功能丰富，提供纯语音通话和视频通话功能，支持 PC、移动端、IoT 设备等多终端接入。音视频通话可实现一对一通话及多人会议，通话过程中支持屏幕录制、屏幕共享、截图等功能，同时支持即时文字消息和文件传输。此外，支持实时语音识别，能够识别对端的语音确认，辅助本端判断对端的意向；点播功能可实现在视频通话过程中，播放视频、PPT 等多种提示画面。

音视频通话具有安全性高的特点，端到端全链路加密，符合国密标准，视频录制传输及存储均进行加密。

功能特点

- **视频通话**
  - 多种参与模式：支持一对一视频通话及多人视频通话。
  - 多平台：支持 iOS、Android、PC Web、H5 以及小程序。
  - 多端互通：支持手机、PC、IoT 设备之间互联互通。
  - 会话保持：网络短暂异常、网络切换时，业务流程不中断，保持会话的持续性。
  - 自定义视频规格、自适应视频规格：支持自定义宽、高、最大帧率、最大码率，并能在上限范围内根据网络状况自适应调整视频规格。
- **通话录制**
  - 多粒度录制控制：纯语音录制、纯视频录制、音视频混合录制、一人录制、两人录制、多人录制。
  - 多端录制：根据安全要求，既可以在坐席侧录制，也可以在服务端录制。
  - 多种画面组合模式：支持画中画、九宫格模式。
  - 叠加水印：支持实时叠加文字、图片水印、时间戳，以防篡改。
  - 多种存储系统：支持 NAS、OSS、AFTS、本地文件系统。
  - 多格式：支持多种常见格式，包括 FLV、MP4 等。
- **数据通信**
  - 即时通讯：支持简单的文字消息，方便在视频通话前进行沟通。
  - 文件传输：上传文件后，传输文件链接到对端，供对端下载获取文件。
  - 命令透传：为业务提供透明可靠的命令传输通道。
- **自动化和智能化**
  - 自动语音风险提示：自动播报风险提示等内容，代替人工说明。
  - 视频点播：在视频通话过程中，播放视频、PPT 等风险提示画面。
  - 自动纪要：实时识别视频通话的语音内容，并按照时间分角色交替文字记录对话内容。
  - 自动确认：识别客户对风险提示的语音确认，辅助坐席判断客户的意向。
- **增强功能**
  - 屏幕共享：将本端的屏幕内容投送到其他端。
  - 截图：支持远程控制摄像头拍照、本端截图。
- **系统转接**
  - 旁路直播：云端混音合图，转接直播系统。支持推流到 CDN、第三方直播平台。

产品优势

- **高安全性**

  - 动态加签验签。
  - 端到端全链路加密，符合国密标准。
  - OpenAPI 动态密钥鉴权。
  - 录制加密传输、存储。
- **弱网环境下视频质量自适应**
  - 根据网络丢包类型、丢包率、RTT、网络抖动评估，来准确的评估网络有效带宽。
  - 经过优化的视频质量自适应算法，视频清晰度、流畅度、尺寸可随目标码率实时变化。
  - 优先保证感兴趣区域的视频质量。
- **高可靠的媒体流和信令传输**
  - 实时流媒体传输平台，确保传输稳定。
  - 冗余链路和优选，动态切换。
  - 信令在节点到节点、端到端实现了双层高可靠机制，既降低了时延，又提高了可靠性，使可靠性达到了99.95%。
- **低时延**
  - 超低时延，远程操控领域可达到 200 ms。
  - 快速对焦。
  - 改进的 FEC 算法，抗丢包 40%。
  - H.264-highprofile 技术，可减少高达 50% 的带宽消耗。
- **多端接入**
  - 支持 PC、手机、Pad、小程序、IoT 设备等多终端接入，不同终端和 App 之间能够互通。
- **功能丰富**
  - 双录：云端双向录制、多粒度、多格式、个性化水印、多种存储方式。
  - 智能坐席：自动语音风险提示、视频点播、自动纪要。
- **接口简洁，接入方便**
  - 提供轻量版本，便于接入。
  - 已有业务最快接入纪录：一周开发上线。

产品架构

音视频通话的产品架构如下：

业务支持

- 视频客服
- 视频开户
- 视频会议
- 在线业务办理

产品功能

- 语音通话
- 双方通话
- 多方通话
- 点播
- 录制
- 屏幕共享
- 截屏
- 语音识别
- 美颜
- 即时消息
- 文件传输
- 智能排队

数据通道

- 信令
- 视频流传输
- 会话管理
- 录制服务
- 传输加密
- 点播服务
- 存储服务

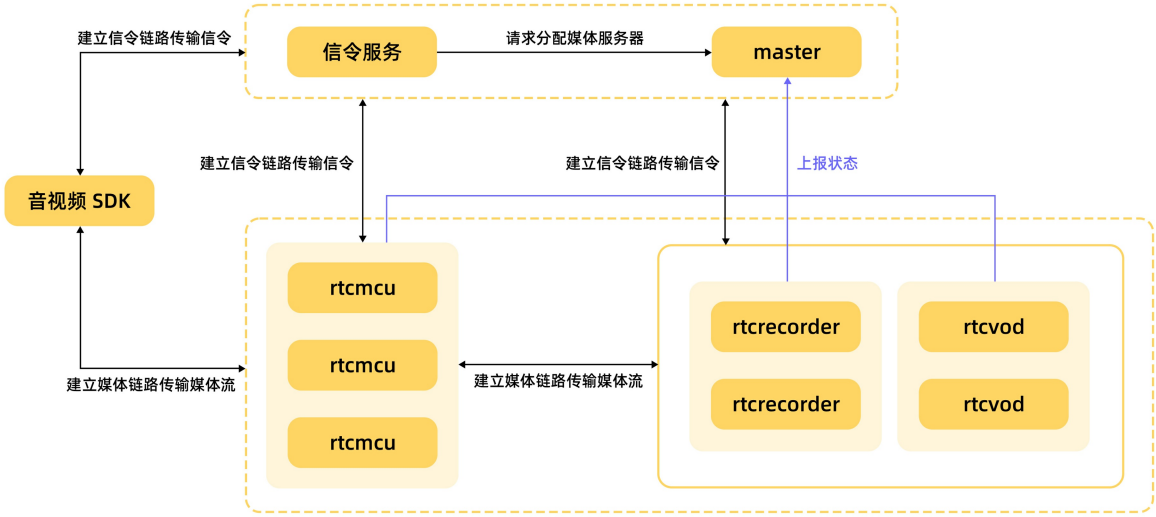
流媒体基础

- 音视频采集
- 编解码
- 视频处理
- 渲染
- Android
- IOS
- PC
- 小程序

数据监控

- 断网重连
- 弱网自适应
- 弱网告警
- 帧率告警
- 流量监控

音视频通话的系统架构如下：



- **音视频 SDK**：客户端 SDK，现有支持 Android、iOS、PC Web、小程序、Linux 以及 Windows。
- **信令服务**：负责信令连接管理、信令转发、房间管理、生命周期维护。
- **master**：均衡调度服务器，负责媒体服务器的均衡调度、存活管理。
- **rtcmcu**：媒体转发服务，负责媒体流接收、转发、处理、状态维护。
- **rtcrecorder**：录制服务，负责录制存储。
- **rtcvod**：点播服务，负责音视频点播。

2.快速开始

本节将为您介绍快速使用音视频通话的操作步骤。

1. 登录 mPaaS 控制台，点击 **+ 创建应用**，输入应用名并创建 mPaaS 应用。
2. 进入 mPaaS 应用，在左侧导航栏中选择 **音视频通话** > **通话应用管理** 进入音视频通话产品页。
3. 创建通话应用。创建后可查看 `bizName` 和密钥，相同 `bizName` 下的用户可互相通话。
4. 生成签名。音视频通话中提供两种签名生成方法：
   - 服务端生成：通过安全加签生成签名。
   - 控制台生成：在控制台中生成临时签名，可用于临时体验视频通话产品。
5. 使用音视频通话。客户端 SDK 初始化时传入生成的签名和 API 参数，开始音视频通话。

> **说明**
>
> 如果有更多接入相关问题，欢迎搜索群号 34717743 加入钉钉群进行咨询交流。除接入外，该群还提供全方位的技术支持。

3.接入客户端

3.1. 接入 Android

3.1.1. 快速开始

音视频通话只在 10.1.68 及以上版本基线中提供支持。

前置条件

- 若采用 **原生 AAR 方式** 接入，需先完成 将 mPaaS 添加至您的项目 。并确保工程根目录 `build.gradle` 文件中，有如下依赖：

  ```
  classpath 'com.android.boost.easyconfig:easyconfig:2.7.5'
  ```

  确保主工程（android main module）的 `build.gradle` 中有如下配置：

  ```
  apply plugin: 'com.alipay.apollo.baseline.config'
  ```

- 若采用 **组件化（Portal&Bundle）方式** 接入，需先完成 组件化接入流程 。

添加 SDK

原生 AAR 方式

参考 AAR 组件管理，通过 **组件管理（AAR）** 在工程中安装 **音视频通话** 组件。

组件化方式

在 Portal 和 Bundle 工程中通过 **组件管理** 安装 **音视频通话** 组件。更多信息，参考 管理组件依赖 。

初始化 mPaaS

如果使用 **原生 AAR 接入**，则需要初始化 mPaaS。

在 Application 中添加以下代码：

```
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS 初始化回调设置。
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // 此回调表示 mPaaS 已经初始化完成，mPaaS 相关调用可在这个回调里进行。
            }
        });
    }

    @Override
    public void onCreate() {
        super.onCreate();
        // mPaaS 初始化。
        QuinoxlessFramework.init();
    }
}
```

使用 SDK

1. 初始化 engine，设置自动发布和订阅。将 `autoPublish` / `autoSubscribe` 设置为 true。

```
AlipayRtcEngine rtcEngine = AlipayRtcEngine.getInstance( this );
rtcEngine.setRtcListenerAndHandler( engineEventListener, eventHandler );
rtcEngine.setImListener( imListener );
rtcEngine.setInviteListener( inviteListener );
rtcEngine.setServerAddr( SERVER_ONLINE );
rtcEngine.setAutoPublishSubscribe( autoPublish, autoSubscribe );
publishConfig = new PublishConfig();
rtcEngine.configAutoPublish( publishConfig );
```

2. 创建房间/加入房间。

```
CreateRoomParams createRoomParams = new CreateRoomParams();
rtcEngine.createRoom( createRoomParams );
JoinRoomParams joinRoomParams = new JoinRoomParams();
rtcEngine.joinRoom( joinRoomParams );
```

3. 监听本端和对端的 view。
   - 本端：`onCameraPreviewInfo(final ARTVCView view )`
   - 对端：`void onRemoteViewFirstFrame( final FeedInfo info, final ARTVCView _view )`

3.1.2. Android 进阶功能

本文介绍的是音视频通话 API 在 Android 中的进阶功能。

设置音视频通话或纯音频通话

设置纯音频通话，在 初始化 engine 时，按如下方法配置 `publishConfig`：

```
publishConfig = new PublishConfig();
```

- 纯音频通话

```
publishConfig.videoSource = VIDEO_SOURCE_NULL;
publishConfig.audioSource = AUDIO_SOURCE_MIC;
```

- 视频通话

```
publishConfig.videoSource = VIDEO_SOURCE_CAMERA;
publishConfig.audioSource = AUDIO_SOURCE_MIC;
```

动态调整分辨率

在通话成功建立后，可按如下方法动态调整本端发送的分辨率：

```
public void updateVideoProfile( VideoProfile videoProfile, int maxBitrate )
```

其中：

- `videoProfile` 设置为新配置的分辨率配置。
- `maxBitrate` 设置为新配置的码率，`0` 为使用默认值。

通话质量监控

在通话成功建立后，监听以下两个回调，可以实时获取当前视频通话质量：

```
void onStatisticDebugInfo( StatisticInfoForDebug infoForDebug, FeedInfo feedInfo ); //
获取通话过程调试信息
void onRealTimeStatisticInfo( RealTimeStatisticReport report, FeedInfo feedInfo );  //
获取实时监控信息
```

其中 `feedInfo` 为对应流的信息，包括对端和本端。

网络变化监控

在通话成功建立后，监听以下回调，获取带宽信息：

```
void onBandwidthImportanceChangeNotify( boolean isLow, double currentBandwidth, FeedInf
o feedInfo ); //带宽不足通知
```

带宽过低会通过 `isLow` 通知，如果带宽过低，可能会断开或者通话质量差。

屏幕共享

开启屏幕共享功能，按如下设置后，可实现在视频通话的同时共享屏幕。

1. 在 初始化 engine 时，不能配置自动发布，但是可以配置自动订阅。即：

   ```
   rtcEngine.setAutoPublishSubscribe( autoPublish, autoSubscribe );
   ```

   其中：

   - `autoPublish` 必须为 `false` 。
   - `autoSubscribe` 无特殊限制。

2. 初始化后，在 创建或加入房间 后，调用：

```
publishConfig = new PublishConfig();
publishConfig.videoSource = VIDEO_SOURCE_SCREEN;
rtcEngine.publish( publishConfig );
```

截屏功能

在通话建立后，需要截屏，可以通过以下方法进行截屏：

```
public void snapshot( FeedInfo info )
```

其中 `feedInfo` 为对应截屏的流信息。

截屏结果通过以下方法进行回调：

```
void onSnapShotComplete( Bitmap image, FeedInfo feedInfo ); //截屏图像回调
```

自定义推流

开启自定义推流。

1. 与屏幕共享相同，在 初始化 engine 时，不能配置自动发布，但是可以配置自动订阅。即：

   ```
   rtcEngine.setAutoPublishSubscribe( autoPublish, autoSubscribe );
   ```

   其中：

   - `autoPublish` 必须为 `false` 。
   - `autoSubscribe` 无特殊限制。

2. 在 创建或加入房间 后，调用：

   ```
   publishConfig = new PublishConfig();
   publishConfig.videoSource = VIDEO_SOURCE_CUSTOM;
   rtcEngine.publish( publishConfig );
   ```

3. 在发布成功之后，通过以下方法的结果不断的输入图像数据：

   ```
   public void pushCustomVideoData(byte[] bytes, int width, int height, int rotation ) /
   /自定义推流数据，目前仅支持 nv21 格式数据
   ```

本地预览

如果需要在未进入视频通话时，提前开启预览，可以在 初始化 engine 后调用以下方法开启摄像头预览：

```
public void startCameraPreview()
```

预览的 `view` 可以参考 监听本端和对端的 view 。

3.1.3. 代码示例

本文介绍音视频通话接入 Android 的代码示例及其使用步骤。

下载代码示例

单击这里 获取音视频通话代码示例。

使用示例代码

1. 替换示例工程中的配置文件。
   i. 登录 mPaaS 控制台。
   ii. 创建 mPaaS 应用。
   iii. 下载 mPaaS 应用的 `.config` 格式 配置文件 放入音视频通话示例工程的 `app/` 目录下，用来替换已有的配置文件。
2. 创建通话应用 并获取 bizName。在控制台左侧导航栏选择 **音视频通话** > **通话应用管理**，创建通话应用或进入已有的通话应用，获取通话应用的 bizName。
3. 控制台 生成临时签名。在控制台左侧导航栏选择 **音视频通话** > **签名校验**，在上方的 **生成临时签名** 区域，选择通话应用，输入 **UserID** 生成临时签名。

   > **说明**
   > - 更换通话应用或 UserID 后需要重新生成临时签名。
   > - 注意临时签名的有效期，过期后需重新生成。

4. 配置示例工程。运行示例工程，在 App 设置页填写从以上步骤中获取到的用户 ID（userId）、业务名称（bizName）和临时签名（signature）。

   > **说明**
   > 为保障您的流量安全，业务上线后务必 通过服务端生成签名。

5. 发起或加入音视频通话。 您可在 App 中创建通话房间发起音视频通话，或者在页面上输入其他用户创建的视频通话房间号（roomId）和 token 加入音视频通话。

   > **说明**
   > 在同一个通话应用中（使用同一个 bizName）的不同端的 userId 必须互不相同。

3.2. 接入 iOS

3.2.1. 添加 iOS SDK

音视频通话只在 10.1.68 及以上版本基线中提供支持。目前 iOS 音视频通话中暂不支持 ARMv7 架构。

音视频通话支持 **基于 mPaaS 框架**、**基于原生框架且使用 mPaaS 插件** 以及 **基于原生框架且使用 CocoaPods** 这 3 种接入方式，您可以自由选择。

前置条件

您已经接入工程到 mPaaS。更多信息，请参见以下内容：

- 基于 mPaaS 框架接入
- 基于已有工程且使用 mPaaS 插件接入
- 基于已有工程且使用 CocoaPods 接入

添加 SDK

根据您采用的接入方式，请选择相应的添加方式。

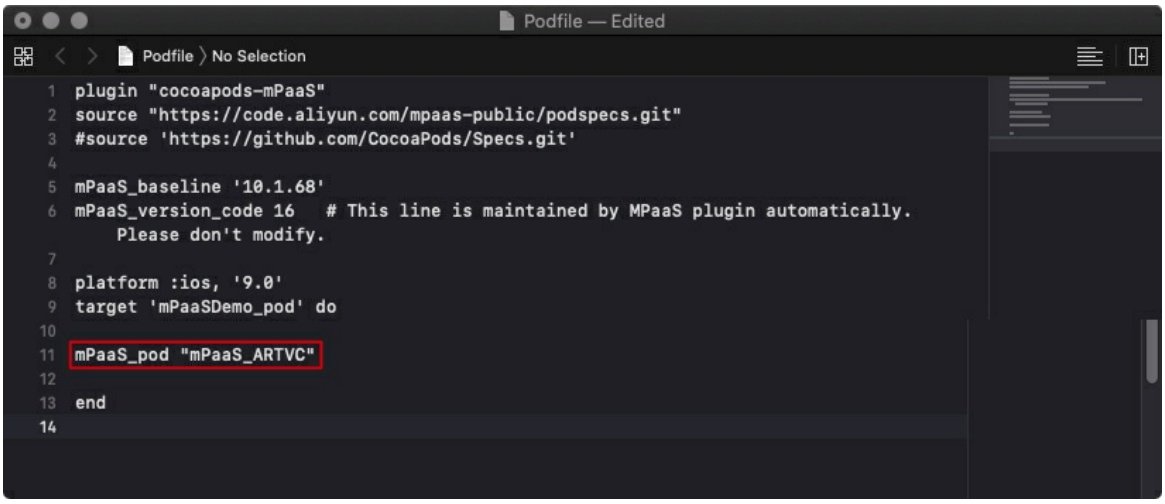
- **使用 mPaaS Xcode Extension**。此方式适用于采用了 **基于 mPaaS 框架接入** 或 **基于已有工程且使用 mPaaS 插件接入** 的接入方式。

i. 点击 Xcode 菜单项 **Editor** > **mPaaS** > **编辑工程**，打开编辑工程页面。

ii. 选择 **视频通话**，保存后点击 **开始编辑**，即可完成添加。

- **使用 cocoapods-mPaaS 插件**。此方式适用于采用了 **基于已有工程且使用 CocoaPods 接入** 的接入方式。

  i. 在 Podfile 文件中，使用 `mPaaS_pod "mPaaS_ARTVC"` 添加音视频通话组件依赖。



  ii. 执行 `pod install` 即可完成接入。

3.2.2. 使用 iOS SDK

本文介绍 iOS SDK API 的使用方法。

权限与隐私

- 添加摄像头权限。接入方 App 需要在 Info.plist 中添加如下代码：

```
<key>NSCameraUsageDescription</key>
<string>Camera access needed for video calling</string>
```

- 添加麦克风权限。接入方 App 需要在 Info.plist 中添加如下代码：

```
<key>NSMicrophoneUsageDescription</key>
<string>Microphone access needed for video calling</string>
```

> **重要**
>
> 由于通话功能需要开启麦克风权限，因此在进入通话界面之前，请检查接入方 App 是否具备麦克风权限。如果具备麦克风权限，则直接进入通话界面；否则引导用户开启麦克风权限。

麦克风权限检查请参见系统中的 AVCaptureDevice 类，示例如下：

```
[AVCaptureDevice requestAccessForMediaType:mediaType completionHandler:^(BOOL granted
) {
        if (granted) {

        }
    }];
```

初始化 ARTVCEngine 并监听事件回调

- 初始化设置 uid 和 delegate。

```
_artvcEgnine = [[ARTVCEngine alloc] init];
_artvcEgnine.uid = self.uid;
_artvcEgnine.delegate = self;
```

- 监听事件回调（按需实现回调，比如需要实现错误处理）。

```
#pragma mark - ARTVCEngineDelegate
-(void)didReceiveRoomInfo:(ARTVCRoomInfomation*)roomInfo{
}
-(void)didReceiveLocalFeed:(ARTVCFeed*)localFeed{
  self.feedForPreview = localFeed;
}
-(void)didEncounterError:(NSError *)error forFeed:(ARTVCFeed*)feed{
  [self showToastWith:[NSString stringWithFormat:@"%@, Error:%@",feed,error] duration
:2.0];
}
.....
```

设置视频编码分辨率

设置视频编码分辨率代码如下：

```
//设置视频编码分辨率，默认是 ARTVCVideoProfileType_640x360_15Fps。
artvcEgnine.videoProfileType = ARTVCVideoProfileType_640x360_15Fps;
```

设置自动/手动推拉流开关

设置自动/手动推拉流开关代码如下，默认为自动推流/拉流。

```
_artvcEgnine.autoPublish = YES;
_artvcEgnine.autoSubScribe = YES;
```

设置是音视频通话还是纯音频通话

- 音视频通话

```
ARTVCPublishConfig* config = [[ARTVCPublishConfig alloc] init];
config.videoEnable = YES;//默认是 YES
config.audioEnable = YES;//默认是 YES
config.videoProfile = _artvcEgnine.videoProfileType;
_artvcEgnine.autoPublishConfig = config;
ARTVCSubscribeOptions* options = [[ARTVCSubscribeOptions alloc] init];
_artvcEgnine.autoSubscribeOptions = options;
```

- 纯音频通话

```
ARTVCPublishConfig* config = [[ARTVCPublishConfig alloc] init];
config.videoEnable = NO;
config.audioEnable = YES;//默认是 YES
_artvcEgnine.autoPublishConfig = config;
ARTVCSubscribeOptions* options = [[ARTVCSubscribeOptions alloc] init];
options.receiveVideo = NO;
_artvcEgnine.autoSubscribeOptions = options;
```

音视频通话下启动相机预览

> **说明**
>
> 如果是纯音频通话，则跳过此步骤。

```
//默认使用前置摄像头，如果设置为 YES 则使用后置摄像头。
[_artvcEgnine startCameraPreviewUsingBackCamera:NO];
```

- 启动相机预览后，如果本地 feed 没有被回调过。之后回调会返回一个 `ARTVCFeed` 对象，可用于关联后续返回的渲染 view。

```
-(void)didReceiveLocalFeed:(ARTVCFeed*)localFeed forPublishConfig:
(ARTVCPublishConfig*)publishConfig{
self.feedForPreview = localFeed;
}
```

- 预览 view 初始化的回调。

```
-(void)didVideoRenderViewInitialized:(UIView*)renderView forFeed:(ARTVCFeed*)feed{
if([feed isEqual:self.feedForPreview]){
[self showToastWith:@"video preview view created" duration:1.0];
}else{
self.feedForRemote = feed;
};
//可触发 UI 布局，把 renderView add 到 view 层级中去
}
```

- 预览首帧渲染的回调。

```
-(void)didFirstVideoFrameRendered:(UIView*)renderView forFeed:(ARTVCFeed*)feed{
}
```

创建或者加入房间

1. 作为主叫方，创建房间。

```
ARTVCCreateRoomParams* params = [[ARTVCCreateRoomParams alloc] init];
 params.uid = self.uid;
 params.bizName = DEMO_BIZ;
 params.subBiz = DEMO_SUBBIZ;
 params.signature = DEMO_SIGNATURE;
 [_artvcEgnine createRoom:params];
```

   - 创建房间成功，会有房间信息回调。

```
-(void)didReceiveRoomInfo:(ARTVCRoomInfomation*)roomInfo{
}
```

   - 创建房间失败，会有 Error 回调。

```
//error.code == ARTVCErrorCodeProtocolErrorCreateRoomFailed
-(void)didEncounterError:(NSError *)error forFeed:(ARTVCFeed*)feed{
}
```

   - 其他人加入房间后，会有成员加入房间的回调。

```
-(void)didParticepantsEntered:(NSArray<ARTVCParticipantInfo*>*)participants{
}
```

2. 作为主被叫方，加入房间。

```
ARTVCJoinRoomParams* params = [[ARTVCJoinRoomParams alloc] init];
 params.uid = self.uid;
 params.bizName = DEMO_BIZ;
 params.subBiz = DEMO_SUBBIZ;
 params.roomId = self.roomId;
 params.signature = DEMO_SIGNATURE;
 params.rtoken = self.rtoken;
 [_artvcEgnine joinRoom:params];
```

   - 加入房间成功，会有加入房间成功的回调以及房间已有成员的回调。

```
-(void)didJoinroomSuccess{
}
-(void)didParticepantsEntered:(NSArray<ARTVCParticipantInfo*>*)participants{
}
```

   - 加入房间失败，会有 error 回调。

```
//error.code == ARTVCErrorCodeProtocolErrorJoinRoomFailed
-(void)didEncounterError:(NSError *)error forFeed:(ARTVCFeed*)feed{
}
```

   - 后续其他人加入房间后，会有成员加入房间的回调。

```
-(void)didParticepantsEntered:(NSArray<ARTVCParticipantInfo*>*)participants{
}
```

创建或者加入房间成功后开始推流与拉流

默认是自动推流与拉流。

- 推流/拉流过程中，有如下相关状态回调。

```
-(void)didConnectionStatusChangedTo:(ARTVCConnectionStatus)status forFeed:(ARTVCFeed*
)feed{
  [self showToastWith:[NSString stringWithFormat:@"connection
status:%d\nfeed:%@",status,feed] duration:1.0];
  if((status == ARTVCConnectionStatusClosed)  && [feed.uid isEqualToString:[self uid]
]){
      [self.artvcEgnine stopCameraPreview];//音视频通话下，停止摄像头预览。
      [self.artvcEgnine leaveRoom];
  }
}
```

- ARTVCConnectionStatus 状态解释。

枚举	值	说明
ARTVCConnectionStatusConnecting	200	开始发布和订阅时，首先回调该状态。
ARTVCConnectionStatusConnected	202	发布/订阅成功时回调该状态。
ARTVCConnectionStatusDisConnected	203	出现闪断，底层媒体流断开，底层会做自动重连。业务拿到这个回调仅做用户提示，不需要做 leaveRoom 处理。
ARTVCConnectionStatusFailed	204	底层 ICE 失败无法继续，是一个终极错误。业务拿到这个回调可做停止摄像头 leaveRoom 处理。
ARTVCConnectionStatusClosed	206	每个发布和订阅结束时，回调该状态。它是最后的状态，业务拿到这个回调可做停止摄像头 leaveRoom 处理。

- 推流成功后，其他房间成员会收到新 feed 的回调。

```
-(void)didNewFeedAdded:(ARTVCFeed*)feed{
  [self showToastWith:[NSString stringWithFormat:@"new feed published by
others:%@",feed] duration:2.0];
}
```

- 自动订阅模式下，会主动订阅该 feed。
- 订阅成功后，房间其他成员会收到如下回调。

```
-(void)didSubscriber:(NSString*)subscriber subscribedAFeed:(ARTVCFeed*)feed{
  [self showToastWith:[NSString stringWithFormat:@"subscriber subscribed :%@",feed] d
uration:2.0];
}
```

通话结束

- 音视频通话停止摄像头并离开房间。

```
[_artvcEgnine stopCameraPreview];
[_artvcEgnine leaveRoom];
```

- 纯音频通话离开房间。

```
[_artvcEgnine leaveRoom];
```

- 成员离开房间后，房间其他成员会收到成员离开的回调。

```
-(void)didParticepant:(ARTVCParticipantInfo*)participant leaveRoomWithReason:(ARTVCPa
rticipantLeaveRoomReasonType)reason{
  [self showToastWith:[NSString stringWithFormat:@"participant left:%@
reason:%d",participant,reason] duration:2.0];
}
```

- 自动发布订阅模式下离开房间，会自动取消发布本地的流以及取消订阅曾订阅过的流。
- 取消发布后，房间其他成员会收到取消发布的回调。

```
-(void)didFeedRemoved:(ARTVCFeed*)feed{
  [self showToastWith:[NSString stringWithFormat:@"feed unpublished by
others:%@",feed] duration:2.0];
}
```

- 取消订阅后，房间其他成员会收到取消订阅的回调。

```
-(void)didSubscriber:(NSString*)subscriber unsubscribedAFeed:(ARTVCFeed*)feed{
  [self showToastWith:[NSString stringWithFormat:@"subscriber unsubscribed :%@",feed]
duration:2.0];
}
```

3.2.3. iOS 进阶功能

本文介绍的是音视频通话 API 在 iOS 中的进阶功能。

渲染 View 相关

实现 `ARTVCEngineDelegate` 中部分和渲染相关的回调接口。

1. 渲染 View 对象被创建。

```
//渲染 View 对象被创建，同时 feed 相关联，业务收到此回调时可将此 view 加入到布局中，//并设置其 fr
ame。
-(void)didVideoRenderViewInitialized:(UIView*)renderView forFeed:(ARTVCFeed*)feed{
}
```

2. 渲染首帧视频。

```
-(void)didFirstVideoFrameRendered:(UIView*)renderView forFeed:(ARTVCFeed*)feed{
}
```

3. 停止渲染视频。

```
-(void)didVideoViewRenderStopped:(UIView*)renderView forFeed:(ARTVCFeed*)feed{
}
```

统一错误处理

- 所有错误。通过统一的错误回调接口回调业务，业务根据不同的错误码，进行相应的错误处理。

```
-(void)didEncounterError:(NSError *)error forFeed:(ARTVCFeed*)feed{
  //业务根据不同错误码，进行错误处理。
}
```

- 错误定义如下：

```
typedef NS_ENUM(int,ARTVCErrorCode){
  /**
  bad parameters passed to API
  */
  ARTVCErrorCodeBadParameters                  = - 103,
  /**
   camera permission is denied by user
   without this permission,video call cann't be continued,please advise user enable c
amera permission in settings.
   */
  ARTVCErrorCodeCameraPermissionNOtAllowed     = -104,
  /**
  microphone permission is denied by user
  without this permission,video call cann't be continued,please advise user enable ca
mera permission in settings.
  */
  ARTVCErrorCodeMicrophonePermissionNOtAllowed = -105,
  /**
  timeout happened,publish/subscribe cann't be finishied successfully
  */
  ARTVCErrorCodeTimeout                        = -108,
  /**
   you has already published or subsrcibed a feed .
  you cann't publish or subsrcibe the same feed once again when it has NOt been unpub
lished or unsubscribed.
  */
  ARTVCErrorCodeAlreadyPublishedOrSubsrcibed   = -111,
  /**
   you has NOt published or subsrcibed the feed .so you cann't do unpublish or unsubs
cribe operation.
  */
  ARTVCErrorCodeFeedHasNOtBeenPublishedOrSubsrcibed   = -119,
  /**
  internal  webrtc-relative error when doing publish/subscribe,for example,setting sd
p failed,creating sdp failed,e.g.
  */
  ARTVCErrorCodeInternalError                  = -113,
```

```
  /**
   * current room has become invalid .most of all,it's because network's
down,heartbeat abNOrmal.
   * if you wanna continue,you MUST call createRoom again to get a new valid room.
   */
  ARTVCErrorCodeCurrentRoomHasBecomeInvalid    = -114,
  /**
  server error hanppened.CreateRoom request failed,it's a server internal error.
  */
  ARTVCErrorCodeProtocolErrorCreateRoomFailed  = -115,
  /**
  server error hanppened.JoinRoom request failed,it's a server internal error.maybe t
he room you joined has been became invalid yet.
  */
  ARTVCErrorCodeProtocolErrorJoinRoomFailed    = -116,
  /**
  server error hanppened.Publish request failed,it's a server internal error.
  */
  ARTVCErrorCodeProtocolErrorPublishFailed     = -117,
  /**
  server error hanppened.Subscribe request failed,it's a server internal error.maybe
the stream you subscribed has been unpunlished yet or some error else.
  */
  ARTVCErrorCodeProtocolErrorSubscribeFailed   = -118,
};
```

切换摄像头

切换摄像头代码如下：

```
[_artvcEgnie switchCamera];
```

mute 远端视频

mute 远端视频代码如下：

```
ARTVCFeed* feed = 要 mute 操作的远端 feed;
[_artvcEgnie muteRemoteVideo:YES forFeed:feed];
```

麦克风静音

麦克风静音代码如下：

```
[_artvcEgnie muteMicrophone:YES];
```

mute 远端音频

mute 远端音频代码如下：

```
ARTVCFeed* feed = 要 mute 操作的远端 feed;
[_artvcEgnie muteRemoteAudio:YES forFeed:feed];
```

听筒扬声器模式

- 切换模式：

```
[_artvcEgnie switchAudioPlayModeTo:ARTVCAudioPlayModeReceiver complete:nil];
```

- **变化通知：**

```
-(void)didAudioPlayModeChangedTo:(ARTVCAudioPlayMode)audioPlayMode{
  NSString *toast = nil;
  switch (audioPlayMode) {
      case ARTVCAudioPlayModeSpeaker:{
          toast = @"扬声器模式";
      }
          break;
      case ARTVCAudioPlayModeReceiver:{
          toast = @"听筒模式";
      }
          break;
      case ARTVCAudioPlayModeHeadphone:{
          toast = @"耳机模式";
      }
          break;
      case ARTVCAudioPlayModeBluetooth:{
          toast = @"蓝牙设备模式";
      }
          break;
      case ARTVCAudioPlayModeInit:{
          toast = @"未知模式";
      }
          break;
  }

  [self showToastWith:toast duration:2.0];
}
```

网络变化通知

网络变化通知代码如下，对于移动网络，业务可做弹窗提示用户有流量风险。

```
-(void)didNetworkChangedTo:(APMNetworkReachabilityStatus)netStatus{
    if(netStatus == APMNetReachabilityStatusReachableViaWiFi){
        return ;
    }
    [self showToastWith:[NSString stringWithFormat:@"网络切换到:%@",
[APMNetworkStatusManager stringOfNetworkStatus:netStatus]] duration:2.0];
}
```

带宽不足通知

带宽不足通知代码如下：

```
-(void)didAvailabeSendBandwidthBecomeLow:(BOOL)isLow currentBandwidth:(double)bw forFee
d:(ARTVCFeed*)feed{
    if(isLow){
        [self showToastWith:@"当前通话质量不佳" duration:2.0];
    }
}
```

截屏功能

截屏功能代码如下，可对任意流进行截屏。

```
ARTVCFeed* feed = 要截屏的 feed;
[_artvcEgnie snapshotForFeed:feed  complete:^(UIImage* image){
//根据需要对截屏图片进行处理
}];
```

获取通话质量的 Debug 信息

通过回调来返回任意流的 Debug 信息。

```
/**
 brief debug information is generated(including bitrate/cpu/codec,e.g.),you can show th
is on your debug information view.
 */
- (void)didBriefDebugInformationGenerated:(NSString*)debugInfo forFeed:
(ARTVCFeed*)feed{
//如果有需要，可以将 debugInfo 按流展示在 Debug 窗口。
}
```

获取通话质量的实时监控信息

- 通过回调可获取任意流的实时监控信息，包括码率、帧率、CPU 等。

```
-(void)didRealtimeStatisticGenerated:(ARTVCRealtimeStatisticSummary*)summary forFeed:
(ARTVCFeed*)feed{
}
```

- 返回的数据内容如下：

```
@interface ARTVCRealtimeStatisticSummary : NSObject
//connection stats googCandidatePair
/** 实际总发码率 单位 bps*/
@property(NOnatomic,copy) NSString* totalSendBitrate;
/** 实际总的收码率 单位 bps*/
@property(NOnatomic,copy) NSString* totalRecvBitrate;
/** 网络延迟（毫秒）*/
/** 网络延迟（毫秒）*/
@property(NOnatomic,copy) NSString* rtt;
//video send
/** 视频码率发 单位 bps*/
@property(NOnatomic,copy) NSString* videoSendBitrate;
/** 实际视频发帧率 */
@property(NOnatomic,copy) NSString* videoSendFps;
//video recv
/** 视频码率收 单位 bps*/
@property(NOnatomic,copy) NSString* videoRecvBitrate;
/** 实际视频收帧率 */
@property(NOnatomic,copy) NSString* videoRecvFps;
/** 声音码率发 单位 bps*/
@property(NOnatomic,copy) NSString* audioSendBitrate;
/** 声音码率收 单位 bps*/
@property(NOnatomic,copy) NSString* audioRecvBitrate;
/** 视频发送丢包率*/
@property(NOnatomic,copy) NSString* videoLossRate;
//audio send
/** 音频发送丢包率*/
@property(NOnatomic,copy) NSString* audioLossRate;
/** cpu */
@property(NOnatomic,copy) NSString* cpu;
@end
```

通话过程中动态调整编码分辨率和相机 FPS（仅使用于内置 camera）

- 当前分辨率以及 FPS 分别为 640x360、15 FPS 时，设置如下：

```
_artvcEgnine.videoProfileType = ARTVCVideoProfileType_640x360_15Fps;
```

- 假设，因业务需求需要将分辨率以及 FPS 临时调高至 960x540、30 FPS，设置如下：

```
_artvcEgnine.videoProfileType = ARTVCVideoProfileType_960x540_30Fps;
```

- 处理完特定业务需求后，恢复至 640x360，15 FPS。

```
_artvcEgnine.videoProfileType = ARTVCVideoProfileType_640x360_15Fps;
```

通话过程中动态调整编码分辨率（适应于所有 video source）

- 自定义推流中，当前分辨率为 640x360，若因业务需求需要临时调高至 960x540，设置如下：

```
[ _artvcEgnine changeVideoProfileTo:ARTVCVideoProfileType_960x540_15Fps
forVideoSource:ARTVCVideoSourceType_Custom];
```

- 内置 camera，当前分辨率为 640x360，若因业务需求需要临时调高至 960x540，设置如下：

```
[ _artvcEgnine changeVideoProfileTo:ARTVCVideoProfileType_960x540_30Fps
forVideoSource:ARTVCVideoSourceType_Camera];
```

自定义推视频流

自定义推视频流约束

- 只支持 NV12 格式的视频帧，以 CVPixelBufferRef 格式输入.
- 调用方目前需要保证 FPS，按照一定的 FPS 来输入。建议使用 15-24FPS。
- 只支持手动模式推流。

流程

自定义推视频流的流程如下：

```
//自定义推流中，必须设置为手动推流模式
[ _artvcEgnine setAutoPublish:NO];
 //创建一个自定义推流类
ARTVCCreateCustomVideoCaputurerParams* params = [[ARTVCCreateCustomVideoCaputurerParams
alloc] init];
//如果需要 SDK 帮忙渲染，需要设置 provideRenderView 为 YES，SDK 默认不帮业务渲染。
            params.provideRenderView = YES;


            self.customCapturer = [_artvcEgnine createCustomVideoCapturer:params];
            //按照一定的频率喂数据给 SDK,格式是 CVPixelBufferRef，只支持 NV12 格式。
            [self.customCapturer provideCustomVideoFramePeriodlyWith:CVPixelBufferRef];
//主动调用推流
ARTVCPublishConfig* config = [[ARTVCPublishConfig alloc] init];
            config.videoSource = ARTVCVideoSourceType_Custom;
            config.videoProfile = ARTVCVideoProfileType_640x360_15Fps;
            self.customPublishConfig = config;
            [_artvcEgnine publish:config];
```

屏幕共享

屏幕共享代码如下：

```
//必须设置为手动推流模式
[ _artvcEgnine setAutoPublish:NO];
 //开始屏幕共享
-(void)startScreenSharing{
    NSLog(@"start screen sharing");
    ARTVCCreateScreenCaputurerParams* screenParams = [[ARTVCCreateScreenCaputurerParams
alloc] init];
    screenParams.provideRenderView = YES;
    [_artvcEgnine startScreenCaptureWithParams:screenParams complete:^(NSError* error){
        NSLog(@"start screen sharing finish,error:%@",error);
        if(error){
             //这里的 code 见启动屏幕捕捉的错误码章节

        }else{
           //屏幕捕捉成功后再调用 publish 进行推流
            ARTVCPublishConfig* config = [[ARTVCPublishConfig alloc] init];
            config.videoSource = ARTVCVideoSourceType_Screen;
            config.audioEnable = NO;
            config.videoProfile = ARTVCVideoProfileType_1280x720_30Fps;
            [_artvcEgnine publish:config];
        }
    }];
}
//停止屏幕共享
-(void)stopScreenSharing{
    NSLog(@"stop screen sharing");
    //停止屏幕捕捉
    [_artvcEgnine stopScreenCapture];
    //取消发布屏幕共享流
    ARTVCUnpublishConfig* config = [[ARTVCUnpublishConfig alloc] init];
    config.feed = self.screenLocalFeed;
    [_artvcEgnine unpublish:config];
}
```

启动屏幕捕捉的错误码

启动屏幕捕捉的错误码如下：

```
/**
    screen capture alrady under running, you cann't start it again before you call stop
    */
    ARTVCErrorCodeScreenCapturerAlreadyUnderRunning            = -1011,
    /**
    starting screen capture failed
    */
    ARTVCErrorCodeStartScreenCaptureFailed                     = -1012,
    /**
    start screen capture success,but may be encounter errors during the processing of t
he capure operation.
    */
    ARTVCErrorCodeScreenCaptureFailedInProcessing              = -1013,
```

3.2.4. 代码示例

本文介绍的是音视频通话接入 iOS 的代码示例及其使用步骤。

下载代码示例

点击这里 获取音视频通话代码示例。

使用示例代码

1. 在示例工程中添加配置文件。
   i. 登录 mPaaS 控制台。
   ii. 创建 mPaaS 应用。
   iii. 下载 mPaaS 应用的 `.config` 格式 配置文件 放入音视频通话示例工程中。
   iv. 执行 `pod mpaas update 10.1.68 && pod install` 命令。
2. 创建通话应用 并获取 bizName。 在控制台左侧导航栏选择 **音视频通话** > **通话应用管理**，创建通话应用或进入已有的通话应用，获取通话应用的 bizName。
3. 在控制台 生成临时签名。 在控制台左侧导航栏选择 **音视频通话** > **签名校验**，在上方的 **生成临时签名** 区域，选择通话应用，输入 **UserID** 生成临时签名。

   > **说明**
   > - 更换通话应用或 UserID 后需要重新生成临时签名。
   > - 注意临时签名的有效期，过期后需重新生成。

4. 配置示例工程。 运行示例工程，在 App 设置页填写从以上步骤获取的用户 ID（userId）、业务名称（bizName）和临时签名（signature）。

   > **说明**
   > 为保障您的流量安全，业务上线后务必通过 安全加签 在服务端生成签名。

5. 发起或加入音视频通话。 您可在 App 中创建通话房间发起音视频通话，或者在页面上输入其他用户创建的视频通话房间号（roomId）和 Token 加入音视频通话。

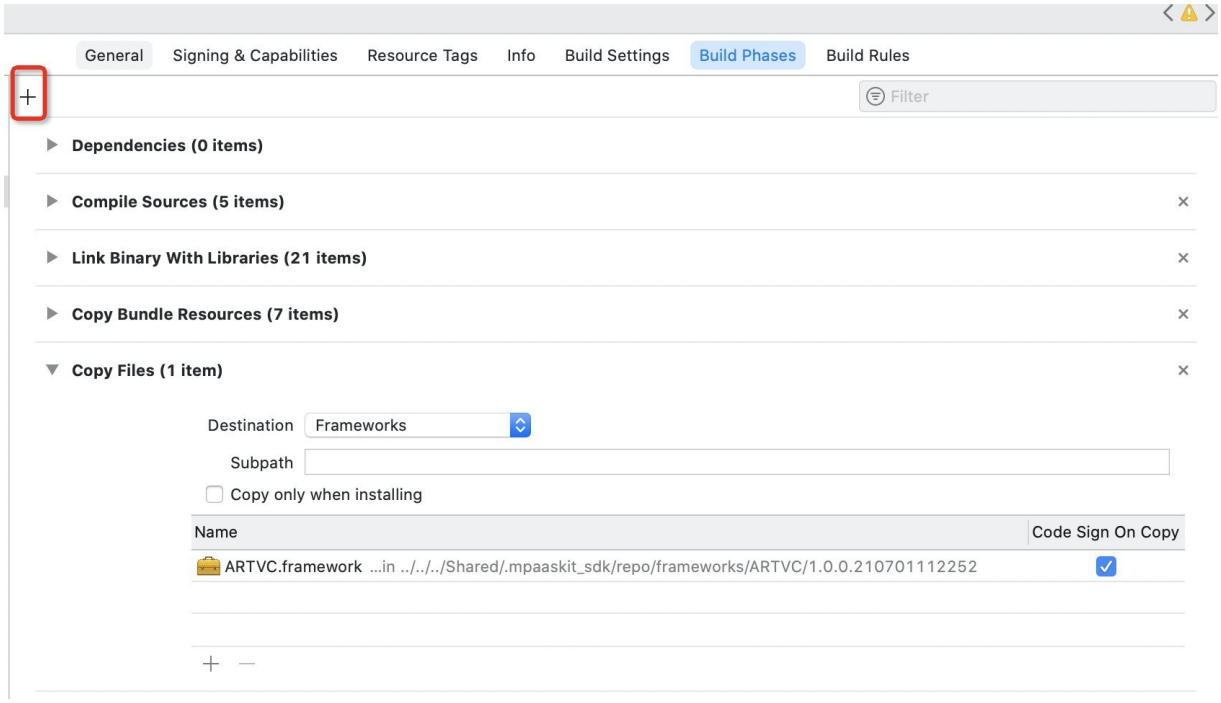
   > **说明**
   > 在同一个通话应用中（使用同一个 bizName）的不同端的 userId 必须互不相同。

3.2.5. 常见问题

本文介绍接入音视频 iOS SDK 的常见问题和解决方案。

集成音视频 SDK 后，工程编译时出现错误： dyld: Library not loaded: @rpath/ARTVC.framework/ARTVC..... Reason: image not found。

解决方案：在 Build Phases 菜单栏下，点击“+”，新建 Copy Files 文件夹，参考下图完成配置。



集成音视频 SDK 后，工程编译时出现错误：Building for iOS, but the linked and embedded framework artvc.framework was built for iOS + iOS Simulator。

解决方案：选择 TARGETS > Build Settings，在 Build Options 下，将 Validate Workspace 设置为 YES。

3.3. 接入 Web

本文介绍接入音视频通话到 Web 端的操作步骤。

前置条件

开发环境要求如下：

支持平台	浏览器	浏览器版本
Mac	Chrome	不低于 67
Windows	Chrome	不低于 67

> **说明**
> - 集成 Web 端 SDK 必须使用 HTTPS 协议。
> - 建议将 SDK 升级至最新版本 1.5.0，单击此处 可下载最新版 SDK。

操作步骤

1. 下载 `artvc-web-sdk` 。

2. 将 `artvc-web-sdk` 程序包保存到本地项目下。
3. 对照下表引入项目所需要的依赖文件。

依赖类型	依赖文件	说明
强依赖	◦ mcu.js ◦ meeting_api.js ◦ adapter.js	-
可选依赖	remote_record.js	使用服务端录制
	◦ client_record.js ◦ RecordRTC.min.js（`https://gw.alipayobjects.com/os/lib/recordrtc/5.5.9/RecordRTC.min.js`） ◦ EBML.js	使用浏览器录制
	meeting_camera_stream.js	使用摄像头或麦克风能力
	meeting_desk_stream.js	桌面屏幕共享功能 **重要** 如果您需要使用桌屏幕共享功能，请将 Chrome 升级到 72 版本以上。
	◦ meeting_file_stream.js ◦ pdf.js ◦ pdf.worker.js（必须和 `meeting_api.js` 处在 `./lib/pdf.worker.js` 相对位置）	使用文件投屏共享。如果文件是 PDF 格式，需要额外引入 pdf.js 和pdf.worker.js 文件。
	meeting_html_stream.js	使用自定义区域投屏共享
	meeting_im.js	使用 IM 能力
	meeting_invite.js	使用邀请能力，只支持邀请在线座席。

依赖类型	依赖文件	说明
	◦ dialogue.css ◦ getMediaInfo.js ◦ jquery.min.js（`https://gw.alipayobjects.com/os/lib/jquery/3.5.1/dist/jquery.min.js`） ◦ iconfont.js ◦ eruda.js	Demo 相关依赖在实际使用时无需引入。

4. 添加如下代码，实例化 SDK。

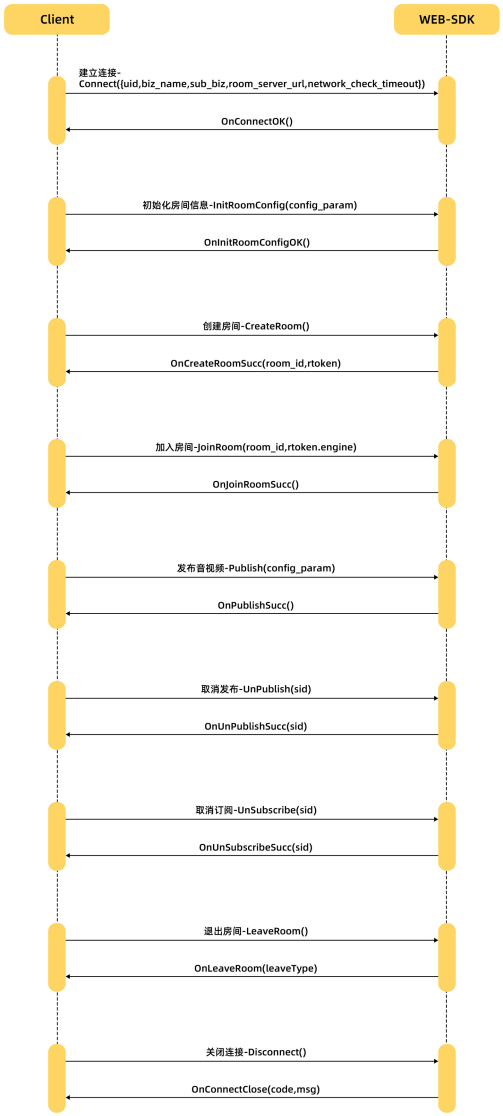
```
let test_controller = new McuController();  // 实例化 SDK
```

接口说明

整个 API 封装在 `McuController` 的类中，接口分为以下两类：

- 主调接口：供业务主动调用。
- 回调接口：主调接口的反馈回调，作为业务上的通知。

使用流程



3.4. 接入 Linux

3.4.1. 快速开始

本文主要提供 Linux 端音视频通话服务的接入指引。

系统版本要求

OS	GCC	CPU	libc 版本	libstdc++ 版本
CentOS 7.5	5.4.0 以上	X86_64	不低于 2.23	不低于 GLIBCXX_3.4.21
Ubuntu 16.04	5.4.0 以上	X86_64	不低于 2.23	不低于 GLIBCXX_3.4.21

为了顺利接入，请您保证 OS 与 GCC 版本一 一对应。另外 UID 也需要与签名一 一对应，应避免混用。

集成 Linux 客户端 SDK 必须使用 HTTPS 协议。点击这里 下载最新版 SDK。

操作步骤

开通音视频通话服务

1. 替换示例工程中的配置文件。
   i. 登录 mPaaS 控制台。
   ii. 创建 mPaaS 应用。
   iii. 下载 mPaaS 应用的 `.config` 格式 配置文件 放入音视频通话示例工程的 `app/` 目录下，用以替换已有的配置文件。
2. 创建通话应用 并获取 bizName。在控制台左侧导航栏选择 **音视频通话** > **通话应用管理**，创建通话应用或进入已有的通话应用，获取通话应用的 bizName。
3. 控制台 生成临时签名。在控制台左侧导航栏选择 **音视频通话** > **签名校验**，在上方的 **生成临时签名** 区域，选择通话应用，输入 **userId** 生成临时签名。

   > **说明**
   > - 更换通话应用或 userId 后需要重新生成临时签名。
   > - 注意临时签名的有效期，过期后需重新生成。

4. 配置示例工程。运行示例工程，在浏览器设置页填写从以上步骤获取的用户 ID（userId）、业务名称（bizName）和临时签名（signature）。

   > **说明**
   > 为保障您的流量安全，业务上线后务必 通过服务端生成签名 。

5. 发起或加入音视频通话。您可在浏览器中创建通话房间发起音视频通话，或者在页面上输入其他用户创建的视频通话房间号（roomId）和 token 加入音视频通话。

   > **说明**
   > 在同一个通话应用中（使用同一个 bizName）的不同端的 userId 必须互不相同。

添加 SDK

1. 下载 MRTC-Linux-SDK 程序包并解压到本地项目文件夹下。
2. 引入 SDK 中的头文件以及相关 lib 到工程文件中。
3. 检查相应配置是否完成，编译运行程序。

后续步骤

将音视频 SDK 接入 Linux 端后，您可以实现音视频通话的功能，详情请参见 Linux SDK API 说明。

3.4.2. 使用 Linux SDK

本文介绍的是接入 Linux 端音视频通话的主要流程。

初始化引擎

指定音视频输入输出格式。

```
//音频方面，当前支持输入输出都是 16k 采样率的单声道 PCM，同时也支持 48k 双声道的输入输出
//视频方面，当前支持 I420P YUV 和 H264 格式的数据输入和输出
//以下示例为指定了视频输入输出都是 YUV 格式，分辨率为 640x360，指定 YUV 编码为 H264 后发送
//同时指定编码帧率为 24fps，编码码率为 600kbps
RtcVideoFormat video;
video.fps = 24;
video.width = 640;
video.height = 360;
video.kbps = 600;
video.videoCodec = VIDEO_CODEC_YUV420P;
video.yuvCodec = VIDEO_CODEC_H264;

//音频格式默认为单声道 16k 采样率
RtcAudioFormat audio;
audio.bytesPerSample = 2;
audio.channels = 1;
audio.sampleRate = 16000;
audio.audioCodec = AUDIO_CODEC_PCM;

//对接开发环境的房间服务器
//业务正式上线需要修改为对接线上环境 wss://artvcroom.alipay.com/ws
MRtcEngineInitParam param;
param.roomUrl = "wss://artvcroomdev.dl.alipaydev.com/ws";
param.audioFormat = audio;
param.videoFormat = video;
param.logLevel = "info";

//初始化引擎，需要同时指定引擎事件监听器
MRtcEngine* engine = MRtcEngine::Create(this);
engine->Init(param);
```

响应引擎事件

等待引擎初始化成功。

```
void Demo::OnEngineEvent(RtcEvent event) {
    if (event.type == INIT) {
        if (event.code == 0) {
            initOK = true;
        } else {
            std::cout<<"Engine Init Fail,code:"<<event.code<<std::endl;
        }
    } else if(event.type == WSS_LINK) {    //通话过程中与房间服务器的信令通道断连或者重新连接成功的通知
        if (event.code == 0) {
            std::cout<<"WSS_LINK OK!"<<std::endl;
        } else {
            std::cout<<"WSS_LINK Fail,code:"<<event.code<<std::endl;
        }
    }
}
```

创建会话

每个会话代表一次音视频通话过程。

```
//创建 session，同时需要指定 session 事件监听器
MRtcSession* session = engine->CreateSession(this);
```

创建通用的房间或者加入已经存在的房间

```
//创建房间的时候，需要指定业务相关的信息，此类信息需到 mPaaS 平台自助申请获取
//创建房间的时候，还能同时指定启动录制或者关闭录制
CreatRoomParam param;
param.autoSubscribe = true;
param.bizName = "demo";
param.subBiz = "default";
param.sign = "signature";
param.uid = "uidxxx";
param.workspaceId = "default";
param.ext = R"({"defaultRecord":false,"recordStrongDepend":false})";
session->CreateRoom(param);

or

JoinRoomParam param;
param.autoSubscribe = true;
param.bizName = "demo";
param.subBiz = "default";
param.sign = "signature";
param.token = "token";
param.roomId = "roomxxx;
param.uid = "uidxxx";
param.workspaceId = "default";
session->JoinRoom(param);
```

监听会话事件

等待底层数据链路打通（ICE_LINK）。

```
    void LinuxSession::OnSessionEvent(RtcEvent event) {
        if (event.type == CREATE_ROOM) {
            if (event.code == 0) {
                std::cout<<"CREATE_ROOM OK,roomId and token:" << event.ext <<std::endl;
                if(doPub && !isP2P) {
                    PublishParam param;
                    param.enableAudio = true;
                    param.enableVideo = true;
                    std::cout<<"Do publish" << std::endl;
                    session->Publish(param);
                }
               StartRecord();
            } else {
                std::cout<<"CREATE_ROOM FAIL,code:"<<event.code<<std::endl;
            }
        } else if (event.type == JOIN_ROOM) {
            if (event.code == 0) {
                std::cout<<"JOIN_ROOM OK,Begin Do publish"<<std::endl;
```

```
                if(doPub) {
                    PublishParam param;
                    param.enableAudio = enableAudio;
                    param.enableVideo = enableVideo;
                    session->Publish(param);
                }
                StartRecord();
            } else {
                std::cout<<"JOIN_ROOM FAIL,code:"<<event.code<<std::endl;
            }
        } else if (event.type == PUBLISH) {
            if (event.code == 0) {
                localPubStreamId = event.ext;
                if(sourceConfig.videoType != AV_FILE_H264) {
                    std::string pcmFileName = sourceConfig.audioFile;
                    pcmFileReader.reset(new PCMFile());
                    pcmFileReader-
>Open(pcmFileName,sourceConfig.audioSampleRate,sourceConfig.audioChannels,"rb");
                    std::string yuvFileName = sourceConfig.videoFile;
                    yuvFileReader.reset(new YuvFile());
                    yuvFileReader-
>Open(yuvFileName,sourceConfig.yuvWidth,sourceConfig.yuvHeight,"rb");
                }
                std::cout<<"PUBLISH OK"<<std::endl;
            } else {
                std::cout<<"PUBLISH FAIL,code:"<<event.code<<std::endl;
            }
        } else if (event.type == SUBSCRIBE) {
            if (event.code == 0) {
                std::string pcmFileName = event.ext + ".pcm";
                pcmFileWriter.reset(new PCMFile());
                pcmFileWriter->Open(pcmFileName,16000,1,"wb+");
                if(sourceConfig.videoType == AV_FILE_YUV) {
                    std::string yuvFileName = event.ext + ".yuv";
                    yuvFileWriter.reset(new YuvFile());
                    yuvFileWriter->Open(yuvFileName,"wb+");
                } else if(sourceConfig.videoType == AV_FILE_H264) {
                    std::string h264FileName = event.ext + ".264";
                    h264File = fopen(h264FileName.c_str(),"wb+");
                }
                std::cout<<"SUBSCRIBE OK,Begin to recv data"<<std::endl;
            } else {
                std::cout<<"SUBSCRIBE FAIL,code:"<<event.code<<std::endl;
            }
        } else if (event.type == ICE_LINK) {
            if(event.code == 0) {
                if(localPubStreamId == event.ext) {
                     std::cout<<"ICE OK,Begin to send data"<<std::endl;
                     if(sourceConfig.videoType != AV_FILE_H264) {
                        if(audioThread == 0 && enableAudio) {
                            pthread_create((pthread_t*)&audioThread,(const
pthread_attr_t*)NULL,PcmThread,(void*)this);
                        }
                     }
```

```
                        if(videoThread == 0 && enableVideo) {
                            if(sourceConfig.videoType == AV_FILE_YUV) {
                                pthread_create((pthread_t*)&videoThread,(const
pthread_attr_t*)NULL,YuvThread,(void*)this);
                            } else if(sourceConfig.videoType == AV_FILE_H264) {
                                pthread_create((pthread_t*)&videoThread,(const
pthread_attr_t*)NULL,H264Thread,(void*)this);
                            }
                        }
                    }
                } else {
                    std::cout<<"ICE_LINK FAIL,code:"<<event.code<<std::endl;
                }
            } else if (event.type == EXIT_ROOM) {
                if(session) {
                    engine->DestroySession(session);
                    session = nullptr;
                }
                if (event.code == 0) {
                    std::cout<<"EXIT_ROOM OK"<<std::endl;
                } else {
                    std::cout<<"EXIT_ROOM FAIL,code:"<<event.code<<std::endl;
                }
            } else if (event.type == START_RECORD) {
                if (event.code == 0) {
                    std::cout<<"Start record OK"<<std::endl;
                } else {
                    std::cout<<"Start record FAIL"<<std::endl;
                }
            }
        }
```

ICE_LINK 完成后发送音视频数据

```
//对于接入方，建议按照视频帧率来发送视频数据，按照音频采样周期来发送音频数据
//如果视频帧率为 25 帧，则每 1/25 秒发送一次视频数据
//如果音频采样周期为 10ms，那么每 10ms 发送一次音频数据
  - 如果是 TTS 场景，也可以一次性发送某短文本的音频数据。SDK 内部有音频缓存，则会先缓存，然后再慢慢按
照 10ms 的间隔发送音频数据到对端
  - 如果是用手机麦克风来采集音频的场景，这种场景不走 TTS，手机采集声音按照 10ms 的间隔输入 PCM 数据
session->FeedAudio(audio);
session->FeedVideo(video);
```

ICE_LINK 完成后接收音视频数据

```
listener->OnAudio(audio,streamId)
listener->OnVudio(audio,streamId)
```

通话结束后销毁会话

```
//引擎只需要创建一次，整个业务应用不退出就无需销毁引擎
engine->DestroySession(session);

//如果整个业务应用退出，则销毁引擎
engine->Destory();
```

3.4.3. Demo 示例

本文提供 Demo 示例以供您参考使用。该示例借助 SDK 模拟了个 Linux 的客户端。

它的音视频输入来自文件，比如单独的 audio.pcm 和 video.yuv 或者音视频整合在起的 video.mp4。如果是 MP4 文件，那么 Demo 会利用 FFmpeg 解码其中的音频为 PCM，视频不解码，只解封装为 H264-NALU，然后再根据 MP4 文件中的视频帧率，按照这个频率输入音视频到 SDK 中。

点击这里 获取 Demo 示例。

如何体验

上的 Demo 只是个 Linux 的模拟端。要体验 2 人音视频通话，还需要 Web 端的协助，您可以在 Web 端上创建房间并发布好音视频，然后启动 Linux 端加这个房间，这样在 Web 端就能看到 Linux 端的音视频画了。

以下演示为一个浏览器端和一个 Linux 端音视频通话的场景。请使用 Chrome 浏览器打开 Web 端的地址。

如上图所示，您可以通过以下简单的 3 步就能在 Web 端创建一个房间。

1. 在 **房间服务器** 列表中，选择开发环境。
2. 配置好相应的 `bizName` 、 `uid` 等，单击 **连接** 即可连接到房间服务器。
3. 单击 **创建房间**。

房间创建成功后， `room_id` 框会显示房间的 ID。

然后启动 Linux-Demo 端，使用命令 `./mrtcdemo -r xxx` （xxx 代表的是房间 ID）加入这个房间。

房间密码（ rtoken）般是 123，SDK 是默认的，可以不用填写，如果是其他值，可以用 `-t` 指定。

3.4.4. 常见问题

本文列举一些接入过程中的常见问题及其解决方案。

如何获取房间 ID 和录制 ID

房间 ID 统由 MRTC 后台房间管理系统 RoomServer 产，端创建房间时，RoomServer 会产生个房间 ID，以及录制 ID 给到端（如果创建房间的时候指定需要录制）。 `OnSessionEvent（RtcEvent event）` 接口中，在 `event.type == CREATE_ROOM && event.code == SUCCESS` 的情况下， event.ext（JSON 字符串）中存放了房间 ID、房间密码以及录制 ID。例如： `event.ext = {"recordId":"xxx","roomId:"yyy","token":"zzz"}` 。

如何指定录制

如果业务需要录制，可以通过以下两种方式来指定录制。

- 方法一：创建房间时， `CreatRoomParam` 里面的 ext（JSON 字符串）字段可以用来指定录制，比如 `param.ext = R"({"defaultRecord":true,"recordStrongDepend":false})"` 。其中 `defaultRecord` 指定是否开启录制， `recordStrongDepend` 指定房间的创建成功是否要考虑录制服务的状态，假设房间服务器在创建房间的时候，发现录制服务器异常，而如果这里指定了录制强相关的话，创建房间会失败，否则房间还是会创建成功，只不过这个房间没有启动录制。
- 方法二：创建房间时，默认不启动录制，等到对端加入房间以后，再调用 `StartRecord` 接口启动录制。这样做的好处是可以避免录制资源的浪费，假设对方拒绝加入房间，那么录制服务就不会提前启动。

如何单独指定音频或者视频

发布或者订阅参数 `PublishParam/SubscribeParam` 里面的 `enableVideo/enableAudio` 可以指定是否发布音频和视频与是否订阅音频和视频。

如何开启 P2P 模式

P2P 模式指的是客户端和 Linux 之间的媒体数据不经过服务器中转，直接点到点打通。如果业务不需要录制，同时 Linux 端的部署满足全国各地用户的就近接入，可以考虑开启 P2P 模式。追求极致的时延体验，开启 P2P 模式需要注意以下两点：

- 在创建房间或者加入房间时，设置 `param.engine = ENGINE_P2P` 。
- 收到对方加入房间的事件通知（OnNewJoiner）以后，再做 Publish 操作。

如何传递文本消息

Linux-SDK 提供了两种文本消息的传递模式：

- P2P 场景：推荐 DataChannel 传递，调用 Session -> SendData 方法（高频消息，能保证按顺序收发）。
- 其他场景：使用 Room 中转的模式，调用 Session -> SendText 方法（低频消息可能会乱序）。

3.5. 接入支付宝小程序

3.5.1. 快速开始

实时音视频通话插件通过集成 阿里云 mPaaS 音视频通话服务，可以在支付宝小程序之间，以及支付宝小程序与其他应用之间实现一对一和多对多的实时音视频通话功能。本文提供了将实时音视频通话插件接入支付宝小程序的指引。

前置条件

1. 订购 **实时音视频通话插件** 和关联小程序。

   在接入插件之前，确保本账号或者本账号所属的主账号已经订购了本插件，并且已经关联到需要使用本插件的小程序。点击这里 查看获取插件的详细步骤。

2. 配置主体小程序项目。

   在 小程序开发工具 里面打开需要使用插件的项目，然后配置 app.json。请修改以下文件中的“thePlugin”名称，与使用的插件代码进行匹配。

```
{
  "plugins": {
    "thePlugin": {
      "version": "*", // 目前只支持设置 * 拉取当前上架的最新版本
      "provider": "2021002126663572"
    }
  }
}
```

操作步骤

开通音视频通话服务

音视频通话通过阿里云 mPaaS 提供服务，请使用阿里云账号登录 mPaaS 控制台开通音视频通话并获取小程序接入需要的 `bizName` 、 `subBiz` 、 `worksapceId` 、密钥等参数。具体流程如下：

1. 登录 阿里云 mPaaS 控制台 ，点击 **+ 创建应用**，输入应用名并创建 mPaaS 应用。
2. 进入 mPaaS 应用，在左侧导航栏中选择 **音视频通话** > **通话应用管理** 进入音视频通话产品页。
3. 创建通话应用。创建后可查看 `bizName` 和密钥，相同 `bizName` 下的用户可互相通话。
4. 查看 `subBiz` 、 `workspaceId` 。
   i. mPaaS 应用主页面左侧导航栏中选择 **总览** > **快速下载代码配置**，进入代码配置页。

ii. 在代码配置页中查看 App ID、Workspace ID。分别对应小程序接入所需要的 `subBiz` 、 `workspaceId` 参数。

5. 获取签名（signature）。音视频通话中提供两种签名生成方法：
   - 服务端生成：通过安全加签生成签名。
   - 控制台生成：在控制台中生成临时签名，可用于临时体验视频通话产品。
6. 将以上步骤中获取的 `bizName` 、 `subBiz` 、 `workspaceId` 、 `signature` 等参数设置到 config 中即可开始音视频通话。
7. 如需小程序用户与其他终端用户进行音视频通话，其他终端的接入方式参考以下文档：
   - 接入Android
   - 接入iOS
   - 接入Web
   - 接入Linux
8. 如需了解更多关于接入支付宝小程序的信息，参见 API 概述。

3.5.2. 集成插件

本文介绍如何集成支付宝小程序端的实时音视频通话插件。

插件支持 3 种集成模式：

- 全屏：调用时，插件自动跳转视频通话全屏页。
- Flex：视频通话界面可以嵌入到自己的小程序的指定区域中。
- Answer：可以将视频通话界面嵌入到自己的小程序指定区域中。可以自定义覆盖已有的 UI 样式，不可以与 Flex 模式同时使用。

> **说明**
>
> - 配置上如果同时禁止摄像头和语音，会订阅失败，客户端目前逻辑判断中必须要有一个为 true。
> - 组件在引入的时候会做权限引导（麦克风、摄像头），但也提供方法供您在使用组件前做权限判断。

全屏模式（通过 JS API 调用）

示例代码

```
const { rtc } = requirePlugin('rtc')

// 视频通话区分两个角色
// - 视频发起人：发起视频通过的人
// - 加入的人员：要加入通话的人

// 流程
// - 1、发起，调用 start(config) ，得到 roomId 和 token
// - 2、将 roomId  和 token  通过 PUSH、短信等渠道给到需要加入的人员
// - 3、其他人唤起插件，调用 join(roomId, config) 加入到通话中来

// 发起人调用这个 API
rtc.start(config).then(roomInfo => {
  const { roomId, token } = roomInfo
  // 这里拿到创建的 roomId 和 token ，用于后续其他人加入使用
})

// =======================

// 加入的人调用这个 API
rtc.join(roomId, config) // config 是需要的参数，参见
重要参数
。
```

调用流程

视频通话需要区分两个角色。一是视频发起人，二是要加入视频通话的人。

- 视频发起人发起视频通话，调用 `start(config)` ，得到 roomId 和 token。
- 将 roomId 和 token 通过 PUSH、短信等渠道给到需要加入的人员。
- 其他人唤起插件，调用 `join(roomId, config)` 加入到通话中来。

API 接口

```
const { rtc, EventType } = requirePlugin('rtc')
```

- 发起视频通话 `start (config: Config): Promise<RoomInfo>` 。
- 加入视频通话 `join (roomId: string, config: Config): Promise<RoomInfo>` 。
- 监听通话事件 `on(type: EventType, handler: data => void)` 。

获取 RoomInfo 的方法

视频通话由一方发起，发起后得到 RoomInfo，用于后续其他人加入，获取 RoomInfo 有以下两种方式（选择其中一种即可）。

- 通过 API `join` ，调用后返回 `Promise<RoomInfo>` 。

  ```
  rtc.start(config).then(roomInfo => {
    const { roomId, token } = roomInfo
    // 这里拿到创建的 roomId 和 token，用于后续其他人加入使用
  })
  ```

- 通过 `ROOM_CREATED` 事件得到。

```
rtc.on(EventType.ROOM_CREATED, data => {
  // 这里拿到发起人创建的 roomId 和 token，用于后续其他人加入使用
  const { roomId, token } = data
})
```

Flex 模式（通过组件调用）

引入组件

```
{
  "usingComponents": {
    "rtc": "plugin://rtc/flex"
  }
}
```

使用组件

```
<rtc
  config="{{ config }}"
  currentUserId="{{ currentUserId }}"
  onPlayerChange="handlePlayerChange"
  onUserChange="handleUserChange"
  onHangup="handleHangup"
  onError="handleError"
/>
```

- `config` 对象参见 重要参数 说明。
- `currentUserId` 当前播放的内容可以通过这个参数传入 `userId` 进行切换。
- `onPlayerChange(userId: string, preUserId: string): void` 用户播放界面切换事件。
- `onUserChange(state: UserState): void` 用户变化事件。
- `onError(error: Error): void` 通用报错。
- `onHangup(): void` 挂断事件。

```
interface UserState {
  joined: string[]  // 加入的用户列表
  exited: string[]  // 退出的用户列表
  users: string[]   // 当前房间内的用户列表
  current?: string  // 当前播放内容的用户 ID
}
```

API 接口

```
const { rtc, EventType } = requirePlugin('rtc')
```

- 发起视频通话 `start (config: Config): Promise<RoomInfo>` 。
- 加入视频通话 `join (roomId: string, config: Config): Promise<RoomInfo>` 。
- 监听通话事件 `on(type: EventType, handler: data => void)` 。

- 邀请 `invite (userId: string, data?: InviteInfo): Promise<Result>` 。

Config 对象参见 重要参数 说明。

- `InviteInfo`

以下邀请接口中可选（?）的参数，用于描述被邀请人展示的内容。

```
interface InviteInfo {
  nickName?: string // 邀请人昵称
  page?: string // 小程序的落地页
  inviteType?: InviteType // 被邀请人类型 (0: web, 1: app), 默认为 1
  roomType?: RoomType // 房间类型 (1: 音视频通话, 2: 音频通话), 默认为 1
  params?: Record<string, unknown> | null
}
```

Answer（通过组件调用）

引入组件

```
{
  "usingComponents": {
    "answer": "plugin://rtc/answer"
  }
}
```

使用组件

```
<answer
  screenname="{{screenname}}" // 覆盖 screen 的 class name, 默认'screen'
  namespace="{{namespace}}" // UI 样式 class name, 默认"answer"
  config="{{ config }}"
  currentUserId="{{ currentUserId }}"
  onPlayerChange="handlePlayerChange"
  onUserChange="handleUserChange"
  onHangup="handleHangup"
  onError="handleError"
/>
```

```
需要覆盖的 CSS
.screen {
  position: relative;
  background: rgba(31,33,41,0.95);
}
.screen.full {
  height: 100vh;
}
.screen.full .content {
  height: 100vh;
}
.screen.flex {
  height: 100%;
}
.screen.flex .content {
```

```
  height: 100%;
}
.screen .content {
  position: relative;
}
.screen .actions {
  position: absolute;
  bottom: 100rpx;
  height: 260rpx;
  width: 100%;
  display: flex;
}
.screen .actions .action {
  flex: 1;
  height: 100%;
  position: relative;
}
.answer .pusher-view,
.answer .player-view {
  width: 100%;
  height: 100%;
}
.answer .pusher-view.mini,
.answer .player-view.mini {
  position: absolute;
  top: 168rpx;
  right: 4px;
  width: 240rpx;
  height: 426rpx;
  z-index: 1;
}
.answer .pusher-view.hidden,
.answer .player-view.hidden {
  width: 0;
  z-index: -999;
}
.answer .pusher-view .message,
.answer .player-view .message {
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  width: 100%;
  text-align: center;
  display: flex;
  align-items: center;
  justify-content: center;
  background: rgba(31,33,41,0.95);
  color: #fff;
}
.answer .live-player,
.answer .live-pusher {
  width: 100%;
  height: 100%;
```

```
}
.answer .user-list {
  white-space: nowrap;
  position: absolute;
  display: flex;
  bottom: 360rpx;
  height: 180rpx;
  width: 100%;
}
.answer .user-list-item {
  width: 160rpx;
  height: 160rpx;
  margin: 0 10rpx;
  background: #333;
  overflow: hidden;
  flex-shrink: 0;
  flex-grow: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  box-sizing: border-box;
  padding: 8px;
}
.answer .user-list-item:first-child {
  margin-left: 25rpx;
}
.answer .user-list-item:last-child {
  margin-right: 25rpx;
}
.answer .user-list-item-text {
  color: #fff;
  word-wrap: break-word;
}
.answer .user-list-item.selected .user-list-item-text {
  color: #1677ff;
}
.answer .btn {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin: auto;
  width: 106rpx;
  height: 106rpx;
  line-height: 106rpx;
  border-radius: 50%;
  text-align: center;
  color: #fff;
  background: #262a32;
}
.answer .btn .icon {
  width: 50%;
  height: 50%;
  margin: 25% auto;
```

```
  margin: 25% auto;
}
.answer .btn-mute .icon {
  background: url("xxxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-mute .icon.disable {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-speaker .icon {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-camera .icon {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-camera .icon.disable {
  background: url("xx") center center no-repeat;
  background-size: contain;
}
.answer .btn-hangup {
  background: #fa4b4b;
}
.answer .btn-hangup .icon {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-switch-camera .icon {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
.answer .btn-screenshot .icon {
  background: url("xxx") center center no-repeat;
  background-size: contain;
}
```

API 接口

```
const { rtc, EventType } = requirePlugin('rtc')
```

- **发起视频通话** `start (config: Config): Promise<RoomInfo>` 。
- **加入视频通话** `join (roomId: string, config: Config): Promise<RoomInfo>` 。
- **监听通话事件** `on(type: EventType, handler: data =>void)` 。
- **邀请** `invite (userId: string, data?: InviteInfo): Promise<Result>` 。
- **权限引导** `async authCheck()` 。
- **挂断通话** `hangup(roomId?:string) :void` 。

Config 对象参见 重要参数 说明。

3.5.3. 组件功能

本文介绍了组件事件和与组件事件相关接口的功能。

音视频通话组件

组件标签：rtc

组件描述：用于 Flex 模式的音视频通话接入。

组件属性

属性名称	参数类型	是否必填	默认值	说明
config	Object	否	无	Config 对象。
currentUserId	String	否	无	当前用户的 uid，当前播放的内容可以通过这个参数传入 UserId 进行切换。

组件事件

事件名称	事件描述
onPlayerChange	用户播放界面切换事件。
onUserChange	用户变化事件。
onHangup	挂断事件。
onError	通用报错。

引入组件

```
{
  "usingComponents": {
    "rtc": "plugin://rtc/flex"
  }
}
```

使用组件

```
<rtc
  config="{{ config }}"
  currentUserId="{{ currentUserId }}"
  onPlayerChange="handlePlayerChange"
  onUserChange="handleUserChange"
  onHangup="handleHangup"
  onError="handleError"
/>
```

参数说明

- `config` 参见 重要参数 说明。
- `currentUserId` 当前播放的内容可以通过这个参数传入 userId 进行切换。
- `onPlayerChange(userId: string, preUserId: string): void` 用户播放界面切换事件。
- `onUserChange(state: UserState): void` 用户变化事件。
- `onError(error: Error): void` 通用报错。
- `onHangup(): void` 挂断事件。

UserState 对象

```
interface UserState {
  joined: string[]  // 加入的用户列表
  exited: string[]  // 退出的用户列表
  users: string[]   // 当前房间内的用户列表
  current?: string  // 当前播放内容的用户 ID
}
```

Config 参见 重要参数 说明。

接口列表

invite

接口描述：邀请用户加入通话。入参 如下表所示。

参数名称	参数类型	是否必填	默认值	说明
userId	String	否	无	用户 uid

参数名称	参数类型	是否必填	默认值	描述
inviteInfo	Object	否	无	InviteInfo 对象

InviteInfo 对象：邀请接口可选的参数，用于描述被邀请人展示的内容。

```
interface InviteInfo {
  nickName?: string // 邀请人昵称
  page?: string // 小程序的落地页
  inviteType?: InviteType // 被邀请人类型（0: web, 1: app），默认为 1
  roomType?: RoomType // 房间类型（1: 音视频通话, 2: 音频通话），默认为 1
  params?: Record<string, unknown> | null
}
```

join

接口描述：加入视频通话。入参 如下表所示。

参数名称	参数类型	是否必填	默认值	说明
roomId	String	是	无	无

参数名称	参数类型	是否必填	默认值	说明
config	Object	否	无	Config 对象，参见 重要参数。

出参

参数名称	参数类型	说明
RoomInfo	Object	Promise

```
const { rtc } = requirePlugin('rtc')

// 加入的人调用这个 API
rtc.join(roomId, config) // config 是需要的参数，参见
重要参数
。
```

获取 RoomInfo

视频通话由一方发起，发起后得到 RoomInfo，用于后续其他人加入，获取 RoomInfo 有两种方式（选择其中一种即可）。

1. 通过 api `join` ，调用后返回 `Promise<RoomInfo>` 。

```
rtc.start(config).then(roomInfo => {
  const { roomId, token } = roomInfo
  // 这里拿到创建的 roomId 和 token，用于后续其他人加入使用
})
```

2. 通过 `ROOM_CREATED` 事件得到。

```
rtc.on(EventType.ROOM_CREATED, data => {
  // 这里拿到发起人创建的 roomId 和 token，用于后续其他人加入使用
  const { roomId, token } = data
})
```

Config 对象参见 重要参数 说明。

on

接口描述：监听通话事件。入参 如下表所示。

参数名称	参数类型	是否必填	默认值	说明
type	Object	否	无	事件类型

参数名称	参数类型	是否必填	默认值	说明
handler	Any	否	无	回调函数

```
const { rtc, EventType } = requirePlugin('rtc')

rtc.on(EventType.ROOM_CREATED, data => {
  // 这里拿到发起人创建的 roomId 和 token，用于后续其他人加入使用
  const { roomId, token } = data
})
```

start

接口描述：发起视频通话。入参 如下表所示。

参数名称	参数类型	是否必填	默认值	说明
confiig	Object	否	无	Config 对象，参见 重要参数。

出参

参数名称	参数类型	说明
RoomInfo	Object	Promise

```
const { rtc } = requirePlugin('rtc')

// 发起人调用这个 API
rtc.start(config).then(roomInfo => {
  const { roomId, token } = roomInfo
  // 这里拿到创建的 roomId 和 token，用于后续其他人加入使用
})
```

Config 对象参见 重要参数 说明。

3.5.4. 在 uniapp 中使用音视频插件

本文用来指导用户如何在基于 Hbuilder 开发工具创建的 uniapp 项目下引入支付宝音视频插件。

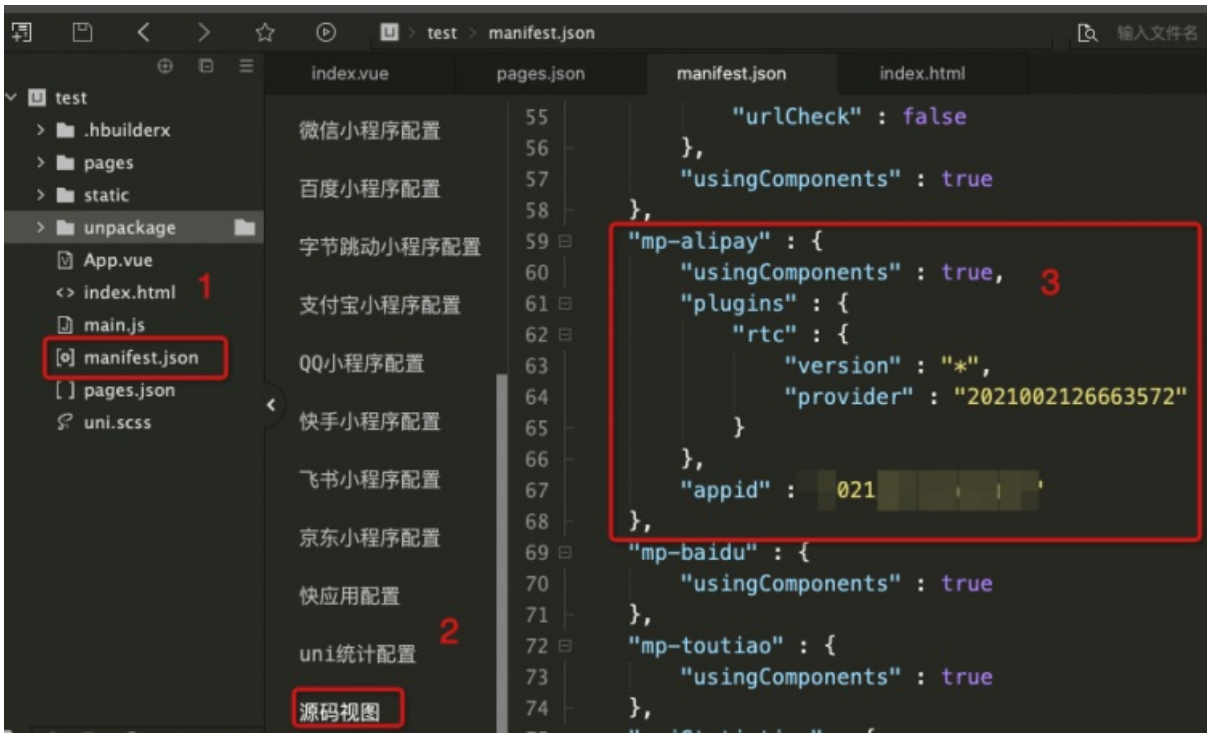
前提条件

在 uniapp 中使用音视频插件之前，请确保您已请先按 支付宝音视频插件 接入指引进行了相关设置，并开启对应的服务。

操作步骤

一、在 uniapp 中引入插件

在项目 manifest.json 中的各平台对应的字段内声明使用的插件，找到支付宝小程序的配置，具体配置参照所用插件的开发文档。



代码示例如下。

```
"mp-alipay" : {
      "usingComponents" : true,
      "plugins" : {
          "rtc" : {
              "version" : "*",
              "provider" : "2021002126663572"
          }
      },
      "appid" : "xxxxxxx"
  },
```

二、在页面中使用插件

在页面内使用插件内包含的组件需要在 `pages.json` 内对应页面的 `style` 节点下配置对应平台的 `usingComponents` ，示例如下：

```
{
    "path": "pages/index/index",
    "style": {
      "navigationBarTitleText": "音视频插件测试",
      "mp-alipay": {
        "usingComponents": {
          "rtcTag": "plugin://rtc/flex"
        }
      }
    }
  }
```

`value` 分为三段， `plugin` 为协议， `rtc` 为插件名称即引入插件时的名称， `rtcTag` 为插件暴露的组件名称。

页面中使用的代码示例如下。

```
<template>
 <view class="content">
  <!-- 使用音视频插件 -->
  <rtcTag
    config="{}"
    currentUserId="{}"
    onPlayerChange="handlePlayerChange"
    onUserChange="handleUserChange"
    onHangup="handleHangup"
    onError="handleError"
  />
 </view>
</template>
```

三、JS 中引入插件的方法及触发

```
// js引入
const { rtc } = requirePlugin('rtc');
export default {
  data() {
    return {
      title: '音视频插件测试'
    }
  },
  onLoad() {
    // this.startRtc();
  },
  methods: {
    startRtc(){
      // 触发方法
      rtc.start({}).then(roomInfo => {
        console.log('roomInfo:',roomInfo);
        // const { roomId, token } = roomInfo;
        // 这里拿到创建的 roomId 和 token，用于后续其他人加入使用
      })
    }
  }
}
```

Demo

点击 demo 即可下载 uniapp 项目的 demo。

3.6. 接入微信小程序

3.6.1. 快速开始

本文介绍的是将音视频通话服务接入微信小程序的操作步骤。WX-SDK 主要提供微信小程序端和其他移动端以及 Web 端进行音视频通话的能力。

集成步骤

> **重要**
>
> - 建议尽快将 SDK 升级至最新版本 1.0.9。
> - websocket 连接的域名地址需要在微信控制台增加白名单才能使用，另外微信小程序仅支持 wss 协议。
> - SDK 已经使用 socketTask 方式连接，如果您需要 websocket，请使用 socketTask 方式，参考我们提供的小程序 demo 使用方式。SocketTask 文档详情见 微信小程序开发文档 。
> - 微信开发工具无法调试音视频通话，必须使用真机调试。
> - live-pusher 可以支持不打开音频权限，仅推送视频。另外一定要开启摄像头权限，才能成功获取到音视频流。

1. 单击此处 下载 `mrtc-wx-sdk` 。
2. 将 SDK 的 JS 文件保存在本地项目下，包含 `rArtvcRoom.js` 和 `mtc.js` 文件。
3. 在项目 `room.js` 中引入 JS 文件： `import ArtvcRoom from './utils/ArtvcRoom.js';` 。

4. 实例化 SDK：`this.artvcChat = new ArtvcRoom(this);`，若其他页面需要使用实例，将实例存储在全局变量中，方便其他页面调用。
5. 设置连接 websocket 的 config，把 config 传入 Connect 接口并调用，参见 API 参考中的 Connect 接口。

   格式如下：

```
let config = {
  "uid":"wx_1635758034240",
  "biz_name":"demo",
  "appId":"default",
  "workspaceId":"default",
  "server_url":"wss://mrtc.mpaas.cn-hangzhou.aliyuncs.com/ws",
  "network_check_timeout":10
}
```

6. 注册由 SDK 提供的进入房间前的回调函数，具体参见 回调接口，具体实现参考小程序 Demo：包含 OnError、OnCreateRoom、OnJoinRoom、OnGetSign、OnLeaveRoom、OnConnect 和 OnGetFeedIds 这 7 个回调函数。

   代码示例：

```
app.globalData.artvcChat.OnCreateRoom = function(data) {
console.log(`这是在 room 页面 onCreateRoom 中收到的消息 data = ${JSON.stringify(data)}`);
        app.globalData.roomId = data.roomId;
        app.globalData.rtoken = data.rtoken;
        wx.navigateTo({
            url:'../pushAndlive/pushAndlive'
        })
}
```

7. 调用 CreatRoom 创建视频通话房间，或者 JoinRoom 加入已存在的房间。
8. 进入房间后，注册房间中实现操作的回调函数，接收加入房间后的发布、订阅和离开房间、对方加入房间和发送信息等回调信息。包含 OnPublish、OnSubscribe、OnGetSign、OnLeaveRoom、OnParticipantLeaveRoom、OnClientJoin、OnSendTxtMessageSucc 和 ArtvcChat.OnGetTxtMessage 这 8 个回调函数。
9. 调用 Publish 接口发布视频，将 OnPublish 回调返回的 rtmp 的推送地址 URL，赋值到 liver-pusher 组件上。
0. 如果是加入房间，会将已存在的房间用户通过 onGetFeedids 推送过来，再通过 subscribe 订阅其他用户的发布流。如果在中途房间有用户加入，会通过 OnClientJoin 推送新用户信息，通过用户信息订阅新用户发布流。

保留事件上报逻辑

SDK 需要通过 reportClientEvent 接口上报小程序事件，例如：开启摄像头，开启麦克风等。只需将 demo 中调用 reportClientEvent 的地方，复制到您自己的小程序代码中。在 onConnect 回调接口中添加如下代码：

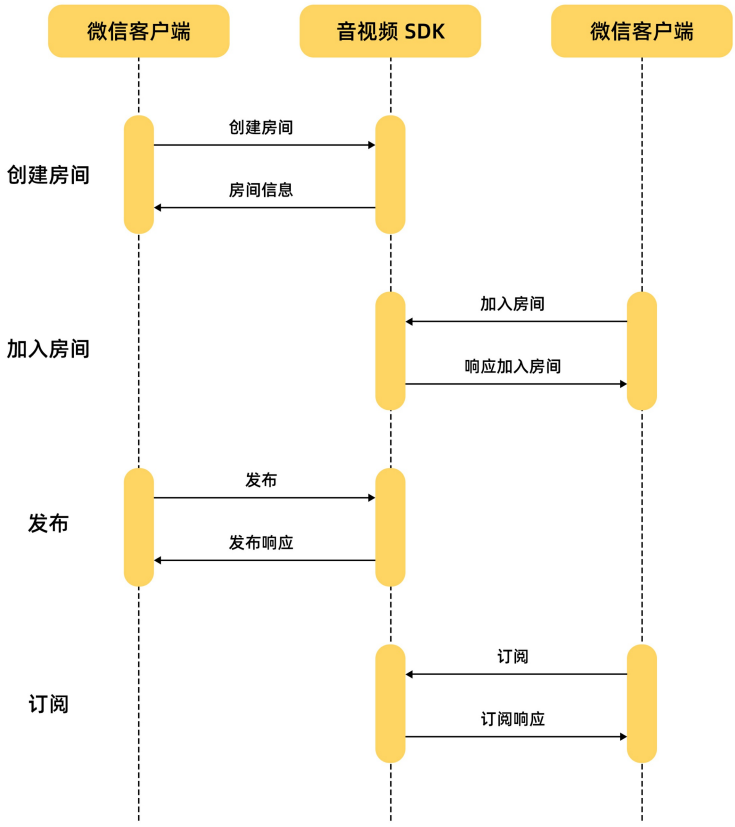
```
wx.onAppShow(app.globalData.artvcChat.reportClientEvent.bind(app.globalData.artvcChat,301
;
wx.onAppHide(app.globalData.artvcChat.reportClientEvent.bind(app.globalData.artvcChat,302
;
//开启耳机播放:
app.globalData.artvcChat.reportClientEvent(329);
//开启外放:
app.globalData.artvcChat.reportClientEvent(330);
//关闭摄像头:
app.globalData.artvcChat.reportClientEvent(321);
//关闭麦克风:
app.globalData.artvcChat.reportClientEvent(323);
//开启麦克风:
app.globalData.artvcChat.reportClientEvent(324);
//发布推流状态变更监听
  statechangePublish(e) {
    app.globalData.artvcChat.statechangePublish(e);
  }
```

接口协议

整个 API 封装在 ArtvcRoom 的类中，接口分为以下 2 类：

- 主调接口：供业务主动调用
- 回调接口：
  - 主调接口的反馈回调
  - 返回业务上的通知

使用流程



3.6.2. 版本更新说明

本文介绍的是微信小程序的音视频通话 SDK 的版本更新说明。

V 1.0.9（2021-12-17）

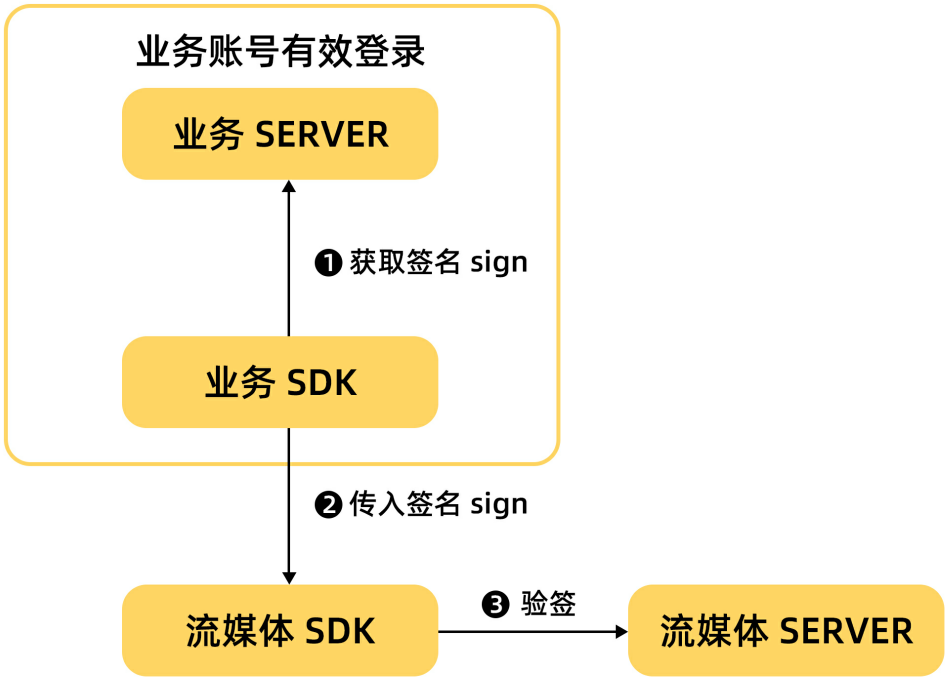
- 修复 websocket 独占导致上层业务无法使用 websocket 问题。
- websocket 开启压缩功能。
- 增加事件上报逻辑。

4.安全加签

为保证通话安全性，在使用音视频通话前，您需要对通话应用添加签名，用于客户端与流媒体服务端之间的安全性校验。

安全加签使用步骤

客户端每次通话前，向业务服务端发送请求获取签名，然后传入流媒体 SDK，并使用此签名与流媒体服务端完成安全性校验。完整的流程如下：



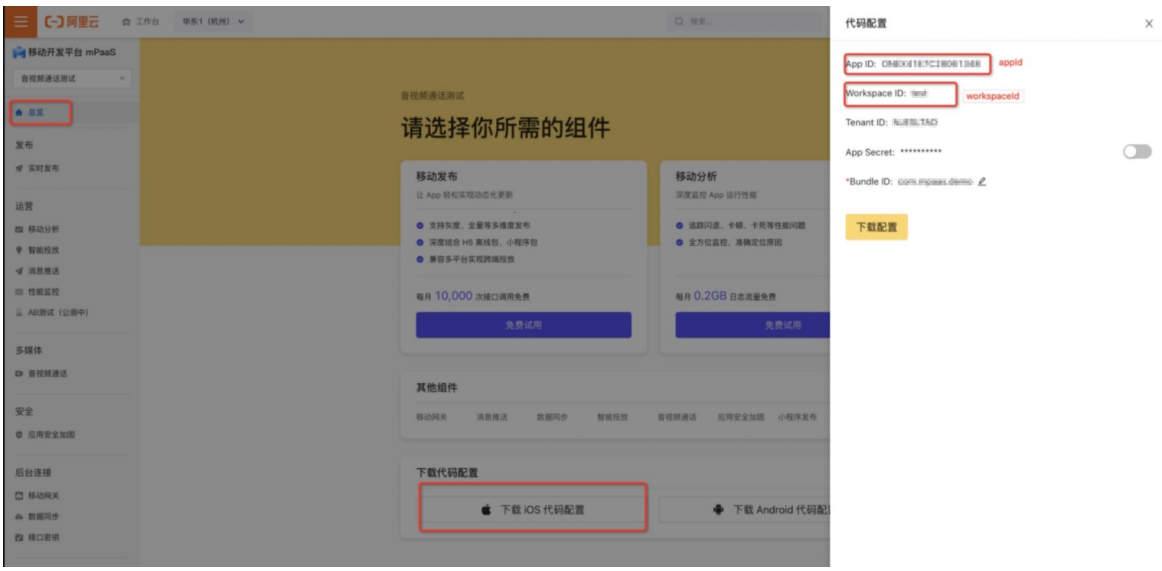
对应用进行安全加签的步骤如下：

1. 在 mPaaS 控制台上获取应用及音视频通话相关参数。加签代码中需要使用到 mPaaS 应用和音视频通话的相关参数，请在加签前获取 appId、workspaceId、bizName 及 key。
2. 在您的 mPaaS 账号已登录的情况下，在业务服务端上按照规则使用密钥生成签名。

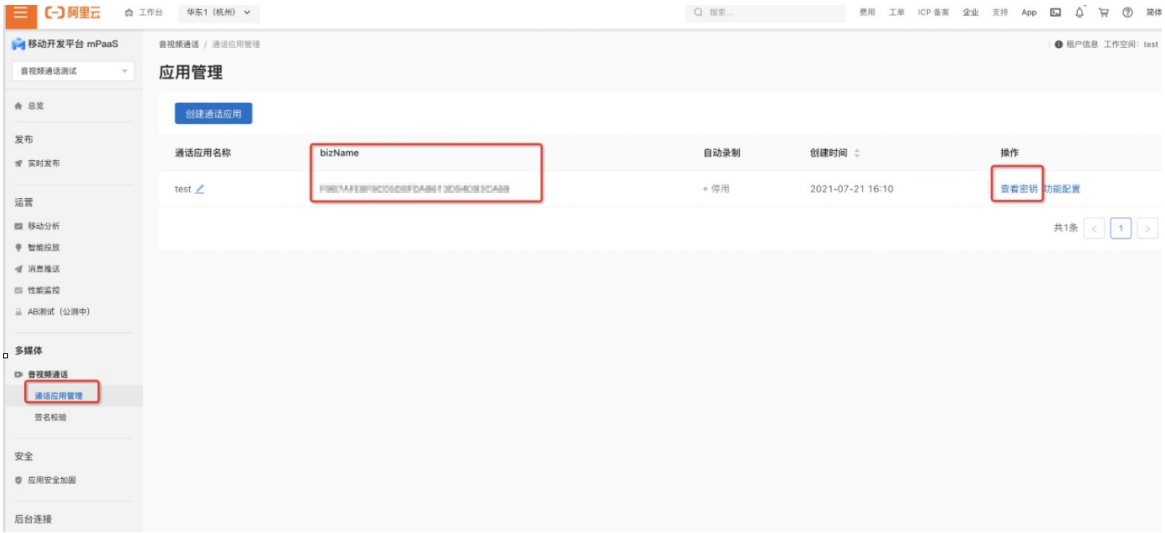
操作步骤

1. 获取 appId 及 workspaceId。
   i. 登录 mPaaS 控制台，进入目标应用。

ii. 单击 **总览**，打开对应客户端的 **代码配置** 面板，在右侧 **代码配置** 面板中获取 appId 及 workspaceId。



2. 获取应用的 bizName 及 key。bizName 为视频应用业务 ID，key 为应用的私钥。
   i. 登录 mPaaS 控制台，进入目标应用，单击 **音视频通话** > **通话应用管理**。
   ii. 在通话应用管理页面获取 bizName。
   iii. 单击 **查看密钥** 获取 key。



3. 在业务服务端生成签名。

i. 按照以下顺序将相关参数拼接成待加密的字符串。

```
String encryptStr = bizName + appId + workspaceId + uid + expireTime;
```

其中：

- `appId`、`workspaceId` 及 `bizName` 为您在 mPaaS 控制台获取的实际值。
- `uid` 和 `expireTime` 为用户自定义参数。`uid` 为业务用户 ID，由业务方传入。`expireTime` 为签名过期时间，单位为毫秒，由当前时间+有效期组成，示例如下。

> **说明**
>
> 目前 uid 只支持英文字母、数字、下划线的组合，且长度不超过 128 个字符。

```
long expire = 5 * 60 * 1000L;// 签名有效期（ms），比如 5 分钟
long expireTime = System.currentTimeMillis() + expire;// 签名生效截止时间：当前时间 +
有效期
```

ii. 使用 RSA 加密生成签名，其中 key 为您在 mPaaS 控制台获取的密钥。

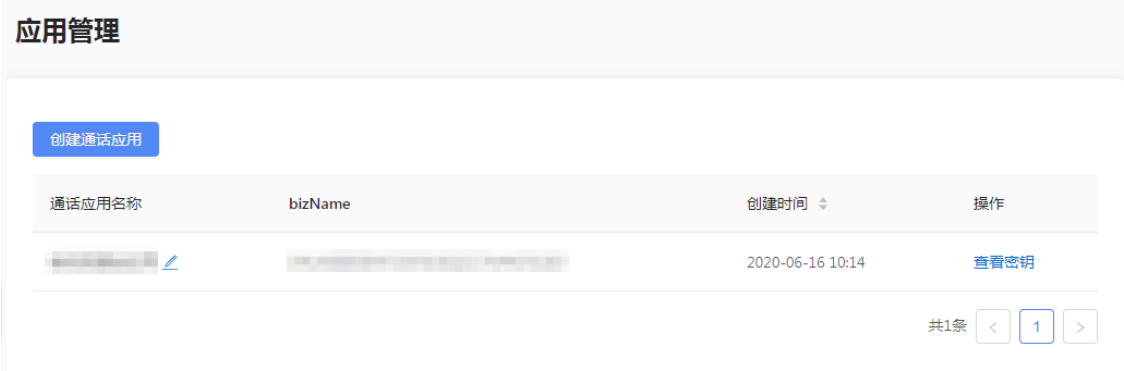
```
String sign = EncryptUtils.encryptByPrivate(encryptStr,
EncryptUtils.getPrivateKey(key));// 加密字符串

//RSA 私钥加密
 public static String encryptByPrivate(String content, PrivateKey privateKey) throw
s Exception {
     Cipher cipher = Cipher.getInstance("RSA");
     cipher.init(Cipher.ENCRYPT_MODE, privateKey);
     return
Base64.getEncoder().encodeToString(cipher.doFinal(content.getBytes("UTF-8")));
 }

  //将 Base64 编码后的 RSA 私钥字符串转成 PrivateKey 实例
 public static PrivateKey getPrivateKey(String privateKey) throws Exception {
     byte[] keyBytes = Base64.getDecoder().decode(privateKey);
     PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);
     KeyFactory keyFactory = KeyFactory.getInstance("RSA");
     return keyFactory.generatePrivate(keySpec);
 }
```

5.使用控制台

5.1. 通话应用管理

5.1.1. 创建通话应用

在接入音视频通话 SDK 后，您可前往控制台创建通话应用，并对通话应用进行管理。



操作步骤

1. 登录控制台，在左侧导航栏中选择 **通话应用管理**。
2. 点击页面右上方的 **创建通话应用**。
3. 在弹出的窗口中，输入通话应用名称。

> **说明**
> 创建通话应用时，后台会为通话应用自动生成 bizName 以及密钥。

4. 点击 **确定**，通话应用创建完成。新创建的通话应用就会出现在应用列表中。

5.1.2. 管理通话应用

在通话应用列表页面，您可以查看通话应用的详细信息，以及配置音视频通话等相关功能。

查看通话应用

在通话应用列表页面，您可以查看通话应用的名称、bizName、自动录制状态、创建时间以及密钥。

- **通话应用名称**：创建应用时输入的名称，可编辑。
- **bizName**：音视频通话场景码，应用创建后自动生成。使用同一个 bizName 的各个终端之间可以进行音视频通话。
- **自动录制状态**：自动录制是否开启。可点击 **操作** 列中的 **功能配置** 进行配置。
- **创建时间**：通话应用的创建时间。
- **密钥**：应用创建后自动生成，用于在服务端生成签名。点击 **查看文档** 可查看密钥使用说明。

功能配置

在通话应用列表页，点击应用 **操作** 列中的 **功能配置**，打开当前应用的功能配置页面。您可以在此页面配置云端自动录制、设置媒体流代理服务器地址。

配置时您需要点击 **编辑** 按钮进入编辑模式；配置完成后，点击 **保存** 按钮，保存您的配置信息。

配置云端自动录制及录制回调地址

云端自动录制指音视频通话服务端对通话内容自动进行录制的功能。您可以根据需要启用或关闭此功能，并设置录制回调地址。

- **云端自动录制：**开启该功能时，在音视频通话发起时服务端会自动对通话内容进行录制。通话结束时会结束录制，并生成录制文件。关闭时，服务端不进行自动录制。

> **说明**
>
> 云端录制的默认状态为 **停用**，需手动开启。

- **设置录制回调地址：**录制回调地址为使用 HTTP 或 HTTPS 协议的 URL。设置录制回调地址后，当开始录制、暂停录制、结束录制或录制文件出现异常时，服务端会将相关状态信息发送到该 URL。

  服务端使用 `POST(application/json)` 方式回调，回调请求的字段含义如下所示。

字段	字段类型	是否必传	说明
bizRequestId	String	是	请求 ID。
bizName	String	是	业务标识。
appId	String	是	mPaaS 应用的 ID。
workspaceId	String	是	工作空间 ID。
roomId	String	是	房间号 ID。
recordId	String	是	录制 ID。

eventCode	Int	是	0：录制某条流成功。 10：初始化成功。 11：录制结束。 50：录制警告。 99：录制结果。 500301：流断开警告。 500302：低帧率警告。 1000xx：录制启动阶段失败。 100001：连接 room 失败。 100002：加入房间失败。 100003：订阅流失败。 100004：订阅路数缺失。 100005：连接 mcu 失败。 100006：视频长宽比异常。 100007：实际自定义混流流数大于设置数目。 1001xx：录制过程中失败。 100101：录制文件创建失败。 100102：录制文件格式转换失败。 100103：文件保存失败。 100104：音视频分离失败。 100105：录制子进程崩溃。 100106：磁盘满或者写文件失败。 1002xx：录制结束阶段失败。 100201：上传文件失败。

recordResult	JSON	否	◦ 当 eventCode 的值为 99 时必传。json body 如下： status：int，录制状态，必传。 ▪ 2：代表录制成功，持久化文件成功。 ▪ 3：代表录制失败。 ◦ fileType：int，文件类型，当 status 的值为 2 时传输。 ▪ 1：local ▪ 2：OSS ▪ 3：AFTS ▪ 4：HDFS ◦ filePath：String，文件路径，当 status 的值为 2 时传输。 ◦ recordStartTime：long，录制第一帧的绝对时间（服务器时间，单位为 ms），当 status 的值为 2 时传输。 ◦ mediaType：int，录制文件的类型，当 status 的值为 2 时传输。 ▪ 0：音视频 ▪ 1：纯音频 ▪ 2：纯视频

业务服务器需返回的数据字段如下：

字段	类型	是否必传	说明
bizRequestId	String	是	回调请求中的 `bizRequestId` 。
code	int	是	状态码，成功为 `0` 。

数据示例如下。请求数据示例：

```
{
"bizRequestId":"123456789",
"bizName":"bizName",
"appId":"appId",
"workspaceId":"workspaceId",
"roomId":"xxxxxx",
"recordId":"record_xxx",
"eventCode":0,
"recordResult": {
    "status":2,
    "fileType":2,
    "filePath":"https://xxxxxx",
    "recordStartTime":1592817186122,
    "mediaType":0
}
}
```

返回数据示例：

```
{
"bizRequestId":"123456789",
"code":0
}
```

配置房间状态回调地址

房间状态回调地址为使用 HTTP 或 HTTPS 协议的 URL。设置房间状态回调地址后，当创建房间、加入房间、离开房间或销毁房间时，服务端会将相关状态信息发送到该 URL 地址。

服务端使用 `POST(application/json)` 方式回调，回调请求的字段含义如下所示。

字段	字段类型	是否必传	说明
bizRequestId	String	是	请求 ID。
uid	String	是	用户 ID。
bizName	String	是	业务标识。
appId	String	是	mPaaS 应用的 ID。
workspaceId	String	是	工作空间 ID。
roomId	String	是	房间号 ID。

eventCode	Int	是	1：创建房间。 2：加入房间。 3：离开房间。 4：销毁房间。

业务服务器需返回的数据字段如下：

字段	类型	是否必传	说明
bizRequestId	String	是	回调请求中的 `bizRequestId` 。
code	int	是	状态码，成功为 `0` 。

数据示例如下。请求数据示例：

```
{
"eventCode":3,
"uid":"uid",
"bizName":"bizName",
"appId":"appId",
"bizRequestId":918479352902861,
"time":1669184793529,
"roomId":"roomId",
"workspaceId":"workspaceId"
}
```

配置代理服务器地址

若您的网络中存在安全隔离区，无法直接访问阿里云服务器，可通过设置代理服务器对媒体流数据进行转发。设置媒体流代理服务器后，音视频流数据将通过该代理服务器进行转发。

> **重要**
>
> 代理服务器地址错误会导致音视频通话异常，若无必要请勿填写。

配置时需分别填入服务器地址和地址对应的出网 IP，请确保输入的地址准确。

- 媒体流代理服务器地址格式为： `example.aliyundoc.com` 。
- 若存在多个出网 IP，使用半角分号（;）隔开。

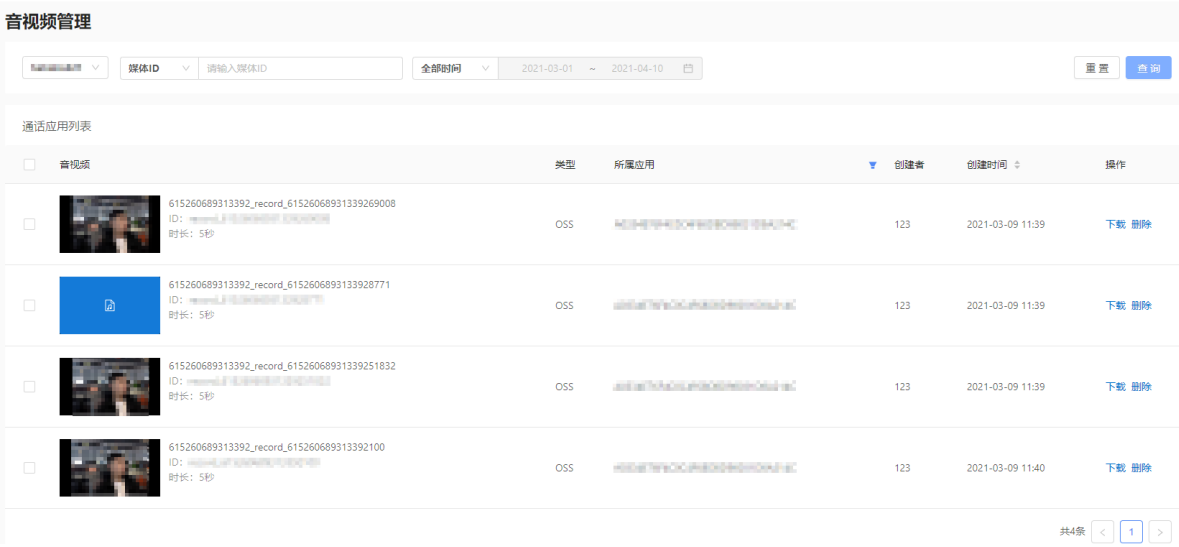
5.2. 录制文件管理

若您在通话过程中使用录制功能，可在 mPaaS 控制台对录制文件进行管理，您可以播放、下载或删除录制文件。

操作步骤

1. 登录 mPaaS 控制台，在左侧导航栏中选择 **音视频通话** > **录制文件管理** 。
2. 选择通话应用，设置查询时间，点击 **查询** 。也可以输入媒体 ID 进行查询。

> **说明**
>
> 查询时间段默认为 **全部时间**，即查询该通话应用下所有录制文件，您也可以在下拉框中选择 **自定义时间**，设置自定义时间段进行查询。



3. 在通话应用列表页面，根据需要执行播放、下载或删除操作。
   - 播放录制文件点击音视频文件的缩略图，并在弹出的播放窗口中点击开始按钮。
   - 下载录制文件
     - 下载单个文件：在列表中找到目标文件，点击 **操作** 列的 **下载**。
     - 批量下载文件：在列表中选择需要下载的文件，点击页面右下角的 **批量下载**。
   - 删除录制文件
     - 删除单个文件：在列表中找到目标文件，点击 **操作** 列的 **删除** 。
     - 批量删除文件：在列表中选择需要下载的文件，点击页面右下角的 **批量删除**。

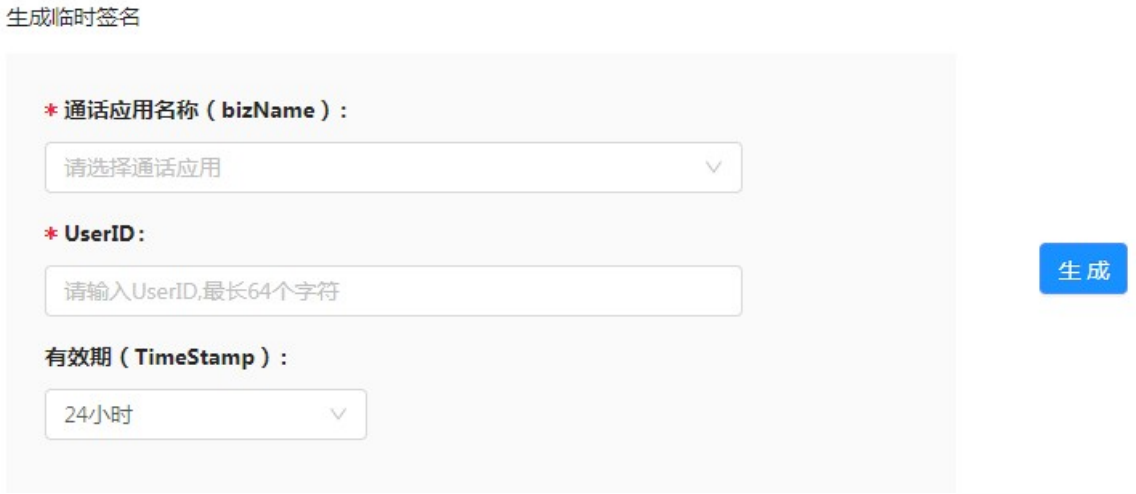
> **重要**
>
> 录制文件删除后不可恢复，请谨慎操作。

5.3. 签名校验

在左侧导航栏中选择 **签名校验**，可进入生成临时签名、验证签名的页面。

生成临时签名

在上方的 **生成临时签名** 区域，可对指定的 UserID 生成针对指定通话应用的临时签名。

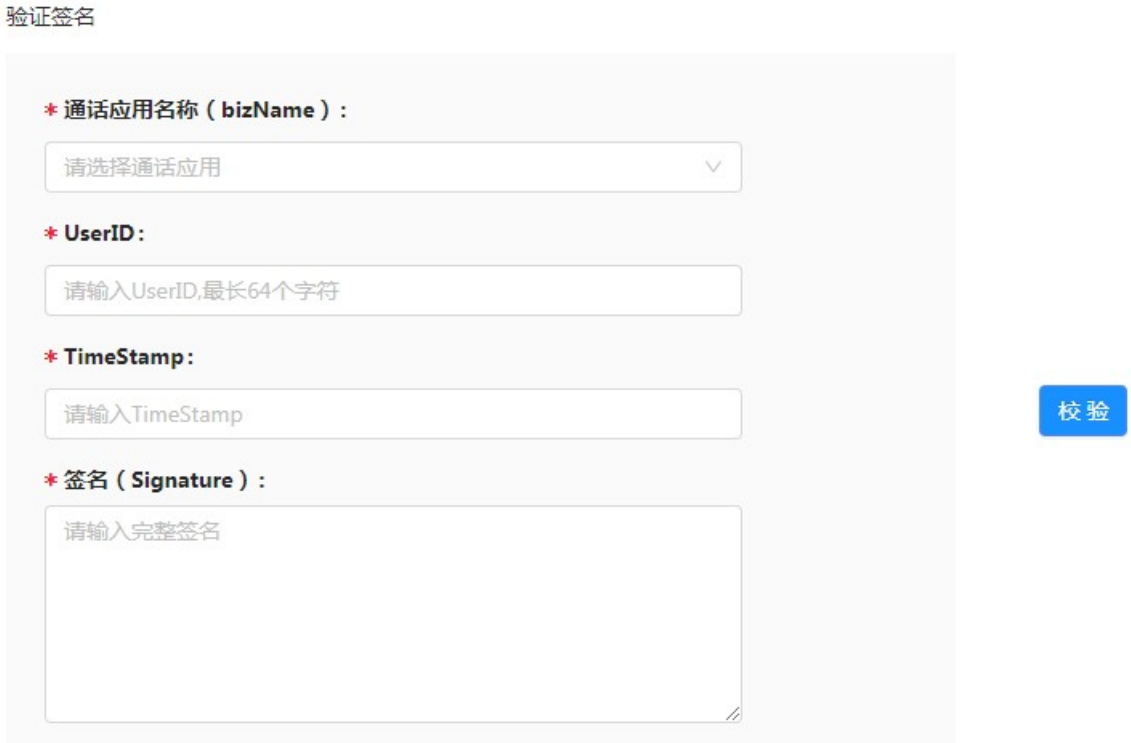


1. 选择 **通话应用名称（bizName）**，即已创建的通话应用。
2. 输入 **UserID**。
3. 选择 **TimeStamp**，即签名有效期。可选择 **24 小时**、**3 天** 或 **7 天**。
4. 单击 **生成**，即可在右侧区域展示生成的临时签名信息。

> **说明**
>
> 如果生成临时签名区域中已有信息，您可单击该区域右上方的 ⊗ 来清空已有信息。

验证签名

在下方的 **验证签名** 区域，可验证上方生成的临时签名，也可以验证您在服务器中生成的签名。



1. 选择 **通话应用名称（bizName）**，即已创建的通话应用。

2. 输入 **UserID**。
3. 输入 **TimeStamp**，即签名有效期。
4. 输入 **签名（Signature）**，可输入在上方生成临时签名区域生成的签名，也可输入您在服务器中生成的签名。
5. 单击 **校验**，即可在右侧区域展示签名的校验结果。

> **说明**
>
> 如果校验临时签名区域中已有信息，您可单击该区域右上方的 ⊗ 来清空已有信息。

5.4. 用量统计

用量统计功能用于在指定通话应用下，根据使用时间段筛选出统计到的音视频通话数据进行展示。主要涉及指定通话应用下的通话时长、在线时长、房间数、用户数四个数据维度。具体包括相关数据维度下的数据指标概览、使用趋势、使用详情三个方面的数据。

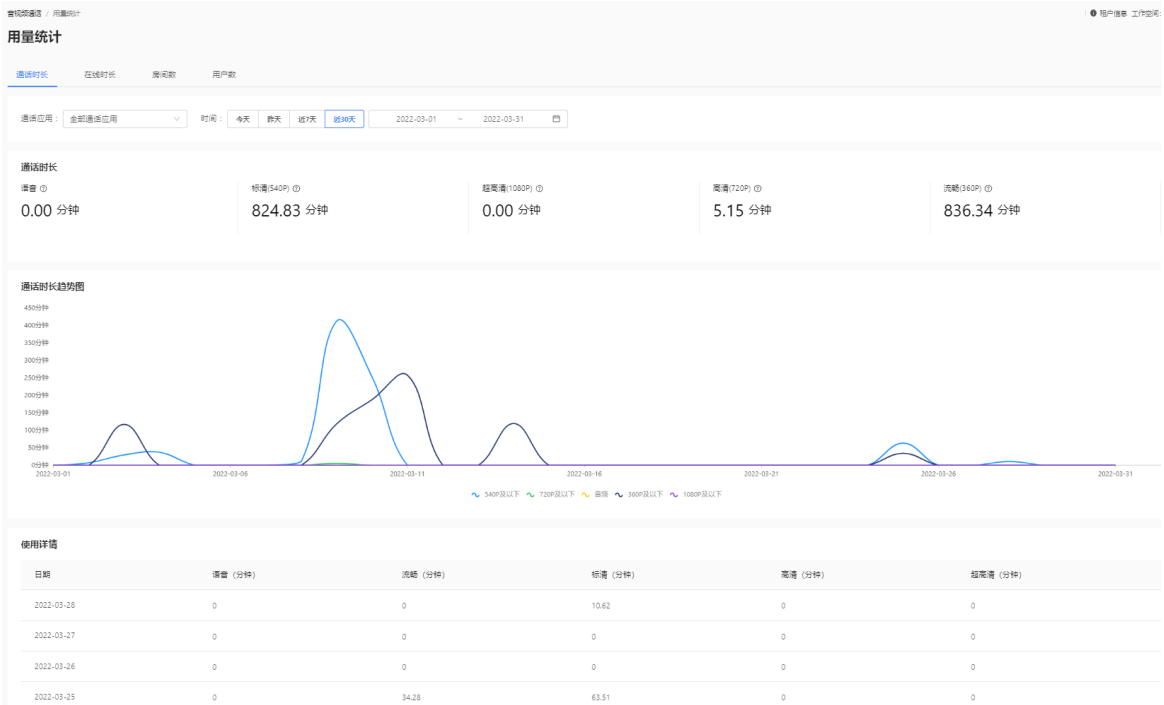
前提条件

您已经在 mPaaS 控制台上创建了项目和应用，详细操作请参见 创建通话应用 。

使用步骤

1. 登录 mPaaS 控制台，在左侧导航栏中选择 **多媒体服务** > **音视频通话** > **用量统计**，默认进入 **通话时长** 页面。
2. 选择 **通话时长**、**在线时长**、**房间数** 或 **用户数** 标签，然后进入相应标签页按通话应用和时间来查询相关数据。
   - 通话应用：单击 **通话应用** 下拉框，可选择查看全部通话应用或指定通话应用的使用数据。
   - 时间范围：可选择查看今天、昨天、近 7 天、近 30 天的通话数据。同时也支持自定义时间段的数据查询。

     > **说明**
     >
     > 自定义时间段支持查询过去一年内，时间跨度最长为 30 天的数据。目前仅支持查询 2022 年 1 月 1 日 之后的数据。
3. 每个标签页可展示全部通话应用或指定通话应用的以下三方面数据。
   - 指标概览：展示相应数据维度下的指标数据。例如在通话时长维度下，展示使用语音和不同清晰度进行视频通话的时长。
   - 趋势图：可以观察到相应数据维度下的各个指标的变化趋势。
   - 使用详情：可以观察到不同数据维度下，每天使用音视频通话的详细数据量。



数据指标

下表分别介绍通话时长、在线时长、房间数、用户数四个维度下的相关数据指标。

数据维度	数据指标	说明
通话时长	语音通话时长	统计时间段内使用纯语音通话的时长。 **说明** 在线用户创建或加入通话房间，与他人通过音视频进行交流时，即为通话状态。
	流畅（360P）通话时长	统计时间段内使用 360P 及以下分辨率进行视频通话的时长。
	标清（540P）通话时长	统计时间段内使用 540P 及以下分辨率进行视频通话的时长。
	高清（720P）通话时长	统计时间段内使用 720P 及以下分辨率进行视频通话的时长。
	超高清（1080P）通话时长	统计时间段内使用 1080P 及以下分辨率进行视频通话的时长。

数据维度	数据指标	说明
在线时长	累计在线时长	统计时间段内所有用户与音视频建立连接的总时长。 说明 用户使用设备与音视频系统建立连接时为在线状态，断开连接时为下线状态。
	发布流时长	统计时间段内所有用户发布的音频流、视频流的总时长。 说明 视频通话中包含音频与视频时，仅按视频统计。
	订阅流时长	统计时间段内所有用户订阅的音频流、视频流的总时长。 说明 视频通话中包含音频与视频时，仅按视频统计。
房间数	累计房间数	统计时间段内所有创建的房间数。
	房间数峰值	统计时间段内同时在线的最高房间数。
用户数	累计在线人数	统计时间段内所有与音视频系统建立连接的人数，以用户 ID 去重。
	累计通话人数	统计时间段内所有参与通话（加入房间）的人数，以用户 ID 去重。
	在线人数峰值	统计时间段内同时在线的最高人数。
	通话人数峰值	统计时间段内同时进行音视频通话的最高人数。

对于每个用户，mPaaS 音视频通话的计费时长均根据当前用户订阅音视频流的时长进行计算。

音视频通话计费采用 后付费模式（以分钟为单位），如您需要了解详细计费方式，请参考 音视频通话计费。

6.API 参考

6.1. Android API

6.1.1. 权限及隐私说明

本文介绍使用音视频通话在 Android 中所需申请的权限与隐私保护。

权限申请

进行音视频通话需要以下权限：

- 网络相关权限。
- 摄像头权限。
- 麦克风权限。
- 声音设置基础权限，包括听筒外放设置、蓝牙耳机管理权限。
- 监听系统的来电和去电广播权限 READ_PHONE_STATE，详见下方 隐私保护。

> **重要**
>
> SDK 需要上层告知权限的申请结果。请参考 onRequestPermissionsResult 接口。

隐私保护

- 为了保护用户的隐私，需要在用户拨打电话或者接听电话时停止麦克风和摄像头数据的发送，此时需要用 READ_PHONE_STATE 权限去监听系统的来电和去电广播。如果用户没有授予该权限也不影响视频通话功能的使用，但是，在未授予此权限的情况下拨打电话或者接听电话时，不保证会终止摄像头和麦克风数据的发送，不利于隐私保护。

  > **重要**
  >
  > 若接通了电话，SDK 会断开音频流和视频流，电话挂断后不会自动恢复。

- 当支付宝应用切至后台运行时，仅关闭摄像头来停止摄像头数据的发送，但此时麦克风数据未停止发送，双方仍可以通过麦克风进行交流。

  > **重要**
  >
  > 非支付宝场景，前后台切换事件需要上层告知 SDK，请参考 onUserLeaveOrReturn 接口。

6.1.2. 重要参数

本文介绍音视频通话 API 在接入 Android 时涉及的重要参数。

参数	说明
CreateRoomParams	创建房间。

参数	说明
JoinRoomParams	加入房间。
PublishConfig	发布媒体流。
UnpublishConfig	取消订阅。
SubscribeConfig	订阅媒体流。
UnsubscribeConfig	取消订阅媒体流。
LocalVideoInfo	本地视频信息，可用于暂停继续本地视频流的推送。
RemoteVideoInfo	远程视频信息，可以用于暂停继续远端视频流的渲染。
LocalAudioInfo	本地音频流信息，可以用于暂停继续本地音频流的推送。
RemoteAudioInfo	远端音频流信息，可以用于暂停继续远程音频的播放。
ParticipantInfo	视频通话参与者的用户信息，可用于订阅，发布等各种用户操作。
RoomInfo	创建房间返回的房间信息。
PublishEventCode	发布相关操作返回的事件码信息。
FeedInfo	可被订阅流的信息。
SubscribeEventCode	订阅相关操作返回的事件码信息。
Logger	业务可提供自定义 log，将 SDK 的日志输出到业务的 log 处。
ErrorCode	错误码。

CreateRoomParams

创建房间。作为创建房间的参数，只有该参数填写正确，才能顺利创建音视频房间。

属性	类型	说明	推荐值	是否必填
uid	String	用户 ID。 说明 目前 uid 只支持英文字母、数字、下划线的组合，且长度不超过 128 个字符。	无	是
bizName	String	业务名称，需要向服务提供方进行申请。	无	是
signature	String	业务签名。	无	是
extraInfo	String	额外信息，与业务相关。	null	否

JoinRoomParams

加入房间。作为进房参数，只有该参数填写正确，才能顺利进入 roomId 所指定的音视频房间。

属性	类型	说明	推荐值	是否必填
uid	String	用户 ID。	无	是
bizName	String	业务名称，需要向服务提供方进行申请。	无	是
roomId	String	房间 ID。	无	是
rtoken	String	进入房间所需要的 Token。	无	是
signature	String	业务签名。	无	是
envType	int	服务提供类型，当前支持 alipay 和 aliyun，alipay=0（默认值）、aliyun=1。	0	否

PublishConfig

发布媒体流。配置需要发布媒体流的参数，包括音频和视频。

属性	类型	说明	推荐值	是否必填
videoSource	PublishVideoSource	待发布媒体流的视频源 VIDEO_SOURCE_CAMERA（默认值）。	无	否
videoProfile	VideoProfile	视频属性配置PROFILE_360_640P_15（默认值）。	PROFILE_540_960P_15	否
audioSource	PublishAudioSource	待发布媒体流的音频源 AUDIO_SOURCE_MIC（默认值）。	无	否
videoCustomWidth	int	用户自定义视频输出 width 仅当 videoProfile = PROFILE_CUSTOM 时生效。	无	否
VideoCustomHeight	int	用户自定义视频输出 height 仅当 videoProfile = PROFILE_CUSTOM 时生效。	无	否
videoCustomFps	int	用户自定义视频输出 fps 仅当 videoProfile = PROFILE_CUSTOM 时生效。	无	否
videoCustomMaxBitrate	int	用户自定义视频输出的最大 bitrate，即自定义视频编码时的最大码率，当输入值大于此值时，MRTC 将根据本地推送模式（流畅度优先模式、分辨率优先模式、平衡模式）进行丢帧或降低清晰度。 仅当 videoProfile = PROFILE_CUSTOM 时生效。	无	否
videoCustomMinBitrate	int	预留参数，暂未使用。 用户自定义视频输出最小比特率，仅当 videoProfile = PROFILE_CUSTOM 时生效。	无	否

无论是横屏还是竖屏（分别对应 640 × 360 和 360 × 640 的分辨率），videoProfile 都会选择 PROFILE_360_640P_15 或者 PROFILE_360_640P_30。

```
enum PublishVideoSource {
VIDEO_SOURCE_CAMERA, //视频源是摄像头
VIDEO_SOURCE_SCREEN, //视频源是屏幕共享
VIDEO_SOURCE_CUSTOM, //视频源是自定义数据
VIDEO_SOURCE_NULL    //禁止输出视频
}
enum VideoProfile {
PROFILE_360_640P_15, //分辨率为 360 * 640, 帧率为 15
PROFILE_360_640P_30, //分辨率为 360 * 640, 帧率为 30
PROFILE_540_960P_15, //分辨率为 540 * 960, 帧率为 15
PROFILE_540_960P_30, //分辨率为 540 * 960, 帧率为 30
PROFILE_720_1280P_15, //分辨率为 720 * 1280, 帧率为 15
PROFILE_720_1280P_30, //分辨率为 720 * 1280, 帧率为 30
PROFILE_1080_1920P_15, //分辨率为 1080 * 1920, 帧率为 30
PROFILE_1080_1920P_30, //分辨率为 1080 * 1920, 帧率为 30
PROFILED_EFAULT, //默认, 分辨率为 360 * 640, 帧率为 15
PROFILE_CUSTOM //自定义模式, 需要设置自定义视频属性
}
enum PublishAudioSource {
AUDIO_SOURCE_MIC, //音频源是麦克风
AUDIO_SOURCE_NULL //禁止输出音频
}
```

UnpublishConfig

取消订阅。配置需要取消发布的媒体流信息。

属性	类型	说明	推荐值	是否必填
videoSource	PublishVideoSource	待取消发布的媒体流视频源 VIDEO_SOURCE_CAMERA（默认值）。	无	否
audioSource	PublishAudioSource	待取消发布媒体流的音频源 AUDIO_SOURCE_MIC（默认值）。	无	否

SubscribeConfig

订阅媒体流。配置需要订阅的媒体流参数。

属性	类型	说明	推荐值	是否必填
info	FeedInfo	需要订阅的流信息，包含 uid、feedId 和 tag 信息。	无	是

UnsubscribeConfig

取消订阅媒体流。配置需要取消订阅的媒体流参数。

属性	类型	说明	推荐值	是否必填
info	FeedInfo	需要订阅的流信息，包含 uid、feedId 和 tag 信息。	无	是

LocalVideoInfo

本地视频信息，可用于暂停继续本地视频流的推送。包含了视频源信息。

属性	类型	说明	推荐值	是否必填
videoSource	PublishVideoSource	本地媒体流视频源信息 VIDEO_SOURCE_CAMERA（默认值）。	无	否

RemoteVideoInfo

远程视频信息，可以用于暂停继续远端视频流的渲染。包含了远端视频流信息。

属性	类型	说明	推荐值	是否必填
feedId	String	远端媒体流 ID。	无	是

LocalAudioInfo

本地音频流信息，可以用于暂停继续本地音频流的推送。包含了本地音频流源信息。

属性	类型	说明	推荐值	是否必填
audioSource	PublishAudioSource	本地音频源信息 AUDIO_SOURCE_MIC（默认值）。	无	否

RemoteAudioInfo

远端音频流信息，可以用于暂停继续远程音频的播放。包含了远程音频信息。

属性	类型	说明	推荐值	是否必填
feedId	String	远端媒体流 ID。	无	是

FeedInfo

可被订阅流的信息。一条可被订阅流的信息，包含了流所属用户的 uid、流的 feedId 和流的额外属性 tag。

属性	类型	说明	推荐值	是否必填
uid	String	可被订阅流的所属用户 ID。	无	是
feedId	String	可被订阅流的 ID。	无	是
tag	String	可被订阅流的额外属性，目前是 PublishVideoSource 的字符串表现形式。	无	否

ParticipantInfo

视频通话参与者的用户信息，可用于订阅、发布等各种用户操作。用户信息，包含用户 ID、用户发布的流信息和用户订阅的流信息。

属性	类型	说明
uid	String	用户 uid。
feedList	List	用户推送的流信息，一个用户可以推送多路流供其他用户订阅。
subscribeIdList	List	用户订阅的流信息，一个用户可以订阅多路流。

RoomInfo

创建房间返回的房间信息。通过该房间信息，可以加入对应的房间，进行音视频通话。

属性	类型	说明
roomId	String	房间 ID。
rtoken	String	房间对应的 Token。

PublishEventCode

发布相关操作返回的事件码信息。在发布、取消发布后，返回的对应事件码信息。

属性	说明
PUBLISH_START	发布操作有效，服务器接受该发布请求，本地准备资源进行发布。

属性	说明
PUBLISH_SUCCESS	发布操作成功，媒体流成功推送到服务器上。
PUBLISH_FAIL	发布操作失败，可能是媒体流获取失败，也可能是媒体流推送失败。
PUBLISH_DISCONNECT	发布断开，媒体流推送遇到错误，与服务器断开连接。
UNPUBLISH_SUCCESS	取消发布成功。
UNPUBLISH_FAIL	取消发布失败，参数有误或者服务器出错。

SubscribeEventCode

订阅相关操作返回的事件码信息。在订阅、取消订阅后，返回的对应事件码信息。

属性	说明
SUBSCRIBE_START	订阅操作有效，服务器接受该订阅请求，本地准备拉流操作。
SUBSCRIBE_SUCCESS	订阅操作成功，成功从服务器拉取媒体流到本地。
SUBSCRIBE_FAIL	订阅操作失败，可能是参数错误，也可能是服务器错误。
SUBSCRIBE_DISCONNECT	订阅断开，媒体流推送遇到错误，与服务器断开连接。
UNSUBSCRIBE_SUCCESS	取消订阅成功。
UNSUBSCRIBE_FAIL	取消订阅失败，参数有误或者服务器出错。

Logger

业务可提供自定义 log，将 SDK 的日志输出到业务的 log 处，只需实现该 Logger 接口，然后通过 setLogger 设置到 SDK 即可。

```
interface Logger {
int v(String tag, String msg);
int d(String tag, String msg);
int i(String tag, String msg);
int w(String tag, String msg);
int e(String tag, String msg);
int e(String tag, String msg, Throwable tr);
}
```

ErrorCode

各个错误码的具体说明见下表。

错误	错误码	说明
ERROR_NULL	0	无错误。
ERROR_CREATE_ROOM	-115	创建房间失败，可能参数为 null，或者参数在服务器端验证不通过，需要查阅参数正确配置后，再次创建房间。也可能是服务器内部处理出错（概率很低），如果遇到服务器内部错误，请通知服务提供商。
ERROR_JOIN_ROOM	-116	加入房间失败，可能参数为 null，或者参数在服务器端验证不通过，需要查阅参数正确配置后，再次加入房间。也可能是服务器内部处理出错（概率很低），如果遇到服务器内部错误，请通知服务提供商。
ERROR_LEAVE_ROOM	-1002	退出房间失败，可能当前不在房间内，或者服务器验证不通过。无论退出房间成功与否，都会释放所有的发布和订阅。
ERROR_INVITE_TO_JOIN_ROOM	-1003	邀请对方加入房间失败，可能参数为 null，或者参数在服务端验证不通过。
ERROR_NETWORK	-1004	网络错误，无法正常连接到服务器。此时，无法正常执行任何发布订阅等操作。
ERROR_CAMERA_PERMISSION	-104	打开相机预览失败，相机没有正确授权。
ERROR_MIC_PERMISSION	-105	打开相机预览失败，麦克风没有正确授权。
ERROR_OPEN_CAMERA	-1007	打开相机预览失败，无法正常打开摄像头。
ERROR_OPEN_MIC	-1008	打开相机预览失败，无法正常打开麦克风。

6.1.3. 公共接口

本文介绍音视频通话 API 在接入 Android 时涉及的公共接口。

接口类型	API	说明
基础接口	getInstance	创建 AlipayRtcEngine 实例（同一时间只会存在一个实例）。
	setRtcListenerAndHandler	设置回调事件的监听和对应的 Handler。
	setImListener	设置 IM 文本信息交互的监听。
	setInviteListener	设置邀请模式下邀请信息的监听。
	setCustomPublishListener	设置自定义推流模式下的本地预览监听。
	destroy	销毁引擎。
	onUserLeaveOrReturn	非支付宝 App 用于通知压后台回前台事件，用于压后台关闭摄像头、悬浮框权限监听、悬浮框弹出和隐藏。 • onResume 回前台为1。 • onPause 压后台为 0。
房间相关接口	createRoom	创建房间。
	joinRoom	加入房间。
	leaveRoom	离开房间。
	setAutoPublishSubscribe	设置是否自动发布，是否自动订阅。
发布相关接口	isAutoPublish	查询当前是否为自动发布模式。
	configAutoPublish	设置是否允许发布视频流。
	getCurrentAutoPublishConfig	获取当前自动发布的配置。
	publish	手动发布设置的音频流和视频流。

接口类型	API	说明
	unpublish	取消发布流。
	updateVideoProfile	切换发布分辨率。
	pushCustomVideoData	自定义推流，推送自定义图像数据。
订阅相关接口	isAutoSubscribe	查询当前是否为自动订阅模式。
	subscribe	手动订阅视频和音频流。
	unsubscribe	取消订阅流。
视频相关接口	muteLocalVideo	设置是否暂停发布本地视频流。
	muteAllLocalVideo	设置是否暂停发布本地所有的视频流。
	switchCamera	切换前后摄像头。
	muteRemoteVideo	设置是否暂停播放远端视频流。
	muteAllRemoteVideo	设置是否暂停播放所有的远端视频流。
音频相关接口	muteLocalAudio	设置是否暂停发布本地音频。
	muteAllLocalAudio	设置是否暂停本地所有音频流。
	muteRemoteAudio	设置是否暂停播放远端音频流。
	muteAllRemoteAudio	设置是否暂停播放所有远端音频流。
	enableSpeaker	设置是否切换扬声器输出。
预览接口	startCameraPreview	开始本地相机预览。
	stopCameraPreview	停止本地相机预览。

接口类型	API	说明
查询远端用户接口	getRemoteUsers	获取远端用户信息。
其他接口	setLogger	设置自定义日志输出。
	invite	邀请对方加入。
	replyInvite	回复对方邀请。
	sendMessage	发送文本消息。
	snapshot	针对某路视频流进行截屏。
	onRequestPermissionsResult	如果需要处理权限申请，则业务需要将该权限申请结果通过该接口通知到 SDK 层进行对应的处理。
	onActivityResult	屏幕共享权限申请回调。

getInstance

- **声明：** `public static AlipayRtcEngine getInstance( Context context )`
- **说明：** 创建 AlipayRtcEngine 实例（同一时间只会存在一个实例）。

> **说明**
> 由于涉及到权限申请，此处的 context 必须是 activity 的 context。

- **参数：**

参数	类型	说明
context	Context	上下文内容

- **返回值：** 无。

setRtcListenerAndHandler

- **声明：** `public void setRtcEngineEventListener( AlipayRtcEngineEventListener listener, Handler handler )`
- **说明：** 设置回调事件的监听。
- **参数：**

参数	类型	说明
listener	AlipayRtcEngineEventListener	接收回调事件的监听器。
handler	Handler	回调事件的 handler，必须设置，不建议使用主线程的 handler，建议新建一个线程处理回调。 如果没有设置，则无法回调任何结果。

- **返回值：**无。

setImListener

- **声明：** `public void setImListener( AlipayRtcEngineIMListener imListener )`
- **说明：**设置文本信息交互的监听。
- **参数：**

参数	类型	说明
listener	AlipayRtcEngineIMListener	文本交互监听器。

- **返回值：**无。

setInviteListener

- **声明：** `public void setInviteListener( AlipayRtcEngineInviteListener inviteListener )`
- **说明：**设置邀请模式下的邀请信息监听。
- **参数：**

参数	类型	说明
listener	AlipayRtcEngineInviteListener	邀请模式下的交互监听器。

- **返回值：**无。

setCustomPublishListener

- **声明：** `public void setCustomPublishListener( AlipayRtcEngineCustomPublishListener listener )`
- **说明：**设置自定义推流模式下的本地预览监听，当前包括屏幕共享和自定义推流。如果当前没有使用到自定义推流模式，可以不设置该参数。
- **参数：**

参数	类型	说明
listener	AlipayRtcEngineCustomPublishListener	自定义推流的本地 view 交互 listener。

- **返回值：**无。

destroy

- **声明：** `public void destroy()`
- **说明：**销毁引擎，destroy 后，会退出房间，同时销毁本地引擎，再次使用功能，需要重新 `getInstance()` 初始化引擎。
- **参数：**无。
- **返回值：**无。

onUserLeaveOrReturn

- **声明：** `public void onUserLeaveOrReturn( int type )`
- **说明：**业务调用此接口告知 App 的前后台状态发生了改变。
- **参数：**

参数	类型	说明
type	int	前后台状态标志： ◦ 1：表示 App 从后台切到前台，传入值为 1。 ◦ 0：表示 App 从前台切到后台，传入值为 0。

- **返回值：**无。

createRoom

- **声明：** `public void createRoom( CreateRoomParams params )`
- **说明：**创建房间，房间创建后可以通知其他人，邀请他们加入该房间。创建房间的结果由回调中的 `void onRoomInfo( RoomInfo info )` 返回，详情请参见 回调函数。
- **参数：**

参数	类型	说明
params	CreateRoomParams	创建房间所需要的信息，详情请参见 重要参数。

- **返回值：**无。

joinRoom

- **声明：** `public void joinRoom( JoinRoomParams params )`
- **说明：**加入房间，加入他人创建好的房间，若该房间失效，则加入失败。加入房间的结果由回调中的 `void onEnterRoom( int result )` 返回，详情请参见 回调函数。

- **参数：**

参数	类型	说明
params	JoinRoomParams	加入房间所需要的信息，详情请参见 重要参数。

- **返回值：**无。

leaveRoom

- **声明：** `public void leaveRoom()`
- **说明：**离开房间，如果是加入者（joinRoom 的方式加入），在离开房间后，可以再次加入，直到所有的与会者退出，房间没人后该房间被解散，无法再次进入。离开房间的结果由回调中的 `void onLeaveRoom( int result )` 返回，详情请参见 回调函数。
- **参数：**无。
- **返回值：**无。

setAutoPublishSubscribe

- **声明：** `public void setAutoPublishSubscribe( boolean autoPub, boolean autoSub )`
- **说明：**设置是否自动发布，是否自动订阅。默认是自动发布和订阅，必须在 createRoom 和 joinRoom 之前设置。
- **参数：**

参数	类型	说明
autoPub	boolean	是否自动发布： ◦ true：表示自动发布。 ◦ false：表示手动发布。
autoSub	boolean	是否自动订阅： ◦ true：表示自动订阅。 ◦ false：表示手动订阅。

- **返回值：**无。

isAutoPublish

- **声明：** `public boolean isAutoPublish()`
- **说明：**查询当前是否处于自动发布模式。
- **参数：**无。
- **返回值：**如果处于自动发布模式，返回 true，否则返回 false。

configAutoPublish

- **声明：** `public void configAutoPublish( PublishConfig config )`
- **说明：**配置自动发布音视频格式，如果没有配置，则使用默认配置进行发布，默认发布音视频，同时视频使用 PROFILE_360_640P_15 格式。

- **参数：**

参数	类型	说明
config	PublishConfig	发布的配置信息，详情请参见 重要参数。

- **返回值：**无。

getCurrentAutoPublishConfig

- **声明：** `public PublishConfig getCurrentAutoPublishConfig()`
- **说明：**获取当前自动发布的配置信息。
- **参数：**无。
- **返回值：**PublishConfig，发布的配置信息，详情请参见 重要参数。

publish

- **声明：** `public void publish( PublishConfig config )`
- **说明：**发布媒体流。在没有设置自动发布的情况下，可以主动发布媒体流。
- **参数：**

参数	类型	说明
config	PublishConfig	本次发布媒体流的配置信息，详情请参见 重要参数。

- **返回值：**无。

unpublish

- **声明：** `public void unpublish( UnpublishConfig config )`
- **说明：**取消发布媒体流。用户在发布媒体流后，可以在期间取消发布，取消发布后，如果用户想要再次发布流，则需要调用 publish 接口进行重新发布。
- **参数：**

参数	类型	说明
config	UnpublishConfig	需要取消发布的流信息，详情请参见 重要参数。

- **返回值：**无。

updateVideoProfile

- **声明：** `public void updateVideoProfile( VideoProfile videoProfile, int maxBitrate, PublishVideoSource videoSource )`
- **说明：**用户在发布媒体流中间阶段，可以动态切换发布的分辨率。
- **参数：**

参数	类型	说明
videoProfile	VideoProfile	目标分辨率的配置值，详情请参见 重要参数。
maxBitrate	int	目标分辨率对应的最大码率，如果设置为 0（或者 < 0），那么则使用默认的码率，否则，使用设置的最大的码率，单位为 kbps。
videoSource	PublishVideoSource	切换目标分辨率的媒体源。

- **返回值：**无。

pushCustomVideoData

- **声明：** `public void pushCustomVideoData(byte[] bytes, int width, int height, int rotation )`
- **说明：**自定义推流图像数据输入，Android 仅支持 nv21 数据。
- **参数：**

参数	类型	说明
bytes	byte[]	图像数据，NV21 格式。
width	int	图像宽
height	int	图像高
rotation	int	图像旋转角度

- **返回值：**无。

isAutoSubscribe

- **声明：** `public boolean isAutoSubscribe()`
- **说明：**查询是否处于自动订阅模式。
- **参数：**无。
- **返回值：**如果处于自动订阅模式，则返回 true，否则返回 false。

subscribe

- **声明：** `public void subscribe( SubscribeConfig config )`
- **说明：**订阅某路媒体流。
- **参数：**

参数	类型	说明
config	SubscribeConfig	被订阅流的配置信息，详情请参见 重要参数。

- **返回值：**无。

unsubscribe

- **声明：** `public void unsubscribe( UnsubscribeConfig config )`
- **说明：**取消订阅某路媒体流。
- **参数：**

参数	类型	说明
config	UnsubscribeConfig	被订阅流的配置信息，详情请参见 重要参数。

- **返回值：**无。

muteLocalVideo

- **声明：** `public void muteLocalVideo( LocalVideoInfo info, boolean muted )`
- **说明：**暂停本地视频流的发布。调用该接口，可以暂停继续本地视频流的推送，适用用临时暂停，后面继续的场景。
- **参数：**

参数	类型	说明
info	LocalVideoInfo	本地视频流信息，详情请参见 重要参数。
muted	boolean	是否暂停本地视频流的发布： ◦ true：表示暂停本地视频推送。 ◦ false：表示继续本地视频推送。

- **返回值：**无。

muteAllLocalVideo

- **声明：** `public void muteAllLocalVideo( boolean muted )`
- **说明：**暂停所有本地视频流的发布。当本地存在多路流需要同时暂停推送时，可以调用该接口进行暂停继续推送操作。
- **参数：**

参数	类型	说明

参数	类型	说明
muted	boolean	是否暂停所有本地视频流的发布： ◦ true：表示暂停所有本地视频流推送。 ◦ false：表示继续所有本地视频流推送。

- **返回值：**无。

switchCamera

- **声明：** `public void switchCamera()`
- **说明：**切换手机上的前后摄像头。
- **参数：**无。
- **返回值：**无。

muteRemoteVideo

- **声明：** `public void muteRemoteVideo( RemoteVideoInfo info, boolean muted )`
- **说明：**暂停远端视频流的订阅。调用该接口，可以暂停继续远端视频流的订阅，适用于临时暂停，后面继续的场景。
- **参数：**

参数	类型	说明
info	RemoteVideoInfo	远端视频流信息，详情请参见 重要参数。
muted	boolean	是否暂停所有远端视频流的订阅： ◦ true：表示暂停远端视频流的订阅。 ◦ false：表示继续远端视频流的订阅。

- **返回值：**无。

muteAllRemoteVideo

- **声明：** `public void muteAllRemoteVideo( boolean muted )`
- **说明：**暂停所有远端视频流的播放。当本地存在多路流需要同时暂停订阅时，可以调用该接口进行操作。
- **参数：**

参数	类型	说明
muted	boolean	是否暂停所有远端视频流的播放： ◦ true：表示暂停所有的远端视频流的播放。 ◦ false：表示继续所有的远端视频流的播放。

- **返回值：**无。

muteLocalAudio

- **声明：** `public void muteLocalAudio( LocalAudioInfo info, boolean muted )`
- **说明：** 暂停本地音频流推送。
- **参数：**

参数	类型	说明
info	LocalAudioInfo	本地音频流信息，详情请参见 重要参数。 在摄像头模式下，可以填入 null。
muted	boolean	是否暂停本地音频流推送： ◦ true：表示暂停推送音频流。 ◦ false：表示继续推送音频流。

- **返回值：** 无。

muteAllLocalAudio

- **声明：** `public void muteAllLocalAudio( boolean muted )`
- **说明：** 暂停所有本地音频流推送。当本地存在多路音频流推送时，可以使用该接口进行暂停继续操作。
- **参数：**

参数	类型	说明
muted	boolean	是否暂停所有本地音频流推送 ◦ true：表示暂停推送所有音频流。 ◦ false：表示继续推送所有音频流。

- **返回值：** 无。

muteRemoteAudio

- **声明：** `public void muteRemoteAudio( RemoteAudioInfo info, boolean muted )`
- **说明：** 暂停远端音频流订阅。
- **参数：**

参数	类型	说明
info	RemoteAudioInfo	远端音频流信息，详情请参见 重要参数。
muted	boolean	是否暂停远端音频流订阅： ◦ true：表示暂停订阅音频流。 ◦ false：表示继续订阅音频流。

- **返回值：**无。

muteAllRemoteAudio

- **声明：** `public void muteAllRemoteAudio( boolean muted )`
- **说明：**暂停远端所有音频流订阅。当订阅了远端多路音频流是，可以使用该接口进行暂停继续操作。
- **参数：**

参数	类型	说明
muted	boolean	是否暂停远端所有音频流订阅： ◦ true：表示暂停所有音频流订阅。 ◦ false：表示继续所有音频流订阅。

- **返回值：**无。

enableSpeaker

- **声明：** `public void enableSpeaker( boolean enable )`
- **说明：**切换扬声器输出。
- **参数：**

参数	类型	说明
enable	boolean	是否切换为扬声器输出： ◦ true：切换至扬声器输出。 ◦ false：切换到非扬声器输出。

- **返回值：**无。

startCameraPreview

- **声明：** `public void startCameraPreview()`
- **说明：**开始相机预览。相机预览需要 camera 权限，需提前准备硬件资源。
- **参数：**无。
- **返回值：**无。

stopCameraPreview

- **声明：** `public void stopCameraPreview()`
- **说明：**停止本地相机预览。相机预览停止后，会触发停止发布操作。如果想要再次发布，需先启动相机预览，成功后方可进行发布操作。
- **参数：**无。
- **返回值：**无。

getRemoteUsers

- **声明：** `public List<ParticipantInfo> getRemoteUsers()`
- **说明：**获取房间中其他与会者的信息。

- **参数：**无。
- **返回值：**返回房间中其他成员列表，ParticipantInfo 详情请参见 重要参数。

setLogger

- **声明：** `public void setLogger( Logger logger )`
- **说明：**设置自定义日志输出。
- **参数：**

参数	类型	说明
logger	Logger	用户使用自定义日志输出的接口，需要实现该接口。

- **返回值：**无。

invite

- **声明：** `public String invite( InviteParams params )`
- **说明：**邀请对方加入。
- **参数：**

参数	类型	说明
params	InviteParams	邀请的配置信息，详情请参见 重要参数。

- **返回值：**String，本次邀请的 ID，后续的结果通知，以此 ID 进行标识。

replyInvite

- **声明：** `public void replyInvite(ReplyOfInviteParam params)`
- **说明：**回复对方邀请。
- **参数：**

参数	类型	说明
params	ReplyOfInviteParam	回复邀请的配置信息，详情请参见 重要参数。

- **返回值：**无。

sendMessage

- **声明：** `public void sendMessage( Msg4Send msg4Send )`
- **说明：**发送文本信息。
- **参数：**

参数	类型	说明
msg4Send	Msg4Send	发送文本信息的配置信息，详情请参见 重要参数。

- **返回值：**无。

snapshot

- **声明：** `public void snapshot( FeedInfo info )`
- **说明：**针对某路流进行截帧。
- **参数：**

参数	类型	说明
info	FeedInfo	截帧流的信息，详情请参见 重要参数。

- **返回值：**无。

onRequestPermissionsResult

- **声明：** `public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)`
- **说明：**透传权限申请结果。
- **参数：**参考 Android 权限申请，直接将对应的结果透传到底层进行权限处理。如果需要处理权限申请，则业务需要将该权限申请结果通过该接口通知到 SDK 层进行对应的处理。
- **返回值：**无。

onActivityResult

- **声明：** `public void onActivityResult(int requestCode, int resultCode, Intent data)`
- **说明：**屏幕共享权限申请 activity 的结果回调。
- **参数：**参考获取 Android activity 的结果回调，直接将结果透传到 SDK 底层进行处理。
- **返回值：**无。

6.1.4. 回调函数

本文介绍音视频通话 API 在接入 Android 时涉及的回调函数。

回调类型	函数	说明
全局错误事件	void onError(int errorCode, String errorMessage, Bundle extra);	SDK 不可恢复的错误，一定要监听，并根据不同情况有适当的界面提示。仅包含全局错误，不包括发布和订阅发生的错误。

回调类型	函数	说明
房间事件	void onRoomInfo(RoomInfo info);	创建房间成功的回调，包含房间信息，如果房间创建失败，则通过 onError 回调对应的错误。
	void onEnterRoom(int result);	加入房间的回调，`0` 为成功，`非 0` 为失败，详情请参见 错误码。
	void onLeaveRoom(int result);	退出房间回调，`0` 为成功，`非 0` 为失败，详情请参见 错误码。
	void onRecordInfo(String recordId)	录制结果的回调，返回录制 ID，通过该 ID 可以使用额外的查询接口进行录制文件操作。
预览事件	void onCameraPreviewInfo(ARTVCView view);	预览成功的回调，返回一个 view，业务可以将该 view 布局到界面上进行展示。如果预览失败，则通过 onError 返回错误信息。
	void onCameraPreviewFirstFrame();	预览首帧的回调。
	void onCameraPreviewStop();	预览结束的回调。
发布订阅事件	void onPublishEvent(PublishConfig config, PublishEventCode eventCode, String eventDetail, FeedInfo feedInfo);	发布相关事件的通知，业务调用 publish 发布接口后，相关的结果从这里通知出去。
	void onUnpublishEvent(UnpublishConfig config, PublishEventCode eventCode, String eventDetail);	取消发布事件的通知，业务调用 unpublish 取消发布接口后，相关的结果从这里通知出去。
	void onSubscribeEvent(FeedInfo feedInfo, SubscribeEventCode eventCode, String eventDetail, ARTVCView view);	订阅事件的通知，业务调用 subscribe 订阅接口后，相关的结果从这里通知出去。渲染视频的 view 将从后续的第一帧数据的回调通知中返回。
	void onUnsubscribeEvent(FeedInfo feedInfo, SubscribeEventCode eventCode, String eventDetail);	取消订阅事件通知，当用户调用 unsubscribe 取消订阅某路媒体流时，相关结果从这里通知出去。

回调类型	函数	说明
房间成员事件	void onParticipantsEnter(List infos);	用户进入房间的通知。有新用户加入房间后，会将该用户的信息推送给房间的其他用户。同时，新用户也会通过该通知收到房间其他成员的信息。
	void onParticipantsLeave(List<ParticipantLeaveInfo>uidList);	用户退出房间的通知。有用户退出房间时，会将该信息推送给房间的其他用户。
	void onPublishNotify(FeedInfo publisherInfo);	用户新发布的通知。
	void onUnpublishNotify(FeedInfo publisherInfo);	用户取消发布的通知。
	void onSubscribeNotify(FeedInfo publisherInfo, ParticipantInfo subscriberInfo);	用户订阅的通知。
	void onUnsubscribeNotify(FeedInfo publisherInfo, ParticipantInfo subscribeInfo);	用户取消订阅的通知。
视频开始渲染事件	void onRemoteViewFirstFrame(FeedInfo publisherInfo, ARTVCView view);	订阅的视频流开始渲染通知事件。
	void onRemoteViewStop(FeedInfo feedInfo, ARTVCView view);	订阅的视频流停止渲染通知事件。
当前网络状态	void onCurrentNetworkType(int type);	手机当前网络状态，当手机网络状态有变化的时候，会通过该回调通知到上层。同时，在用户通过 `setRtcListenerAndHandler` 接口设置整体回调的时候，也会通过该接口通知业务当前网络类型。
	void onBandwidthImportanceChangeNotify(boolean isLow, double currentBandwidth, FeedInfo feedInfo);	带宽变化通知，当带宽从正常变化到弱网，或者从弱网重新回到正常时，会回调该通知。

回调类型	函数	说明
当前音频输出模式	void onCurrentAudioPlayoutMode(int mode);	手机当前声音输出模式，当手机声音输出模式有变化的时候，会通过该回调通知到上层。同时，在用户通过 `setRtcListenerAndHandler` 接口设置整体回调的时候，也会通过该接口通知业务当前声音输出模式。
截帧结果回调	void onSnapShotComplete(Bitmap image, FeedInfo feedInfo);	执行截帧后，在获取到结果后，会通过该回调通知。
调试信息回调	void onStatisticDebugInfo(StatisticInfoForDebug infoForDebug, FeedInfo feedInfo);	调试统计信息。
	void onRealTimeStatisticInfo(RealTimeStatisticReport report, FeedInfo feedInfo);	调试实时质量信息。
文本交互回调	void onMsgReceive(Msg4Receive msg4Receive);	接收到文本信息，详情请参见 重要参数。
	void onMsgSend(int error, String errorMessage, long msgId);	文本发送结果。
邀请交互回调	void onInviteResponse(String inviteTaskId, int code, String msg);	发送邀请的结果，code 值为 0 时则发送成功。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。
	void onReplyResponse(String inviteTaskId, int code, String msg);	回复邀请的结果，code 值为 0 时则回复成功。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。
	void onReplyOfInviteNotify(ReplyOfInviteInfo replyInfo);	发送邀请之后，对方回复的结果，可以是接受或者拒绝之类的结果，详情请参见 重要参数。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。

回调类型	函数	说明
自定义推流图像本地预览 view 交互回调	void onCustomPublishPreviewInfo(PublishVideoSource videoSource, ARTVCView view);	自定义推流的本地预览 view，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注它。
	void onCustomPublishPreviewFirstFrame(PublishVideoSource videoSource);	自定义推流的本地预览的第一帧显示事件，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注。
	void onCustomPublishPreviewStop(PublishVideoSource videoSource);	自定义推流的本地预览停止显示事件，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注。

onError

- **声明：** `void onError(int errorCode, String errorMessage, Bundle extra );`
- **说明：** 错误回调，SDK 不可恢复的错误，必须监听，并按不同的毁掉结果给用户不同的界面提示。仅包含全局错误，不包括发布和订阅发生的错误。
- **参数：**

参数	类型	说明
errorCode	int	错误码，更多信息请参见 错误码。
errorMessage	String	错误信息。
extra	Bundle	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

- **返回值：** 无。

onRoomInfo

- **声明：** `void onRoomInfo( RoomInfo info );`
- **说明：** 创建房间成功回调，包含房间信息。如果房间创建失败，则通过 onError 回调对应的错误。
- **参数：**

参数	类型	说明
info	RoomInfo	房间信息，详情请参见 重要参数。

- **返回值：**无。

onEnterRoom

- **声明：** `void onEnterRoom( int result );`
- **说明：**加入房间的回调，0 为成功，非 0 为失败，详情请参见 错误码。
- **参数：**

参数	类型	说明
result	int	返回结果，0 为成功，非 0 为失败，更多信息请参见 错误码。

- **返回值：**无。

onLeaveRoom

- **声明：** `void onLeaveRoom( int result );`
- **说明：**退出房间的回调，0 为成功，非 0 为失败，详情请参见 错误码。
- **参数：**

参数	类型	说明
result	int	返回结果，0 为成功，非 0 为失败，更多信息请参见 错误码。

- **返回值：**无。

onRecordInfo

- **声明：** `void onRecordInfo( String recordId );`
- **说明：**录制结果的回调，返回录制 ID，通过该 ID 可以使用额外的查询接口进行录制文件操作。
- **参数：**

参数	类型	说明
recordId	String	录制 ID。

- **返回值：**无。

onCameraPreviewInfo

- **声明：** `void onCameraPreviewInfo(ARTVCView view );`

- **说明：**预览成功的回调，返回一个 view，业务可以将该 view 布局到界面上进行展示。如果预览失败，则通过 onError 返回错误信息。
- **参数：**

参数	类型	说明
view	ARTVCView	视频渲染控件。

- **返回值：**无。

onCameraPreviewFirstFrame

- **声明：** `void onCameraPreviewFirstFrame();`
- **说明：**预览首帧的回调。
- **参数：**无。
- **返回值：**无。

onCameraPreviewStop

- **声明：** `void onCameraPreviewStop();`
- **说明：**预览结束的回调。
- **参数：**无。
- **返回值：**无。

onPublishEvent

- **声明：** `void onPublishEvent(PublishConfig config, PublishEventCode eventCode, String eventDetail, FeedInfo feedInfo );`
- **说明：**发布相关事件的通知，业务调用 publish 发布接口后，相关的结果从这里通知出去。
- **参数：**

参数	类型	说明
config	PublishConfig	本次发布的配置信息。
eventCode	PublishEventCode	事件码，详情请参见 重要参数。
eventDetail	String	事件信息。
feedInfo	FeedInfo	可以正常发布时，返回这个发布的 feedInfo 信息，即当 eventCode == UBLISH_START 时，返回 feedInfo 信息，其它情况则返回 null。

- **返回值：**无。

onUnpublishEvent

- **声明：** `void onUnpublishEvent( UnpublishConfig config, PublishEventCode eventCode, String eventDetail );`
- **说明：** 取消发布事件的通知，业务调用 unpublish 取消发布接口后，相关的结果从这里通知出去。
- **参数：**

参数	类型	说明
config	UnpublishConfig	本次取消发布的配置信息。
eventCode	PublishEventCode	事件码，详情请参见 重要参数。
eventDetail	String	事件信息。

- **返回值：** 无。

onSubscribeEvent

- **声明：** `void onSubscribeEvent(FeedInfo feedInfo, SubscribeEventCode eventCode, String eventDetail, ARTVCView view );`
- **说明：** 订阅事件的通知，业务调用 subscribe 订阅接口后，相关的结果从这里通知出去。渲染视频的 view 将从后续的第一帧数据的回调通知中返回。
- **参数：**

参数	类型	说明
feedInfo	FeedInfo	被订阅流的发布者信息，包含发布者 ID 和对应的流 ID 信息，详情请参见 重要参数。
eventCode	SubscribeEventCode	事件码，详情请参见 重要参数。
eventDetail	String	事件信息。
view	ARTVCView	为 null。

- **返回值：** 无。

onUnsubscribeEvent

- **声明：** `void onUnsubscribeEvent( FeedInfo feedInfo, SubscribeEventCode eventCode, String eventDetail );`
- **说明：** 取消订阅事件的通知，当用户调用 unsubscribe 取消订阅某路媒体流时，相关结果从这里通知出去。
- **参数：**

参数	类型	说明
feedInfo	FeedInfo	被订阅流的发布者信息，包含发布者 ID 和对应的流 ID 信息，详情请参见 重要参数。
eventCode	SubscribeEventCode	事件码，详情请参见 重要参数。
eventDetail	String	事件信息。

- **返回值：**无。

onParticipantsEnter

- **声明：** `void onParticipantsEnter( List infos );`
- **说明：**用户进入房间的通知。有新用户加入房间后，会将该用户的信息推送给房间的其他用户，同时新用户也会通过该通知收到房间其他成员的信息。
- **参数：**

参数	类型	说明
infos	List	成员数组信息，每个成员包含成员 ID，该成员所有的发布流和该成员所有的订阅流，详情请参见 重要参数。

- **返回值：**无。

onParticipantsLeave

- **声明：** `void onParticipantsLeave( List<ParticipantLeaveInfo> uidList);`
- **说明：**用户退出房间的通知。有用户退出房间时，会将该信息推送给房间的其他用户。
- **参数：**

参数	类型	说明
uidList	List<ParticipantLeaveInfo>	成员 uid 数组信息。

- **返回值：**无。

onPublishNotify

- **声明：** `void onPublishNotify(FeedInfo publisherInfo );`
- **说明：**用户新发布的通知。
- **参数：**

参数	类型	说明
publisherInfo	FeedInfo	新发布流的信息，包含发布者的 ID 和对应新发布流的流 ID 信息，详情请参见 重要参数。

- **返回值：**无。

onUnpublishNotify

- **声明：** `void onUnpublishNotify(FeedInfo publisherInfo );`
- **说明：**用户取消发布的通知。
- **参数：**

参数	类型	说明
publisherInfo	FeedInfo	取消发布流的信息，包含发布者的 ID 和对应取消发布流的流 ID 信息，详情请参见 重要参数。

- **返回值：**无。

onSubscribeNotify

- **声明：** `void onSubscribeNotify(FeedInfo publisherInfo, ParticipantInfo subscriberInfo );`
- **说明：**用户订阅的通知。
- **参数：**

参数	类型	说明
publisherInfo	FeedInfo	被订阅流的发布者信息，包含发布者 ID 和对应的流 ID 信息，详情请参见 重要参数。
subscribeInfo	ParticipantInfo	订阅者的 uid 信息。

- **返回值：**无。

onUnsubscribeNotify

- **声明：** `void onUnsubscribeNotify(FeedInfo publisherInfo, ParticipantInfo subscribeInfo );`
- **说明：**用户取消订阅的通知。
- **参数：**

参数	类型	说明

参数	类型	说明
publisherInfo	FeedInfo	被订阅流的发布者信息，包含发布者 ID 和对应的流 ID 信息，详情请参见 重要参数。
subscribeInfo	ParticipantInfo	订阅者 uid 信息。

- **返回值：**无。

onRemoteViewFirstFrame

- **声明：** `void onRemoteViewFirstFrame( FeedInfo publisherInfo );`
- **说明：**订阅的视频流开始渲染的通知事件。
- **参数：**

参数	类型	说明
publisherInfo	FeedInfo	被订阅流的发布者信息，包含发布者 ID 和对应的流 ID 信息，详情请参见 重要参数。

- **返回值：**无。

onRemoteViewStop

- **声明：** `void onRemoteViewStop( FeedInfo feedInfo, ARTVCView view );`
- **说明：**订阅的视频流停止渲染的通知事件。
- **参数：**

参数	类型	说明
feedInfo	FeedInfo	被订阅流的发布者信息，包含发布者ID 和对应的流 ID 信息，详情请参见 重要参数。
view	ARTVCView	对应视频渲染的 view。

- **返回值：**无。

onCurrentNetworkType

- **声明：** `void onCurrentNetworkType( int type );`
- **说明：**手机当前网络状态。当手机网络状态有变化时，会通过该回调通知到上层。同时，在用户通过 `setRtcListenerAndHandler` 接口设置整体回调的时候，也会通过该接口通知业务当前网络类型。
- **参数：**

参数	类型	说明
type	int	网络状态： 0：无网络可用。 1：Wi-Fi 网络。 2：移动网络。在该网络下，需要业务通知用户会消耗用户流量。

- **返回值：**无。

onBandwidthImportanceChangeNotify

- **声明：** `void onBandwidthImportanceChangeNotify( boolean isLow, double currentBandwidth, FeedInfo feedInfo );`
- **说明：**带宽变化通知，当带宽从正常变化到弱网，或者从弱网从新回到正常时，会回调该通知。
- **参数：**

参数	类型	说明
isLow	boolean	是否为弱网状态。
currentBandwidth	double	当前有效带宽值。
feedInfo	FeedInfo	带宽变化对应的流信息。

- **返回值：**无。

onCurrentAudioPlayoutMode

- **声明：** `void onCurrentAudioPlayoutMode( int mode );`
- **说明：**手机当前声音输出模式，当手机声音输出模式有变化时，会通过该回调通知到上层。同时，在用户通过 `setRtcListenerAndHandler` 接口设置整体回调的时候，也会通过该接口通知业务当前声音输出模式。
- **参数：**

参数	类型	说明
mode	int	音频输出模式： 1：EARPIECE，听筒输出。 2：SPEAKERPHONE，外放输出。 3：WIREDHEADSET，耳机输出。 4：BLUETOOTH，蓝牙输出。 5：NONE，没有输出。

- **返回值：**无。

onSnapShotComplete

- **声明：** `void onSnapShotComplete( Bitmap image, FeedInfo feedInfo );`
- **说明：**执行截帧后，在获取到结果后，会通过该回调通知。
- **参数：**

参数	类型	说明
image	Bitmap	截帧后的图像。
feedInfo	FeedInfo	截帧对应的流信息，可以是对方，也可以是本端。

- **返回值：**无。

onStatisticDebugInfo

- **声明：** `void onStatisticDebugInfo( StatisticInfoForDebug infoForDebug, FeedInfo feedInfo );`
- **说明：**调试统计信息。
- **参数：**

参数	类型	说明
infoForDebug	StatisticInfoForDebug	统计信息，详情请参见 重要参数。
feedInfo	FeedInfo	对应的流信息，可以是对方，也可以是本端。

- **返回值：**无。

onRealTimeStatisticInfo

- **声明：** `void onRealTimeStatisticInfo( RealTimeStatisticReport report, FeedInfo feedInfo );`
- **说明：**调试实时质量信息。
- **参数：**

参数	类型	说明
report	RealTimeStatisticReport	实时质量信息，详情请参见 重要参数。
feedInfo	FeedInfo	对应的流信息，可以是对方，也可以是本端。

- **返回值：**无。

onMsgReceive

- **声明：** `void onMsgReceive(Msg4Receive msg4Receive);`
- **说明：** 接收到文本信息。在 `AlipayRtcEngineIMListener` 回调中，如果不使用文本发送功能，则可以不用设置该回调。
- **参数：**

参数	类型	说明
msg4Receive	Msg4Receive	更多信息请参见 重要参数。

- **返回值：** 无。

onMsgSend

- **声明：** `void onMsgSend( int error, String errorMessage, long msgId );`
- **说明：** 文本发送结果。在 `AlipayRtcEngineIMListener` 回调中，如果不使用文本发送功能，则可以不用设置该回调。
- **参数：**

参数	类型	说明
error	int	`0` 为正常，其它则为失败。
errorMessage	String	如果 error 不为 `0`，则这里是对应的失败信息说明。
msgId	long	对应的消息 ID。

- **返回值：** 无。

onInviteResponse

- **声明：** `void onInviteResponse( String inviteTaskId, int code, String msg );`
- **说明：** 发送邀请的结果，code=0 则发送成功。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。
- **参数：**

参数	类型	说明
inviteTaskId	String	邀请任务 ID。用于标识是哪次邀请调用。

参数	类型	说明
code	int	邀请错误码，详细说明如下： ◦ 0：邀请成功。 ◦ 1：参数错误。 ◦ 2：房间非法。 ◦ 3：已有正在进行的邀请。 ◦ 4：服务端返回邀请失败。 ◦ 5：邀请超时。 ◦ 6：预留错误码。
msg	String	服务端返回的详细信息。

- **返回值：**无。

onReplyResponse

- **声明：** `void onReplyResponse( String inviteTaskId, int code, String msg );`
- **说明：**回复邀请的结果，code 值为 0 时则回复成功。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。
- **参数：**

参数	类型	说明
inviteTaskId	String	邀请任务 ID。用于标识是哪次邀请调用。
code	int	邀请错误码，详细说明如下： ◦ 0：回复成功。 ◦ 1：参数错误。 ◦ 3：上次调用还未完成。 ◦ 4：服务端返回请求失败。 ◦ 5：调用超时。 ◦ 6：预留错误码。
msg	String	服务端返回的详细信息。

- **返回值：**无。

onReplyOfInviteNotify

- **声明：** `void onReplyOfInviteNotify(ReplyOfInviteInfo replyInfo );`
- **说明：**发送邀请之后，对方回复的结果，可以是接受或者拒绝之类的结果。在 `AlipayRtcEngineInviteListener` 中，如果不使用邀请功能，可以不用设置和关注。

- **参数：**

参数	类型	说明
replyInfo	ReplyOfInviteInfo	更多信息请参见 重要参数。

- **返回值：**无。

onCustomPublishPreviewInfo

- **声明：** `void onCustomPublishPreviewInfo(PublishVideoSource videoSource, ARTVCView view);`
- **说明：**自定义推流的本地预览 view，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注。
- **参数：**

参数	类型	说明
videoSource	PublishVideoSource	更多信息请参见 重要参数。
view	ARTVCView	更多信息请参见 重要参数。

- **返回值：**无。

onCustomPublishPreviewFirstFrame

- **声明：** `void onCustomPublishPreviewFirstFrame( PublishVideoSource videoSource );`
- **说明：**自定义推流的本地预览的第一帧显示事件，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注。
- **参数：**

参数	类型	说明
videoSource	PublishVideoSource	更多信息请参见 重要参数。

- **返回值：**无。

onCustomPublishPreviewStop

- **声明：** `void onCustomPublishPreviewStop( PublishVideoSource videoSource );`
- **说明：**自定义推流的本地预览停止显示事件，通过 videoSource 区分视频源。在 `AlipayRtcEngineCustomPublishListener` 中，如果不使用自定义推流和屏幕共享，可以不用设置和关注。
- **参数：**

参数	类型	说明
videoSource	PublishVideoSource	更多信息请参见 重要参数。

- **返回值：**无。

6.1.5. 错误码

本文介绍音视频通话 API 在接入 Android 中涉及的错误码。

错误码	错误信息
-103	参数错误
-104	相机权限错误，没有获得相机权限
-105	麦克风权限错误，没有获得麦克风权限
-106	获取手机状态权限失败（电话）
-108	超时错误
-114	房间失效
-115	创建房间失败
-116	加入房间失败
-117	发布失败
-118	订阅失败
-119	取消订阅或取消发布失败
-1002	退出房间失败
-1004	网络错误，无法连接服务器
-1005	room 地址设置无效

错误码	错误信息
-1007	打开摄像头错误
-1008	打开麦克风错误
-1009	机器不支持，可能是其他 SO 库不支持
-1010	屏幕共享能力不支持，例如系统过低
-1011	屏幕共享正在进行中
-1012	屏幕共享启动失败
-1013	屏幕共享中系统出现异常，无法恢复

6.2. iOS API

6.2.1. 重要参数

本文介绍的是音视频通话 API 在接入 iOS 过程中的重要参数。

变量	说明
delegate	设置代理对象。
dynamicConfigProxy	动态配置代理。
uid	设置用户 ID。 说明 目前 uid 只支持英文字母、数字、下划线的组合，且长度不超过 128 个字符。
roomServerCustomUrl	设置服务器地址。
videoProfileType	设置编码分辨率。

autoPublish	设置自动发布。
autoPublishConfig	设置自动发布配置参数。
currentPublishConfig	获取自动订阅配置参数。
autoSubscribe	设置自动订阅。
autoSubscribeOptions	设置自动订阅配置参数。

delegate

- **功能：**设置代理对象。
- **介绍：**通过该字段设置代理对象，weak 持有。只有先设置好 Delegate 才能收到底层事件回调。

dynamicConfigProxy

- **功能：**设置动态配置代理。
- **介绍：**通过该字段设置动态配置代理，可做一些策略动态配置。若没有动态配置需求，可不做设置。

uid

- **功能：**设置用户 ID。
- **介绍：**通过该字段设置用户 ID，调用 API 之前必须先设置好用户 ID。

roomServerCustomUrl

- **功能：**设置服务器地址。
- **介绍：**通过该字段设置服务器地址，调用 API 之前必须先设置好服务器地址。公有云中金区环境与非金区环境的房间服务器地址不同，您可以按您当前的环境选择房间服务器的地址。
  - **非金区：**wss://mrtc.mpaas.cn-hangzhou.aliyuncs.com/ws
  - **金区：**wss://room.mrtc-fin.cn-shanghai.aliyuncs.com/ws

videoProfileType

- **功能：**设置编码分辨率。
- **介绍：**通过该字段设置编码分辨率，调用 API 之前必须先设置好编码分辨率。

autoPublish

- **功能：**设置自动发布。
- **介绍：**通过该字段设置自动发布。可不做设置，默认是自动发布状态。

autoPublishConfig

- **功能：**设置自动发布参数。
- **介绍：**通过该字段设置自动发布配置参数，可设置发布音频、发布视频、编码分辨率和超时等相关操作。可不做设置，底层有默认配置。

currentPublishConfig

- **功能：**获取自动订阅配置参数。

autoSubscribe

- **功能：**设置自动订阅。
- **介绍：**通过该字段设置自动订阅。可不做设置，默认是自动订阅。

autoSubscribeOptions

- **功能：**设置自动订阅配置参数。
- **介绍：**通过该字段设置自动订阅配置参数，可设置是否接受音频，接收视频，超时相关操作。可不做设置，底层有默认配置。

6.2.2. 公共接口

本文介绍的是音视频通话 API 在接入 iOS 中涉及的相关接口。

接口类型	API	说明
房间相关接口	createRoom	创建视频通话房间
	joinRoom	加入视频通话房间
	leaveRoom	离开视频通话房间
邀请应答接口	inviteWith	通知对方加入房间
	replyWith	回复邀请的应答操作
发布订阅接口	publish	发布流
	unpublish	取消发布流
	subscribe	订阅流
	unsubscribe	取消订阅流
摄像头接口	startCameraPreviewUsingBackCamera	开启相机
	stopCameraPreview	停止相机
	switchCamera	切换相机摄像头
	muteMicrophone	麦克风静音

静音接口	muteRemoteAudio	静音单个远端音频
	muteAllRemoteAudios	静音所有远端音频
	muteRemoteVideo	静音单个远端视频
	muteAllRemoteVideos	静音所有远端视频
截图接口	snapshotForFeed	截图
扬声器/听筒接口	currentAudioPlayMode	当前声音播放模式
	switchAudioPlayModeTo	切换听筒扬声器模式
IM 接口	sendMessage	发送 IM 到单个用户
	sendMessage	发送 IM 到多个用户
自定义推流接口	createCustomVideoCapturer	创建自定义推流类
	destroyCustomVideoCapturer	销毁自定义推流类
屏幕共享接口	isScreenCaptureSupported	是否支持屏幕捕捉
	isScreenCaptureStarted	屏幕捕捉是否已启动
	startScreenCaptureWithParams	开始屏幕捕捉
	stopScreenCapture	结束屏幕捕捉

createRoom

- **声明：** `(void)createRoom:(ARTVCCreateRoomParams*) params`
- **说明：** 创建视频通话房间。
- **参数：**

参数	类型	说明	是否可为空
uid	NSString*	用户 UID	NO

bizName	NSString*	主业务名称	NO
signature	NSString*	签名	NO
extraInfo	NSDictionary*	扩展参数	YES

- **返回值：**无。

joinRoom

- **声明：** `(void)joinRoom:(ARTVCJoinRoomParams*) params;`
- **说明：**加入视频通话房间。
- **参数：**

参数	类型	说明	是否可为空
uid	NSString*	用户 UID	NO
bizName	NSString*	主业务名称	NO
signature	NSString*	签名	NO
roomId	NSString*	房间 ID	NO
rtoken	NSString*	房间 Token	NO
extraInfo	NSDictionary*	扩展参数	YES

- **返回值：**无。

leaveRoom

- **声明：** `(void)leaveRoom;`
- **说明：**离开视频通话房间。
- **参数：**无。
- **返回值：**无。

inviteWith

- **声明：** `(void)inviteWith:(ARTVCInviteParams*)params complete:(ARTVCInviteCallback)complete;`
- **说明：**创建视频通话房间后，可调用该接口通知对方来加入房间。C2B 下可邀请 Web 坐席。该接口暂不支持在 C2C 模式下调用。
- **参数：**

- ARTVCInviteParams

参数	类型	说明	是否可为空	默认值
inviteId	NSString	邀请 ID，标识该次邀请。	NO	无
inviteeUid	NSString	对端的 UID	NO	无
inviteType	ARTVCInviteType	目前只支持 ARTVCInviteTypeWebsocket，被邀请方必须是 Web 端。	NO	无
audioEnable	BOOL	邀请对端开启音频。	NO	YES
videoEnable	BOOL	邀请对端开启视频。	NO	YES
timeout	int	超时设置	NO	60
extraInfo	NSDictionary*	扩展参数	YES	无

- ARTVCInviteCallback

```
//error is nil means success,code see ARTVCInviteError
typedef void (^ARTVCInviteCallback)(NSError*_Nullable error);
```

- 错误码定义

```
typedef NS_ENUM(int,ARTVCInviteError){
ARTVCInviteErrorBabParameters                    = 1,
ARTVCInviteErrorNOtHaveValidRoom                 = 2,
ARTVCInviteErrorPreviousInviteUnderProcessing    = 3,
ARTVCInviteErrorRequestFailed                    = 4,
ARTVCInviteErrorTimeout                          = 5,
ARTVCInviteErrorCanceledWhenLeaveRoom            = 6,
};
```

- **返回值：无。**

replyWith

- **声明：** `-(void)replyWith:(ARTVCReplyParams*)params complete:(ARTVCReplyCallback)complete;`
- **说明：应答接口。**
- **参数：**

- ARTVCReplyParams

参数	类型	说明	是否可为空	默认值
uid	NSString	用户 UID	NO	无
inviterUid	NSString	邀请者 UID	NO	无
bizName	NSString	主业务名称	NO	无
roomId	NSString	房间 ID	NO	无
replyType	ARTVCReplyType	应答类型	NO	无
audioEnable	BOOL	开启音频	NO	YES
videoEnable	BOOL	开启视频	NO	YES
timeout	int	超时设置	NO	60
extraInfo	NSDictionary*	扩展参数	YES	无

- ARTVCReplyCallback

```
//error is nil means success,code see ARTVCReplyError
typedef void (^ARTVCReplyCallback)(NSError*_Nullable error);
```

- **错误码定义**

```
typedef NS_ENUM(int,ARTVCReplyError){
ARTVCReplyErrorBabParameters                    = 1,
ARTVCReplyErrorPreviousReplyUnderProcessing     = 3,
ARTVCReplyErrorRequestFailed                    = 4,
ARTVCReplyErrorTimeout                          = 5,
ARTVCReplyErrorCanceledWhenLeaveRoom            = 6,
};
```

- **返回值：**无。

publish

- **声明：** `(void)publish:(ARTVCPublishConfig*)config`
- **说明：**发布流。
- **参数：**ARTVCPublishConfig

参数	类型	说明	默认值
videoEnable	BOOL	是否推视频	YES
audioEnable	BOOL	是否推音频	YES
videoSource	ARTVCVideoSourceType	视频源类型	ARTVCVideoSourceType_Camera
videoProfile	ARTVCVideoProfileType	视频编码分辨率和 FPS。	ARTVCVideoProfileType640x36015Fps
videoCustomWidth	int	自定义视频宽，videoProfile 为 ARTVCVideoProfileType_Custom 时设置有效，不建议使用。	无
videoCustomHeight	int	自定义视频高，videoProfile 为 ARTVCVideoProfileType_Custom 时设置有效，不建议使用。	无
videoCustomFps	int	自定义 FPS，videoProfile 为 ARTVCVideoProfileType_Custom 时设置有效，不建议使用。	无
videoCustomBitrate	int	自定义码率，videoProfile 为 ARTVCVideoProfileType_Custom 时设置有效，不建议使用。	无
timeout	int	超时设置	60
tag	NSString	自定义流 tag。	VIDEO_SOURCE_CAMERA / VIDEO_SOURCE_SCREEN

- **返回值：**无。

unpublish

- **声明：** `-(void)unpublish:(ARTVCUnpublishConfig*)config`
- **说明：**取消发布流。

- **参数：**
  - ARTVCUnpublishConfig

参数	类型	说明
feed	ARTVCFeed*	要取消发布的 Feed。

  - ARTVCFeed

参数	类型	说明	是否可为空
uid	NSString*	Feed 归属的用户 UID。	NO
userType	ARTVCParticipantType	用户类型，默认是普通用户，ARTVCParticipantTypeVirtualVod 表示点播服务对应的虚拟用户。	NO
feedId	NSString*	feed 的标识 ID	NO
tag	NSString*	feed 的标签类型	YES

- **返回值：**无。

subscribe

- **声明：** `(void)subscribe:(ARTVCSubscribeConfig*)config`
- **说明：**订阅流。
- **参数：**
  - ARTVCSubscribeConfig

参数	类型	说明	是否可为空
feed	ARTVCFeed*	要订阅的 feed	NO
options	ARTVCSubscribeOptions*	订阅参数	NO

- ARTVCSubscribeOptions

参数	类型	说明	是否可为空
receiveAudio	BOOL	是否订阅音频流	YES
receiveVideo	BOOL	是否订阅视频流	YES
timeout	int	超时设置	60

- **返回值：**无。

unsubscribe

- **声明：** `(void)unsubscribe:(ARTVCUnsubscribeConfig*)config`
- **说明：**取消订阅流。
- **参数：**ARTVCUnsubscribeConfig

参数	类型	说明
feed	ARTVCFeed*	要取消订阅的 feed

- **返回值：**无。

startCameraPreviewUsingBackCamera

- **声明：** `(void)startCameraPreviewUsingBackCamera:(BOOL)usingBackCamera`
- **说明：**开启相机。
- **参数：**usingBackCamera 为 YES 时使用后置摄像头，否则使用前置摄像头。
- **返回值：**无。

stopCameraPreview

- **声明：** `(void)stopCameraPreview`
- **说明：**停止使用相机。
- **参数：**无。
- **返回值：**无。

switchCamera

- **声明：** `(void)switchCamera`
- **说明：**切换相机摄像头。
- **参数：**无。
- **返回值：**无。

muteMicrophone

- **声明：** `(void)muteMicrophone:(BOOL)muted`

- **说明：**麦克风静音。
- **参数：**muted 为 YES 时静音麦克风，否则取消静音麦克风。
- **返回值：**无。

muteRemoteAudio

- **声明：** `(void)muteRemoteAudio:(BOOL)muted forFeed:(ARTVCFeed*)feed`
- **说明：**对单个远端音频静音。
- **参数：**静音传入 feed 中的音频流。muted 为 YES 时静音，否则取消静音。feed 不能为空。
- **返回值：**无。

muteAllRemoteAudios

- **声明：** `(void)muteAllRemoteAudios:(BOOL)muted`
- **说明：**静音所有远端音频。
- **参数：**静音所有订阅 feed 中的音频流，muted 为 YES 时静音，否则取消静音。
- **返回值：**无。

muteRemoteVideo

- **声明：** `(void)muteRemoteVideo:(BOOL)muted forFeed:(ARTVCFeed*)feed`
- **说明：**静音单个远端视频。
- **参数：**静音传入 feed 中的视频流。Muted 为 YES 时做静音处理，否则取消。Feed 不能为空。
- **返回值：**无。

muteAllRemoteVideos

- **声明：** `(void)muteAllRemoteVideos:(BOOL)muted`
- **说明：**静音所有远端视频。
- **参数：**静音所有订阅 feed 中的视频流。muted 为 YES 时做静音处理，否则取消。
- **返回值：**无。

snapshotForFeed

- **声明：** `(void)snapshotForFeed:(ARTVCFeed*)feed complete:(void(^)(UIImage* image))complete`
- **说明：**截图接口。截取视频帧返回 `.jpg` 格式的 UIImage，支持截取本地和对端任意 feed 的视频帧数据信息。
- **参数：**

- ARTVCFeed

参数	类型	说明	是否可为空
uid	NSString*	Feed 归属的用户 UID	NO
userType	ARTVCParticipantType	用户类型，默认是普通用户，ARTVCParticipantTypeVirtualVod 表示点播服务对应的虚拟用户。	NO
feedId	NSString*	feed 的标识 ID	NO
tag	NSString*	feed 的标签类型	YES

- complete

参数	类型	说明	是否可为空
complete	ARTVCIMCallback	截图图片回调。若为空表示截图失败。	YES

- **返回值：**无。

currentAudioPlayMode

- **声明：** `(ARTVCAudioPlayMode)currentAudioPlayMode`
- **说明：**当前声音的播放模式。
- **参数：**无。
- **返回值：**无。

switchAudioPlayModeTo

- **声明：** `(void)switchAudioPlayModeTo:(ARTVCAudioPlayMode)audioPlayMode complete:(void(^_Nullable)(NSError*_Nullable error))callback`
- **说明：**切换听筒扬声器模式。耳机模式下不作处理。
- **参数：**

- ARTVCAudioPlayMode

```
/**
* 声音播放模式
*/
typedef NS_ENUM(NSUInteger,ARTVCAudioPlayMode) {
ARTVCAudioPlayModeInit = 0,  //初始化状态
ARTVCAudioPlayModeReceiver,  //听筒
ARTVCAudioPlayModeSpeaker,   //扬声器
ARTVCAudioPlayModeHeadphone, //耳机
ARTVCAudioPlayModeBluetooth //蓝牙设备
};
```

- callback 回调解释：当 error 为空表示切换成功，否则切换失败。

- **返回值：**无。

sendMessage

- **声明：** `(void)sendMessage:(ARTVCIMMessage*)message toPariticipant:(NSString*)pariticipant complete:(ARTVCIMCallback)complete`
- **说明：**发送 IM 到单个用户。
- **参数：**

参数	类型	说明	是否可为空
message	ARTVCIMMessage*	IM 消息	NO
pariticipant	NSString*	用户 UID	NO
complete	ARTVCIMCallback	消息发送结果回调	YES

- ARTVCIMMessage

参数	类型	说明	是否可为空
msg	NSString*	消息内容	NO
msgId	NSUInteger	消息 ID	YES
timestamp	NSTimeInterval	消息时间戳	YES

- ARTVCIMCallback

```
//error is nil means success
typedef void (^ARTVCIMCallback)(NSError* error);
```

- **返回值：**无。

sendMessage

- **声明：** `(void)sendMessage:(ARTVCIMMessage*)message toPariticipants:(NSArray<NSString*>*)pariticipants complete:(ARTVCIMCallback)complete`
- **说明：** 发送 IM 到多个用户。
- **参数：**

参数	类型	说明	是否可为空
message	ARTVCIMMessage *	IM 消息	NO
pariticipants	NSArray<NSString>	用户 UID 数组	NO
complete	ARTVCIMCallback	消息发送结果回调	YES

- **返回值：** 无。

createCustomVideoCapturer

- **声明：** `-(ARTVCCustomVideoCapturer*)createCustomVideoCapturer:(ARTVCCreateCustomVideoCaputurerParams*)params;`
- **说明：** 创建自定义推流类。
- **参数：**

参数	类型	说明	是否可为空
params	ARTVCCreateCustomVideoCaputurerParams *	创建自定义推流类参数	NO

ARTVCCreateCustomVideoCaputurerParams 定义：

参数	类型	说明	是否可为空
provideRenderView	BOOL	是否需要提供渲染 view	NO

- **返回值：** 无。

destroyCustomVideoCapturer

- **声明：** `-(void)destroyCustomVideoCapturer;`
- **说明：** 销毁自定义推流类。
- **参数：** 无。
- **返回值：** 无。

isScreenCaptureSupported

- **声明：** `-(BOOL)isScreenCaptureSupported;`
- **说明：** 是否支持屏幕捕捉，需注意 iOS11 以及以上才支持。
- **参数：** 无。
- **返回值：** 返回值为 YES 时表示支持屏幕捕捉，返回值为 NO 时表示不支持屏幕捕捉。

isScreenCaptureStarted

- **声明：** `-(BOOL)isScreenCaptureStarted;`
- **说明：** 屏幕捕捉是否已启动。
- **参数：** 无。
- **返回值：** 无。

startScreenCaptureWithParams

- **声明：** `-(void)startScreenCaptureWithParams:(ARTVCCreateScreenCaputurerParams*)params complete:(ARTVCErrorCallback)callback;`
- **说明：** 开始屏幕捕捉。
- **参数：**

参数	类型	说明	是否可为空
params	ARTVCCreateCustomVideoCaputurerParams *	开始屏幕捕捉参数	NO
callback	ARTVCErrorCallback	开始屏幕捕捉结果回调	YES

ARTVCCreateScreenCaputurerParams 定义：

参数	类型	说明	是否可为空
provideRenderView	BOOL	是否需要提供渲染 view	NO

- **返回值：** 无。

stopScreenCapture

- **声明：** `-(void)stopScreenCapture;`
- **说明：** 结束屏幕捕捉。
- **参数：** 无。
- **返回值：** 无。

6.2.3. 回调函数

本文介绍的是音视频通话 API 在接入 iOS 中涉及的回调函数。

didReceiveLocalFeed

> **说明**
>
> 每一次本地发布，只返回一次。
>
> - **视频通话场景** 开启摄像头会触发回调；publish 也会触发回调。SDK 在底层做了限制处理，对于同一次发布，如果摄像头开启时已经返回了 feed，那么 publish 时不再返回。
> - **音频通话场景** 不存在摄像头开启操作，在 publish 时会触发一次回调。

回调示例：

```
typedef NS_ENUM(int,ARTVCFeedType){
  ARTVCFeedTypeRemoteFeed                = 0,
  //use builtin camera and microphone
  ARTVCFeedTypeLocalFeedDefault          = 1,
  ARTVCFeedTypeLocalFeedCustomVideo      = 2,
  ARTVCFeedTypeLocalFeedScreenCapture    = 3,
};
```

6.3. Web API

6.3.1. Web SDK 发布说明

本文介绍 Web SDK 各版本的发布特性。

V 1.5.0（2022-04-25）

- 发布参数支持降级策略：流畅度优先或清晰度优先，参见 InitRoomConfig 和 Publish 接口下的 degradationType 字段。
- 共享屏幕：支持 tab 中的音频（仅限 tab），参见 InitRoomConfig 和 Publish 接口下的 enableDesktopAudio 字段。
- 共享屏幕：优化桌面共享关闭事件监听逻辑。
- 修复 Safari 下的报错问题：RTCRtpSender.getCapabilities。
- 音视频质量统计：支持部分手机浏览器。

V 1.4.9（2022-01-21）

- 优化摄像头、麦克风在通话时的状态监听逻辑。
- 修复 `RTCIceServer.url` 已废弃问题。
- 废弃 `OnGetSign` 接口，改为业务主动传 `sign`（兼容旧有的接入方式）。
- 新增录制参数 `crf`（清晰度）和 `tagPositions`（自定义视频布局），具体参见 附录 中参数说明。
- 新增发布参数，支持自定义设置分辨率、帧率、码率。
- 新增文件共享关闭回调接口 `OnFileStreamClosed`。
- 新增桌面共享关闭回调接口 `OnDesktopDisplayClosed`。

V 1.4.8（2021-10-12）

- 增加 StreamFilterHandler 功能，支持自定义编辑流等功能。
- 去除非 Chrome 浏览器的版本判断，使用 alert 强提示。

> **说明**
> 以下 V 1.4.8 版本之前的发布说明内容已不再维护。

V 1.4.7（2021-08-23）

- 废弃 `ice_relay_ip_array` 参数，统一在服务端上配置。
- 共享音视频文件支持播放/暂停。
- 自动发布订阅增加条件选项（即 `auto_publish_subscribe` 参数增加选项 5）。
- 支持自定义推流。
- 增加摄像头切换分辨率接口。
- 优化浏览器录制：
  - 支持设置画布大小。
  - 支持时间戳/文字/图片水印。

V 1.4.6（2021-06-08）

- 修复 Chrome 88 以上版本中共享视频文件在对端无法查看的问题。

V 1.4.5（2021-05-18）

- 增加 540P 分辨率档位。
- 拆分 SDK 内部架构。
- 切流时发出事件，并变更 mediaSource。
- 支持服务端录制实时变更水印。
- 调整服务端录制启动参数（兼容旧参数，详见 附录）。

V 1.4.4（2021-03-04）

- 浏览器录制支持 4 路。
- 浏览器录制流与流之间增加白边分屏。

V 1.4.3（2021-02-24）

- 增加 1280P 分辨率档位。
- 调整各档位码率。
- 支持自定义设置多个中转代理外网 IP。
- bundlePolicy 设置为 max-bundle，多路流共用同一个 ice。

V 1.4.2（2021-01-21）

- 修复自动订阅模式下自定义中转服务器设置问题。

V 1.4.1（2021-01-14）

- 修复 SDK 初始化后首次进入房间心跳异常问题。
- 修复服务端推送消息部分没回 ack 问题。
- 优化自定义区域共享。
- 摄像头接口松动监测，支持自动恢复。
- 支持发布纯音频不传 video 标签，订阅纯视频（纯音频）只传 video（audio）标签。
- 提供预启动摄像头/麦克风功能。

V 1.4.0（2020-12-17）

- 修复自定义推流发布成功后没有回调问题。
- 修复帧率切换失效问题。
- 支持服务端录制每个画面打上发布 tag 水印。
- 修复直播 rtmp 地址设置无效问题。

V 1.3.9 （2020-12-09）

- 修复自定义推流模糊问题。

V 1.3.8 （2020-12-02）

- 修复房间中无发布流情况下启动录制会报错的缺陷。

V 1.3.7 （2020-11-23）

- 音视频通话功能支持 H5：Android Chrome 浏览器（VP8/部分机型支持 H264）/iOS Safari（H264）。

V 1.3.6 （2020-11-06）

- 废弃 ASR/TTS 功能。

V 1.3.5 （2020-11-02）

- 服务端录制支持自定义布局（见主调接口 3、21 中的 mixPosition 参数）。
- 支持同一时间进行多个发布。

V 1.3.4 （2020-09-29）

- 服务端录制支持两种结束类型（除主动调用结束）（见主调接口 3、21 中的 endType 参数）。
- 修复外接麦克风插拔导致没声音问题。

V 1.3.3 （2020-09-02）

- 服务端录制支持静默功能（见主调接口 3、21 中的 silentRecord 参数）。

V 1.3.2 （2020-08-25）

- 修复客户端录制视频高度不一致，导致拼接视频后高度小的视频下方出现黑块的问题。
- 修复服务端录制在全房间没有发布情况下会出现初始化超时问题。

V 1.3.1 （2020-08-14）

- 服务端录制回调接口命名调整（兼容旧接口）。
- 服务端录制增加暂停/恢复功能。

V 1.3.0 （2020-08-12）

- 服务端录制增加参数 record_third_id，用于业务方区分调用批次。启动录制时传入（见主调接口 3、21），在初始化成功、启动成功、启动失败的回调携带返回（见被调接口 33、34、63）。
- 支持并发启动录制。

V 1.2.9 （2020-08-11）

- 优化音视频质量统计。
- 服务端录制放开并发限制。

V 1.2.8 （2020-08-07）

- 浏览器录制：升级接口，丰富配置（原主调接口 16 废弃，新接口为主调接口 40）。

- 服务端录制：增加 tagFilter、tagPostion 配置（见主调接口 3、21）。
- 初始化房间增加三方业务 ID 属性（见主调接口 3）。

V 1.2.7 （2020-07-28）

- 发布添加最大码率参数 publish_bitrate：
  - InitRoomConfig 接口（见主调接口 3）。
  - Publish 接口（见主调接口 9）。
  - ChangeMediaStream 接口（见主调接口 37）。
- 自定义推流支持 PDF 文件。
- 服务端录制支持时间戳水印、图片水印，支持多人录制（见主调接口 3、21）。
- 支持设置自定义设置中转服务器 defaultTurnServer（见主调接口 3）。

V 1.2.6

- 修复 ice 建立连接后清除定时器。

V 1.2.5

- 修复离开房间销毁浏览器端录制信息。

V 1.2.4

- 浏览器录制相关接口调整
  - 主调接口调整
    - StartRecord 删除 targetType 参数，添加 timeSlice 参数 recordType 值调整为 1,2,3（见主调接口 16）。
    - DownloadRecord 删除 returnType 参数（见主调接口 20）。
  - 被调通知接口调整
    - OnClientStartRecordSuccess 删除 targetType参数（见被调接口 67）。
    - OnClientStopRecordSuccess 删除 targetType参数（见被调接口 69）。
    - OnClientDownloadRecordSuccess 删除 returnType, clientRecordBlob 参数（见被调接口 75）。
    - 添加 OnClientRecordBlob 接口（见被调接口 81）。

V 1.2.3

- 添加相关。
  - 添加切流成功通知 OnChangeMediaStreamSuccess（见被调接口 79）。
  - 添加切流失败通知 OnChangeMediaStreamFailed（见被调接口 80）。
  - ChangeProfile 添加参数 local_video_width 和 local_video_height（见主调接口 38）。
- 删除相关。
  - 浏览器录制，停止录制 StopRecord 接口删除 targetType 参数（见主调接口 17）。
  - video_profile_type 参数去掉选项 4（见主调接口 3、9）。
- 内部调整。
  - 创建房间和加入房间添加用户浏览器信息（SDK 内部调整）。
  - 修复浏览器录制结果 blob 为空（SDK 内部优化）。
  - 发布流 tag 更新（SDK 内部调整）。

- 添加 web-sdk 业务接入说明。

V 1.2.2

- 修复浏览器录制多次回调的 bug。

V 1.2.1

- publish_device 新增 2 种发布类型，发布文件和发布自定义页面区域（见主调接口 3、9）。
- 发布和初始化接口调整。
  - 新增 videoSource 参数, 摄像头的deviceId, 用于发布指定摄像头（见主调接口 3、9）。
  - 新增 audioSource 参数,麦克风的deviceId,用于发布指定麦克风（见主调接口 3、9）。
  - 新增 aspectRatioStrongDepend 参数，是否指定横纵比（见主调接口 3、9）。
  - 新增 aspectRatio 参数，横纵比（见主调接口 3、9）。
  - 新增 file 参数，用于发布图片或视频文件（见主调接口 9）。
  - 新增 part_of_screen_id 参数，用于发布指定页面区域（见主调接口 9）。
- 新增自定义切流接口 ChangeMediaStream（见主调接口 37）。
- 新增动态切换（分辨率、摄像头麦克风、横纵比）接口 ChangeProfile（见主调接口 38）。
- 新增音视频设备相关接口。
  - 新增获取音视频设备接口 GetDevices（见主调接口 39）。
  - 获取设备信息成功 OnGetDevicesSuccess（见被调接口 77）。
  - 获取设备信息失败 OnGetDevicesFailed（见被调接口 78）。

V 1.2.0

- 浏览器录制
  - 主调接口
    - StartRecord 开启浏览器录制（见主调接口 16）。
    - StopRecord 停□浏览器录制（见主调接口 17）。
    - PauseRecord 暂停浏览器录制（见主调接口 18）。
    - ResumeRecord 继续浏览器录制（见主调接口 19）。
    - DownloadRecord 下载浏览器录制音视频（见主调接口 20）。
  - 回调接口
    - OnClientStartRecordSuccess 开启浏览器录制成功（见被调接口 67）。
    - OnClientStartRecordFailed 开启浏览器录制失败（见被调接口 68）。
    - OnClientStopRecordSuccess 停止浏览器录制成功（见被调接口 69）。
    - OnClientStopRecordFailed 停止浏览器录制失败（见被调接口 70）。
    - OnClientPauseRecordSuccess 暂停浏览器录制成功（见被调接口 71）。
    - OnClientPauseRecordFailed 暂停浏览器录制失败（见被调接口 72）。
    - OnClientResumeRecordSuccess 继续浏览器录制成功（见被调接口 73）。
    - OnClientResumeRecordFailed 继续浏览器录制失败（见被调接口 74）。
    - OnClientDownloadRecordSuccess 下载浏览器录制文件成功（见被调接口 75）。
    - OnClientDownloadRecordFailed 下载浏览器录制文件失败（见被调接口 76）。

V 1.1.9

- ASR 回调接口添加 sessionId。
  - 开始语音识别成功 OnStartAsrSuccess（见被调接口 49）。
  - 开始语音识别失败 OnStartAsrFailed（见被调接口 50）。
  - 停止语音识别成功 OnStopAsrSuccess（见被调接口 51）。
  - 停止语音识别失败 OnStopAsrFailed（见被调接口 52）。
  - 通知识别结果 OnAsrMessage（见被调接口 53）。
  - 长时间没有返回结果通知 OnAsrMsgIdle（见被调接口 54）。

V 1.1.8

- 去掉 ASR 连接时 uid 带 asr 后缀（SDK 内部更新）。

V 1.1.7

- ASR 相关。
  - 说明：
    - tag 是业务透传参数。
    - status 表示会话是否结束，status=0 表示会话未结束，status=1 表示会话结束后的最终结果。
  - StartAsr 接口 config 添加 tag 业务透传参数（见主调接口 30）。
  - OnStartAsrSuccess 接口增加返回参数 tag（见被调接口 49）。
  - OnStartAsrFailed 接口增加返回参数 tag（见被调接口 50）。
  - OnStopAsrSuccess 接口增加返回参数 tag（见被调接口 51）。
  - OnStopAsrFailed 接口增加返回参数 tag（见被调接口 52）。
  - OnAsrMessage 接口增加两个参数 OnAsrMessage(msg, sid)—>OnAsrMessage(msg, sid,tag,status)（见被调接口 53）。
  - OnAsrMsgIdle 接口增加返回参数 tag（见被调接口 54）。
  - 修复 ASR 并发启动。
- TTS 相关。
  - 开始语音合成 StartTts（见主调接口 35）。
  - 停止语音合成 StopTts（见主调接口 36）。
  - 开启语音合成成功 OnStartTtsSuccess（见被调接口 64）。
  - 开启语音合成失败 OnStartTtsFailed（见被调接口 65）。
  - 语音合成结果 OnTtsResult（见被调接口 66）。
- 录制相关。
  - 录制默认参数调整 （见主调接口 3、21）。
    - 如果 recordTotalStream=0，startTimeout 默认为 10s。
    - 如果 recordTotalStream!=0，startTimeout 默认为 5 分钟。
  - startTimeout 优先级：业务指定优先于默认值。

V 1.1.6

- 更新 ASR 生产环境域名。

V 1.1.5

- Chrome 版本号低于 67，弹窗提示。
- 发布分辨率档位调整：
  - 1: 720p（HD）, 1280x720, 16:9。
  - 2: VGA, 640x480, 4:3。
  - 3: 360p（nHD）, 640x360, 16:9。
  - 4: CIF, 352x288, 4:3。
- 发布时优先按照上述分辨率采集，如浏览器不支持，则按浏览器默认分辨率采集。
- 访问设备失败时，alert 弹窗提示原因。
- 服务端录制调整：
  - 启动超时时间 startTimeout：范围在 3 ~ 10s 之间，默认 5s。
  - 录制启动成功标准 recordTotalStream：成功订阅流，订阅流数默认为 0，之前默认为 2（2 即成功录下双方）。
  - 服务端录制增加初始化成功接口 OnInitRecordSucc （见被调接口 63）。说明：正常情况下，开启录制后会先回调 OnInitRecordSucc，之后再回调 OnStartRecordSucc，两个回调之间的时间间隔视业务设置 recordTotalStream 而定。
- ASR 内部优化修复状态管理漏洞。

V 1.1.4

- ASR 内部优化修复识别不准的问题。

V 1.1.3

- 录制通知回调 SDK 内部优化
- ASR 识别结果：完善说明，根据 procType 来确定格式（见被调接口 53）。

V 1.1.2

- 增加点播通知功能，StartVod 接口参数变更 vodParams（见主调接口 32）。
- 增加录制通知功能，InitRoomConfig 和 StartRemoteRecord 接口中 recordParam 参数添加 2 个属性：startTimeout 和 recordTotalStream（见主调接口 3、21）。
- 增加录制过程中录制中断接口 OnRecordingFailed（见被调接口 62）。
- 录制默认音频采样率 sampleRate 从 24000 修改为 48000（见 InitRoomConfig 和 StartRemoteRecord）。
- 废弃 OnRecordInfo 接口，通过 OnStartRecordSucc 替代（见被调接口 32）。

V 1.1.1

- 修复 ASR 验签失败（SDK 内部改动,接口未改动）。

V 1.1.0

- 修复离开房间未关闭摄像头（SDK 内部改动，接口未改动）。

V 1.0.9

- 增加开始语音识别失败接口 OnStartAsrFailed（见被调接口 50）。
- 语音识别结果接口名从 OnAsrMsg 修改为 OnAsrMessage（见被调接口 53）。
- 开始语音识别接口添加参数引擎类型 procType（见主调接口 30）。
- 服务端录制结束通知接口名从 OnRecorderOver 修改为 OnRecordOver（见被调接口 61）。

- 增加视频清晰度参数 video_profile_type（见主调接口 3、9）。
- 弱网回调接口添加 sid 参数（见被调接口 48）。
- 废弃指令消息相关接口（见主调接口 27 和被调接口 41 ~ 44）。

V 1.0.8

- 增加错误日志上报保存到本地（见主调接口 34）。

V 1.0.7

- 增加语音识别功能（见主调接口 30 ~ 31, 被调接口 49 ~ 54）。
- 增加语音点播功能（见主调接口 32 ~ 33, 被调接口 55 ~ 59）。
- 增加自定义截屏文件格式（见主调接口 15）。
- 增加实时音量可视化功能（见主调接口 3、9、10 的 need_volume_analyser 字段，被调接口 60 的回调接口）。
- 增加服务端录制结束通知回调（见被调接口 61）。
- 音视频关闭通知接口添加 sid 参数（见被调接口 30）
- 邀请加入房间添加 channelType 参数 （见主调接口 6）。

V 1.0.6

- 修改发布弱网频率告警（见主调接口 28 SetPublishWeakBitrateLimit）。
- 添加订阅弱网频率告警（见主调接口 29 SetSubscribeWeakBitrateLimit）。
- 添加退出回调参数，退出类型leaveType（见被调接口 45 OnLeaveRoom）。
- 启动服务端录制添加参数 recordParam（见主调接口 21 StartRemoteRecord）。
- 添加退出类型参数（见被调接口 46 OnParticipantLeaveRoom）。
- 添加服务端录制相关参数 recordParam 和 recordStrongDepend（见主调接口 3 InitRoomConfig）。

V 1.0.5

- 添加获取录制信息接口（见 GetRecordInfo）。
- 添加获取服务端录制结果成功（见 OnRecordInfoSucc）。
- 添加获取服务端录制结果失败（见 OnRecordInfoFailed）。
- 添加退出类型参数 OnParticipantLeaveRoom（见被调接口 46）。
- 添加服务端是否录制 defaultRecord参数（见主调接口 3 InitRoomConfig）。

V 1.0.4

- 添加发布共享桌面功能。
- 添加弱网回调接口（见被调接口 46）。
- 添加设置弱网码率告警下限（见主调接口 27）。
- 添加参数 publish_device 标识发布类型（见主调接口 3 和主调接口 9）。

V 1.0.3

- 支持签名在页面重写（被调接口 3）。
- 截屏如果不指定宽高，则默认为视频窗口的宽高。
- TakePicture() 接口参数宽高设置默认值（主调接口 15）。
- 服务端录制，不在录制中时停止录制，直接返回（主调接口 22）。
- StartRemoteRecord 接口添加可选参数服务端录制路径（主调接口 21）。

- publish_video_id 由必填更正为根据 media_type 选填（主调接口 3 和主调接口 9）。
- StartRemoteRecord 接口添加可选参数 filePath 服务端录制路径（主调接口 21）。
- OnSubscribeFailed(sid,err_code, err_msg) 接口去掉 feed 参数（被调接口 23）。
- 如果发布的是纯音频，则 sid 获取方式为 publish_streamId_id 的 name 值，具体参考 meeting.html。
- InitRoomConfig 接口添加服务端录制路径的可选参数 filePath，通过 OnRecordInfo 接口回调 recordId（见主调接口 3 和被调接口 32）。

V 1.0.2

- 添加录制功能：主调接口 16 ~ 22 和被调接口 31 ~ 35。

6.3.2. 主调接口

本文介绍音视频通话 API 在接入 Web 端时涉及到的主调接口。

Connect

建立通话连接。`config_param` 是一个结构体，里面包含如下字段。

- **参数：**

请求参数	类型	是否必填	说明
uid	int	必填	用户唯一标识，由客户端透传而来。 说明 目前 uid 只支持英文字母、数字、下划线的组合，且长度不超过 128 个字符。
room_server_url	String	必填	房间服务器地址，公有云中金区环境与非金区环境的房间服务器地址不同，您可以按您当前的环境选择房间服务器的地址。 ○ 非金区：wss://mrtc.mpaas.cn-hangzhou.aliyuncs.com/ws ○ 金区：wss://room.mrtc-fin.cn-shanghai.aliyuncs.com/ws
biz_name	String	必填	业务标识
sub_biz	String	必填	业务标识，mPaaS 下对应的 appId，可以从 mPaaS 控制台 中获取。
sign	String	必填	签名供验证

- **回调接口异步返回：**
  - **连接成功：** `OnConnectOK()`

- **连接失败：** `OnConnectFailed(err_code, err_msg)`

回调参数	类型	说明
err_code	int	错误码： ▪ -101：初始化超时 ▪ -102：初始化失败
err_msg	String	状态信息

Disconnect

断开连接。调用 `Disconnect()` 接口，如果正处于通话阶段，会触发关闭发布和订阅的流，并离开房间。即 **关闭房间所有流** > **离开房间** > **断开信令连接**。

- **参数：** 无。
- **回调接口异步返回：** 连接关闭 `OnConnectClose()`

InitRoomConfig

初始化房间。

- **参数：**

请求参数	类型	说明
config_param	{}	config_param 是一个结构体，里面包含的字段见下表。

- **config_param 参数：**

请求参数	类型	是否必填	说明
auto_publish_subscribe	int	必填	发布订阅模式： ▪ 1：自动订阅。 ▪ 2：自动发布。 ▪ 3：自动订阅+自动发布。 ▪ 4：由业务自己操作。 ▪ 5：自动订阅+当房间中有其他人时自动发布。
engine	int	选填	引擎类型： ▪ 0：ALIPAY，默认值 ▪ 1：ALIYUN ▪ 2：P2P

third_id	String	-	业务关联 ID，默认为空。
need_volume_analyser	boolean	-	默认为 false，是否需要实时音量值。
liveUrl	String	选填	直播地址，使用直播模式时必填。
publish_type	int	选填	推送模式，使用直播模式时必填。 ▪ 2：RTC 模式 ▪ 3：直播模式
defaultRecord	boolean	选填	服务端是否默认录制，默认为 false。
recordStrongDepend	boolean	-	是否强依赖录制，默认为 true。若为 true，同时录制启动未成功时，音视频通话也无法连接成功。
record_third_id	String	-	录制三方的 ID，用于业务方去区分调用批次，在启动成功、失败的回调中返回，非必传。
recordParam	JSON	选填	具体参数见附录 recordParam。
filePath	String	选填	服务端录制路径，若指定 filePath，则需指定 defaultRecord 为 true。
enableAudio	boolean	非必填	enableVideo 也同时未填时，都默认为 true。
enableVideo	boolean	非必填	enableAudio 也同时未填时，都默认为 true。
enableDataChannel	boolean	选填	是否使用 datachannel，必须在 engine 的值为 2 时才可使用，默认为 false。

- 若 auto_publish_subscribe=1，需要以下参数。

请求参数	类型	说明
initSubscribe	Array	订阅参数
subscribe_video_id	-	用于展示视频流
subscribe_audio_id	-	用于展示订阅音频流
subscribe_streamId_id	-	label 标签，用于展示 streamId。
feedId_id	-	label 标签，用于展示 feedId。

代码示例：

```
// initSubscribe: Array, 订阅参数。
initSubscribe = [
            {
                subscribe_video_id: "video1",
                subscribe_audio_id: "audio1",
                subscribe_streamId_id: "streamId",GetRecordInfo
                feedId_id: "feedId"
            },{},{}];
```

- 若 auto_publish_subscribe 值为 2 时，需要配置以下参数：

请求参数	类型	是否必填	说明
media_type	int	必填	音视频类型： ▪ 1：音视频 ▪ 2：纯音频 ▪ 3：纯视频
publish_bitrate	int	选填	发布最大码率，默认为 600Kbps。
publish_device	int	必填	发布类型： ▪ 1：摄像头 ▪ 2：共享桌面 ▪ 3：发布文件（图片和视频） ▪ 4：发布自定义页面区域 ▪ 5：发布自定义推流

- 若 auto_publish_subscribe 值为 3 或 5 时，需要配置以下参数：

请求参数	类型	是否必填	说明
media_type	int	必填	音视频类型： ▪ 1：音视频 ▪ 2：纯音频 ▪ 3：纯视频
publish_bitrate	int	选填	发布最大码率，默认 600Kbps。
publish_device	int	必填	发布类型： ▪ 1：摄像头 ▪ 2：共享桌面 ▪ 3：发布文件（图片和视频） ▪ 4：发布自定义页面区域 ▪ 5：发布自定义推流
initSubscribe	Array	-	订阅初始化需要以下参数： ▪ subscribe_video_id：用于展示视频流。 ▪ subscribe_audio_id：用于展示订阅音频流。 ▪ feedId_id：label 标签，用于展示 feedId。 ▪ subscribe_streamId_id：label 标签，用于展示 streamId。

代码示例：

```
// initSubscribe: Array, 订阅参数
initSubscribe = [
            {
                subscribe_video_id: "video1",
                subscribe_audio_id: "audio1",
                subscribe_streamId_id: "subscribe_streamId1",
                feedId_id: "feedId1"
            },{},{}];
```

- 当 publish_device 值为 1 时，需要配置以下参数：

请求参数	类型	是否必填	说明

videoSource	String	选填	摄像头的设备 ID，用于发布指定摄像头。
audioSource	String	选填	麦克风的设备 ID，用于发布指定麦克风。
aspectRatioStrongDepend	boolean	-	默认为 false，是否强制指定视频横纵比。
aspectRatio	String	选填	aspectRatioStrongDepend 值为 true 时必填。 视频横纵比： ▪ 1 表示 4:3 ▪ 2 表示 16:9
degradationType	int	选填	流畅度优先或清晰度优先： ▪ 1：流畅度优先（默认） ▪ 2：清晰度优先
video_profile_type	int	选填	本地视频清晰度档位： ▪ 1：720P，1280x720，16:9 ▪ 2：VGA，640x480，4:3 ▪ 3：360P，640x360，16:9 ▪ 4：720P（竖），720x1280，9:16 ▪ 5：360P（竖），360x640，9:16 ▪ 6：1080P，1920x1080，16:9 ▪ 7：1080P（竖），1080x1920，9:16 ▪ 8：90P（竖），90x160，9:16 ▪ 9：540P，960x540，16:9 ▪ 10：540P（竖），540x960，9:16 ▪ 100：自定义
video_profile_diy	Object	选填	自定义视频宽高、帧率、码率。 当 video_profile_type 为 100 时，才需带上参数 video_profile_diy，传入视频宽高、帧率和码率。格式为：`video_profile_diy: {width,height,frameRate,bitrate}`。

enableDesktopAudio	boolean	选填	投屏时是否将声音一并投出（仅限共享tab），为 true 时表示投声音。 当 publishDevice 的值为 2 时才有效。
initPublish	Array	必填	发布初始化参数
publish_video_id	-	选填	用于展示发布的视频流，根据media_type 选填。
publish_streamId_id	-	-	label 标签，用于展示 streamId。
publish_tag	-	-	业务自定义

代码示例如下：

```
// initPublish: Array，必填，发布参数。
initPublish = [
            {publish_video_id: "publish_video1", publish_streamId_id:
"publish_streamId1", publish_tag: "tag1"},
            {publish_video_id: "publish_video2", publish_streamId_id:
"publish_streamId2", publish_tag: "tag2"},
            {publish_video_id: "publish_video3", publish_streamId_id:
"publish_streamId3", publish_tag: "tag3"}
        ];
```

- **回调接口异步返回：**
  - **初始化房间成功：** `OnInitRoomConfigOK()`
  - **初始化房间失败：** `OnInitRoomConfigFail(err_code, err_msg)`

回调参数	类型	说明
err_code	int	初始化房间失败返回： ▪ -201：房间状态错误 ▪ -202：参数错误 ▪ -204：E2EE_ikm 为空 ▪ -205：自定义分辨率、帧率、码率参数未提供。
err_msg	String	状态信息

CreateRoom

创建房间。

- **参数：**

请求参数	类型	说明
sign	String	验证签名

- **回调接口异步返回：**
  - **创建房间成功：** `OnCreateRoomSucc(room_id, rtoken)`

回调参数	类型	说明
room_id	String	房间号
rtoken	String	房间 token，相当于房间密码。

  - **创建房间失败：** `OnCreateRoomFailed(err_code, err_msg)`

回调参数	类型	说明
err_code	int	创建房间失败返回： ▪ -301：房间未初始化 ▪ -302：服务器返回错误 ▪ -303：创建房间超时 ▪ -304：媒体初始化超时
err_msg	String	状态信息

JoinRoom

加入房间。

- **参数：**

请求参数	类型	说明
room_id	String	房间 ID
rtoken	String	房间 token，相当于房间密码。
sign	String	验证签名

- **回调接口异步返回：**
  - **加入房间成功：** `OnJoinRoomSucc()`

- **加入房间失败：** `OnJoinRoomFailed(err_code, err_msg)`

回调参数	类型	说明
err_code	int	加入房间失败返回： ▪ -401：房间未初始化 ▪ -402：参数错误 ▪ -403：服务器返回错误 ▪ -404：加入房间超时
err_msg	String	状态信息

Invite

邀请加入房间。

- **参数：**

请求参数	类型	说明
invitee	String	被邀请方 UID
channelType	int	推送渠道： ◦ 0：WEB ◦ 1：ALIPAY
extra	String	业务透传字段

- **回调接口异步返回：**
  - **发送邀请加入房间成功：** `OnInviteOK()`
  - **发送邀请加入房间失败：** `OnInviteFail(err_code, err_msg)`

回调参数	类型	说明
err_code	int	发送邀请加入房间失败返回： ▪ -501：房间状态错误 ▪ -502：服务器返回错误 ▪ -503：参数错误
err_msg	String	状态信息

ReplyInvite

回复邀请加入房间。

- **参数：**

请求参数	类型	说明
roomId	String	房间 ID
inviter	String	邀请方 UID
reply	int	邀请回复： ◦ 0：接听 ◦ 2：拒绝

- **回调接口异步返回：**
  - **向邀请方发送答复加入房间成功：** `OnReplyInviteOK()`
  - **向邀请方发送答复加入房间失败：** `OnReplyInviteFail(err_code, err_msg)`

回调参数	类型	说明
err_code	int	向邀请方发送答复加入房间失败返回： ▪ -601：房间状态错误 ▪ -602：服务器返回错误
err_msg	String	状态信息

LeaveRoom

退出房间。

- **参数：**无。
- **回调接口异步返回：** `OnLeaveRoom(leaveType)`

回调参数	类型	说明
leaveType	int	退出房间类型： ◦ 1：正常退出 ◦ 2：异常退出 ◦ 3：被踢出房间

Publish

手动发布媒体流。

- **参数：**

请求参数	类型	是否必填	说明
media_type	int	必填	音视频类型： ◦ 1：音视频 ◦ 2：纯音频 ◦ 3：纯视频
publish_bitrate	int	选填	发布最大码率，默认为 600Kbps。
publish_device	int	必填	发布类型： ◦ 1：摄像头 ◦ 2：共享桌面 ◦ 3：发布文件（图片和视频） ◦ 4：发布自定义页面区域 ◦ 5：发布自定义推流
video_profile_type	int	选填	本地视频清晰度档位： ◦ 1：720P，1280x720，16:9 ◦ 2：VGA，640x480，4:3 ◦ 3：360P，640x360，16:9 ◦ 4：720P（竖），720x1280，9:16 ◦ 5：360P（竖），360x640，9:16 ◦ 6：1080P，1920x1080，16:9 ◦ 7：1080P（竖），1080x1920，9:16 ◦ 8：90P（竖），90x160，9:16 ◦ 9：540P，960x540，16:9 ◦ 10：540P（竖），540x960，9:16 ◦ 100：自定义
video_profile_diy	Object	选填	自定义视频宽高、帧率、码率。 当 video_profile_type 为 100 时，才需带上参数 video_profile_diy，传入视频宽高、帧率和码率。格式为：`video_profile_diy: {width,height,frameRate,bitrate}`。
publish_video_id	-	选填	根据 media_type 选填。用于展示发布的视频流
publish_streamId_id	-	-	用于展示流 ID

publish_tag	String	-	业务指定 tag
need_volume_analyser	boolean	-	是否需要实时音量值，默认为 false。
enableDesktopAudio	boolean	选填	投屏时是否将声音一并投出（仅限共享 tab），为 true 时表示投声音。 当 publishDevice 的值为 2 时才有效。
file	file	选填	为图片或视频文件，格式示例： `let file = document.getElementById('file').files[0];` publish_device 值为 3 时必填。
part_of_screen_id	String	选填	需要共享区域的 div 的 ID 值，格式示例：`<div id="part_of_screen_id"></div>` publish_device 值为 4 时必填。
stream	String	选填	媒体流，publish_device 值为 5 时必填。

- 当 publish_device 值为 1 时，需要配置以下参数：

请求参数	类型	是否必填	说明
videoSource	String	选填	摄像头的设备 ID，用于发布指定摄像头。
audioSource	String	选填	麦克风的设备 ID，用于发布指定麦克风。
aspectRatioStrongDepend	boolean	-	默认为 false，是否强制指定视频横纵比。
aspectRatio	String	选填	视频横纵比： ▪ 1 表示 4:3 ▪ 2 表示 16:9 重要 aspectRatioStrongDepend 的值为 true 时必填。
degradationType	int	选填	流畅度优先或清晰度优先： ▪ 1：流畅度优先（默认） ▪ 2：清晰度优先
publish_bitrate	int	选填	发布媒体流的最大码率，默认为 600Kbps。
enableAudio	boolean	非必填	enableVideo 也同时未填时，都默认为 true。
enableVideo	boolean	非必填	enableAudio 也同时未填时，都默认为 true。

- **回调接口异步返回：**
  - **发布媒体流成功：** `OnPublishSucc(sid)`

回调参数	类型	说明
sid	int	发布媒体流的 ID

- **发布媒体流失败：** `OnPublishFailed(sid,err_code, err_msg)`

回调参数	类型	说明
sid	int	流 ID
err_code	int	发布媒体流失败返回： ▪ -1071：服务器返回错误 ▪ -1072：未设置视频标签 ▪ -1073：超出发布限额 ▪ -1074：发布建立媒体链路连接超时
err_msg	String	状态信息

Subscribe

订阅某路媒体流。 `config_param` **是一个结构体，里面包含如下字段：**

- **参数：**

字段名	类型	说明
subscribe_video_id	String	用于展示订阅的视频流
subscribe_audio_id	String	用于展示订阅的音频流
subscribe_streamId_id	String	用于展示订阅的流
feedId_id	String	用于展示 feedId
feedId	String	订阅的流 ID
need_volume_analyser	boolean	默认为 false，是否需要实时音量值。

- **回调接口异步返回：**

- **订阅媒体流成功：** `OnSubscribeSucc(feedId,sid)`

回调参数	类型	说明
sid	String	流 ID
feedId	String	流信息

feedId 流信息代码示例如下：

```
{
      "feedId": ["xxx","xxx"],
      "strategy": 1
}
// feedId: list, 订阅的流 ID
// strategy: int, 策略
```

- **订阅媒体流失败：** `OnSubscribeFailed(sid,err_code, err_msg)`

回调参数	类型	说明
sid	String	流 ID
err_code	int	订阅媒体流失败返回： ▪ -1081：服务器返回错误 ▪ -1082：视频窗口的 subscribe_video_id 已使用 ▪ -1083：超出订阅限额 ▪ -1084：订阅建立媒体链路失败 ▪ -1085：订阅的窗口 subscribe_video_id 为空
err_msg	String	状态信息

SetLocalAudioEnable

开启或关闭本地麦克风。

参数：

请求参数	类型	说明
enabled	int	开启或关闭本地麦克风： • 1：开启 • 0：关闭

sid	int	流 ID

SetLocalVideoEnable

开启或关闭本地摄像头。

参数：

请求参数	类型	说明
enabled	int	开启或关闭本地摄像头： • 1：开启 • 0：关闭
sid	int	流 ID

SetLocalCodecType

设置视频编码格式。

参数：

请求参数	类型	说明
type	String	视频编码格式： • H264 • VP8 • VP9 • AV1

TakePicture

截图。

参数：

请求参数	类型	说明
type	int	截图类型： • 0：截取本地端 • 1：截取远端图像
width	int	图像宽度
height	int	图像高度

sid	int	流 ID
picture_type	int	输出图片格式： • 1：PNG • 2：JPEG 或 JPG

StartBrowserRecord

开启浏览器录制（近端录制），指基于浏览器的 native 录制能力进行录制。目前只支持录制两条流，不支持录制点播机器人。

- **参数：**

请求参数	类型	说明
clientRecordId	String	浏览器录制 ID
stream_list	Array	流 ID（sid）

录制配置参数 record_config 使用 JSON 数据格式，相关说明如下表所示。

请求参数	类型	说明
media_type	int	录制选项： ◦ 1：音视频 ◦ 2：纯音频 ◦ 3：纯视频
time_slice	int	录制 blob 通知间隔时间，默认为 30，单位为分钟。
video_profile	JSON	-
videoCodec	String	支持 H264、VP8、VP9 视频编码格式。
frameRate	int	帧率
record_resolution	String	支持如下视频录制的尺寸大小： ◦ 1920 x 1080 ◦ 1280 x 720 ◦ 640 x 360

overlaps	Array	支持时间戳水印、文字水印、图片水印，视频相关水印说明如下： ◦ 时间戳水印 ▪ enable：是否增加水印。 ▪ type：1 种水印类型。 ▪ id：水印 ID。 ▪ xPosition：水印 X 轴起始位置。 ▪ yPosition：水印 Y 轴起始位置。 ◦ 文字水印 ▪ enable：是否增加水印。 ▪ type：2 种水印类型。 ▪ id：水印 ID。 ▪ text：水印文字。 ▪ fontSize：水印文字大小。 ▪ xPosition：水印 X 轴起始位置。 ▪ yPosition：水印 Y 轴起始位置。 ◦ 图片水印 ▪ enable：是否增加水印。 ▪ type: 3 种水印类型。 ▪ id：水印 ID。 ▪ img：需要添加的水印图片元素 new image()。 ▪ xPosition：水印在 X 轴的起始位置。 ▪ yPosition：水印在 Y 轴的起始位置。

- **回调接口异步返回：**
  - **开启浏览器录制成功：** `OnClientStartRecordSuccess(clientRecordId)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID

- **开启浏览器录制失败：** `OnClientStartRecordFailed(clientRecordId, code, msg)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
code	int	开启浏览器录制失败返回： ▪ -20301：房间状态不对。 ▪ -20302：已经在录制中。 ▪ -20303：clientRecordId 已经使用。 ▪ -20304：未发布或未订阅。 ▪ -20305：media_type 参数错误。 ▪ -20306：stream_list 长度错误。 ▪ -20307：没有要求录制的视频流或者音频流。 ▪ -20308：record_resolution 错误。
msg	String	状态信息

StopRecord

停止浏览器录制，与 StartBrowserRecord 接口配合使用。离开房间之前需要调用此接口，否则无法获取 blob 数据。

- **参数：**

请求参数	类型	说明
clientRecordId	String	浏览器录制 ID

- **回调接口异步返回：**
  - **停止浏览器录制成功：** `OnClientStopRecordSuccess(videoURL, clientRecordBlob, clientRecordId)`

回调参数	类型	说明
videoURL	String	视频文件的 URL
clientRecordBlob	blob	录制文件
clientRecordId	String	浏览器录制 ID

- 停止浏览器录制失败：`OnClientStopRecordFailed(clientRecordId, code, msg)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
code	int	错误码，-20331 为房间状态错误码
msg	String	状态信息

PauseRecord

暂停浏览器录制，与 ResumeRecord 接口配合使用。

- **参数：**

请求参数	类型	说明
clientRecordId	String	浏览器录制 ID

- **回调接口异步返回：**
  - 暂停浏览器录制成功：`OnClientPauseRecordSuccess(clientRecordId)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID

  - 暂停浏览器录制失败：`OnClientPauseRecordFailed(clientRecordId, code, msg)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
code	int	暂停浏览器录制失败返回： ▪ -20371：房间状态错误 ▪ -20372：未开启浏览器录制
msg	String	状态信息

ResumeRecord

继续浏览器录制，与 PauseRecord 接口配合使用。

- **参数：**

请求参数	类型	说明
clientRecordId	String	浏览器录制 ID

- **回调接口异步返回：**
  - **继续浏览器录制成功：** `OnClientResumeRecordSuccess(clientRecordId)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID

  - **继续浏览器录制失败：** `OnClientResumeRecordFailed(clientRecordId, code, msg)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
code	int	继续浏览器录制失败返回： ▪ -20391：房间状态错误 ▪ -20392：未开启浏览器录制
msg	String	状态信息

DownloadRecord

下载浏览器录制音视频，需要在离开房间之前下载。

- **参数：**

请求参数	类型	说明
clientRecordId	String	浏览器录制 ID
fileName	String	指定文件名

- **回调接口异步返回：**
  - **下载浏览器录制文件成功：** `OnClientDownloadRecordSuccess(clientRecordId)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID

- **下载浏览器录制文件失败：** `OnClientDownloadRecordFailed(clientRecordId, code, msg)`

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
code	int	下载浏览器录制文件失败返回： ▪ -20351：房间状态错误 ▪ -20352：未开启浏览器录制 ▪ -20353：未停止浏览器录制
msg	String	状态信息

StartRemoteRecord

开始服务端录制（远端录制，即在媒体服务器上进行录制），与 StopRemoteRecord 接口配合使用。

- **参数：**

请求参数	类型	是否选填	说明
file_path	String	选填	服务端录制路径
record_param	JSON	选填	录制参数，具体参数见 附录 recordParam。
record_third_id	String	-	录制三方 ID，用于业务方区分调用批次，在启动成功或失败的回调中返回，非必传。

- **回调接口异步返回：**
  - **初始化服务端录制成功：** `OnInitRemoteRecordSucc(record_id, record_third_id)`

回调参数	类型	说明
recordId	String	录制 ID
record_third_id	String	启动录制时传入的 ID，区分调用批次

- **启动服务端录制成功：** `OnStartRemoteRecordSucc(record_id, record_third_id)` ，晚于 `OnInitRemoteRecordSucc`

回调参数	类型	说明
record_id	String	服务端录制 ID
record_third_id	String	启动录制时传入的 ID，区分调用批次

- **启动服务端录制失败：** `OnStartRemoteRecordFailed(record_id, err_code, err_msg, record_third_id)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	int	启动服务端录制失败返回： ▪ -10311：服务端返回开启录制失败 ▪ -10312：开启录制超时 ▪ -10313：推送开启录制失败，eventCode 值为 1 ▪ -10314：推送开启录制失败，eventCode 值为 2
err_msg	String	状态信息
record_third_id	String	启动录制时传入的 ID，区分调用批次

StopRemoteRecord

结束服务端录制，与 StartRemoteRecord 接口配合使用。

- **参数：**

请求参数	类型	说明
record_id	String	服务端录制 ID

- **回调接口异步返回：**
  - **停止服务端录制成功：** `OnStopRecordSucc(recordId)`

回调参数	类型	说明
record_id	String	服务端录制 ID

- **停止服务端录制失败：** `OnStopRecordFailed(recordId,err_code, err_msg)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	String	10331（错误码）：服务端返回停止失败
err_msg	String	状态信息

GetRecordInfo

获取服务端录制结果。

- **参数：**

请求参数	类型	说明
record_id	String	服务端录制 ID
room_id	String	房间号
media_type	int	文件内容类型： ◦ 0：音视频，为默认值 ◦ 1：纯音频 ◦ 2：纯视频
sign	String	业务主动提供签名验证。

- **回调接口异步返回：**

- **获取服务端录制结果成功：** `OnRemoteRecordInfoSucc(recordInfo)`

> **说明**
> status 值为 2 时，才会有 fileType 和 filePath。

回调参数	类型	说明

recordInfo	JSON	获取服务端录制结果返回
		▪ recordId：服务端录制 ID
		▪ status：录制状态
		  ▪ 0：录制中
		  ▪ 1：上传中
		  ▪ 2：录制成功
		  ▪ 3：录制失败
		▪ fileType：文件类型
		  ▪ 1：本地文件
		  ▪ 2：OSS
		  ▪ 3：AFTS
		▪ filePath：文件路径

- **获取服务端录制结果失败：** `OnRemoteRecordInfoFailed(record_id, err_code, err_msg)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	int	获取服务端录制结果失败返回： ▪ -10351：连接状态错误 ▪ -10352：服务端返回错误 ▪ -10353：请求参数错误
err_msg	String	状态信息

UnPublish

取消发布媒体流。

- **参数：**

请求参数	类型	说明

sid	int	发布媒体流的 ID

- **回调接口异步返回：**

  取消发布媒体流成功： `OnUnPublishSucc(sid)`

回调参数	类型	说明
sid	String	流 ID

UnSubscribe

取消订阅某路媒体流。

- **参数：**

请求参数	类型	说明
sid	int	订阅某路媒体流的 ID

- **回调接口异步返回：**

  取消订阅某路媒体流成功： `OnUnSubscribeSucc(sid)`

回调参数	类型	说明
sid	String	流 ID

SendTextMsg

发送文本消息。

- **参数：**

请求参数	类型	说明
msg	String	消息文本
participants	Array	发送目标 ID，为空则发给与会其他所有人。

- **回调接口异步返回：**

- **发送文本消息成功：** `OnSendTextMsgSucc(msgId)`

回调参数	类型	说明
msgId	int	消息 ID

- **发送文本消息失败：** `OnSendTextMsgFailed(msgId, code, msg)`

回调参数	类型	说明
msgId	int	消息 ID
code	int	10291（错误码）：服务端返回错误
msg	String	消息文本

SetPublishWeakBitrateLimit

设置发布弱网码率告警下限，在 初始化（InitRoomConfig）时调用此接口。

参数：

请求参数	类型	说明
weak_bitrate	int	发布弱网码率告警下限： • 音频 + 视频：默认为 200 • 单独的音频：默认为 20-25

SetSubscribeWeakBitrateLimit

设置订阅弱网码率告警下限，在 初始化（InitRoomConfig）时调用此接口。

参数：

请求参数	类型	说明
weak_bitrate	int	订阅弱网码率告警下限： • 音频 + 视频：默认为 200 • 单独的音频：默认为 20-25

StartVod()

开始语音点播，仅主站支持。

- **参数：**

请求参数	类型	说明
vodParams	JSON	◦ vodFile：点播的文件名。 ◦ vodStartTimeout：开启点播超时的时间，单位为秒，默认为 5 秒。

- **回调接口异步返回：**
  - **开始语音点播成功：** `OnStartVodSuccess(file, vod_id)`

回调参数	类型	说明
file	String	点播文件
vod_id	String	点播 ID

  - **开始语音点播失败：** `OnStartVodFail(file, vod_id, err_code, err_msg)`

回调参数	类型	说明
file	String	点播文件
vod_id	String	点播 ID
err_code	int	开始语音点播失败返回： ▪ -10371：服务端返回错误 ▪ -10372：房间状态错误 ▪ -10373：服务端通知开启失败 ▪ -10374：点播参数有误 ▪ -10375：开启点播超时
err_msg	String	状态信息

StopVod()

停止语音点播，仅主站支持。

- **参数：**

请求参数	类型	说明
vod_id	String	点播 ID

- **回调接口异步返回：**
  - **停止语音点播成功：** `OnStopVodSuccess(vod_id)`

回调参数	类型	说明
vod_id	String	点播 ID

  - **停止语音点播失败：** `OnStopVodFail(vod_id, err_code, err_msg)`

回调参数	类型	说明
vod_id	String	点播 ID
err_code	int	停止语音点播失败返回： ▪ -10391：服务端返回失败 ▪ -10392：房间状态错误 ▪ -10393：点播 ID 有误
err_msg	String	错误消息

DownloadLog()

日志上报，调用此接口可下载控制台日志到 PC 本机（默认最长 10000 行日志）。

- **参数：**无。

ChangeMediaStream()

自定义切流。录制配置参数 record_config 使用 JSON 数据类型，包含参数如下表所示。

- **参数：**

请求参数	类型	是否必填	说明
publish_device	-	必填	发布类型： ◦ 1：摄像头 ◦ 2：共享桌面 ◦ 3：发布图片和视频文件 ◦ 4：发布自定义页面区域 ◦ 5：发布自定义推流
publish_bitrate	int	选填	发布最大码率，默认为 600Kbps。
media_type	-	必填	音视频类型

sid	-	必填	已发布的流 ID
video_profile_type	-	选填	分辨率，publish_device 值为 1 时，必填。
videoSource	String	选填	摄像头的设备 ID，用于切换指定摄像头。 publish_device 值为 1 时，必填。
audioSource	String	选填	麦克风的设备 ID，用于切换指定麦克风。 publish_device 值为 1 时，必填。
aspectRatioStrongDepend	boolean	选填	默认为 false，是否强制指定横纵比。 publish_device 值为 1 时，必填。
file	-	选填	图片或视频文件。 publish_device 值为 3 时，必填。
part_of_screen_id	-	选填	需要发布的自定义区域。 publish_device 值为 4 时，必填。
stream	-	选填	需要发布自定义推流。 publish_device 值为 5 时，必填。
video_profile_type	int	选填	本地视频清晰度档位： ∘ 1：720P，1280x720，16:9 ∘ 2：VGA，640x480, 4:3 ∘ 3：360P，640x360，16:9 ∘ 4：720P（竖），720x1280，9:16 ∘ 5：360P（竖），360x640, 9:16 ∘ 6：1080P，1920x1080，16:9 ∘ 7：1080P（竖），1080x1920，9:16 ∘ 8：90P（竖）90x160，9:16 ∘ 9：540P，960x540，16:9 ∘ 10：540P（竖），540x960，9:16 ∘ 100：自定义
video_profile_diy	Object	选填	自定义视频宽高、帧率、码率。 当 video_profile_type 为 100 时，才需带上参数 video_profile_diy，里面带入视频宽高，帧率和码率。格式：`video_profile_diy: {width,height,frameRate,bitrate}`。

enableAudio	boolean	选填	enableVideo 也同时未填时，都默认为 true。
enableVideo	boolean	选填	enableAudio 也同时未填时，都默认为 true。

- **回调接口异步返回：**
  - **切流成功通知：** `OnChangeMediaStreamSuccess(sid)`

回调参数	类型	说明
sid	int	流 ID

  - **切流失败通知：** `OnChangeMediaStreamFailed(sid,code, msg)`

回调参数	类型	说明
sid	int	流 ID
code	int	切流失败通知返回： ▪ -1101：当前 sid 未发布 ▪ -1102：切流参数错误 ▪ -1104：获取音视频失败 ▪ -1105：共享屏幕流为空 ▪ -1106：获取视频失败 ▪ -1107：获取文件流失败 ▪ -1108：获取音频失败 ▪ -1109：浏览器不支持音视频 ▪ -1110：未提供自定义流 ▪ -1111：未提供自定义分辨率等参数
msg	String	状态信息

ChangeProfile()

动态切换（分辨率、摄像头麦克风、横纵比）。录制配置参数 record_config 使用 JSON 数据格式，如下表所示。

- **参数：**

请求参数	类型	是否必填	说明
sid	-	必填	已发布的流 ID

video_profile_type	-	选填	视频的分辨率
local_video_width	-	选填	视频的宽度
local_video_height	-	选填	视频的高度
videoSource	String	选填	摄像头的设备 ID，用于切换指定摄像头。
audioSource	String	选填	麦克风的设备 ID，用于切换指定麦克风。
aspectRatioStrongDepend	boolean	-	默认为 false，是否强制指定视频横纵比。
aspectRatio	String	选填	视频横纵比： ∘ 1 表示 4:3 ∘ 2 表示 16:9 aspectRatioStrongDepend 值为 true 时，必填。
video_profile_type	int	选填	本地视频清晰度的档位： ∘ 1：720p，1280x720，16:9 ∘ 2：VGA，640x480，4:3 ∘ 3：360P，640x360，16:9 ∘ 4：720P（竖），720x1280，9:16 ∘ 5：360P（竖），360x640，9:16 ∘ 6：1080P，1920x1080，16:9 ∘ 7：1080P（竖），1080x1920，9:16 ∘ 8：90P（竖），90x160，9:16 ∘ 9：540P，960x540，16:9 ∘ 10：540P（竖），540x960，9:16 ∘ 100：自定义
video_profile_diy	-	选填	自定义视频宽高，帧率，码率。 当 video_profile_type 为 100 时，才需带上参数 video_profile_diy，将视频宽高，帧率和码率带入其中。格式为： `video_profile_diy:{width,height,frameRate,bitrate}`。

- **回调接口异步返回：**
  - **切流成功通知：** `OnChangeMediaStreamSuccess(sid)`

回调参数	类型	说明
sid	int	流 ID

  - **切流失败通知：** `OnChangeMediaStreamFailed(sid,code, msg)`

回调参数	类型	说明
sid	int	流 ID
code	int	切流失败通知返回： ▪ -1101：当前 sid 未发布 ▪ -1102：切流参数错误 ▪ -1104：获取音视频失败 ▪ -1105：共享屏幕流为空 ▪ -1106：获取视频失败 ▪ -1107：获取文件流失败 ▪ -1108：获取音频失败 ▪ -1109：浏览器不支持音视频 ▪ -1110：未提供自定义流 ▪ -1111：未提供自定义分辨率等参数
msg	String	状态信息

GetDevices()

获取设备信息。

- **参数：**无。
- **回调接口异步返回：**
  - **获取设备信息成功：** `OnGetDevicesSuccess(devicesInfo)`

回调参数	类型	说明
devicesInfo	JSON	设备信息

- 获取设备信息失败：`OnGetDevicesFailed(code, msg)`

回调参数	类型	说明
code	int	获取设备信息失败返回： ▪ -30101：浏览器不支持音视频 ▪ -30102：获取媒体错误
msg	String	状态信息

PauseRemoteRecord

暂停服务端录制，与 ResumeRemoteRecord 接口配合使用。

- **参数：**

请求参数	类型	说明
record_id	String	服务端录制 ID

- **回调接口异步返回：**
  - 暂停服务端录制成功：`OnPauseRemoteRecordSucc(record_id)`

回调参数	类型	说明
record_id	String	服务端录制 ID

  - 暂停服务端录制失败：`OnPauseRemoteRecordFailed(record_id, err_code, err_msg)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	int	暂停服务端录制失败返回： ▪ -10341：房间状态错误 ▪ -10342：录制状态错误 ▪ -10343：服务端错误
err_msg	String	状态信息

ResumeRemoteRecord

恢复服务端录制，与 PauseRemoteRecord 接口配合使用。

- **参数：**

请求参数	类型	说明
record_id	String	服务端录制 ID

- **回调接口异步返回：**
  - **恢复服务端录制成功：** `OnResumeRemoteRecordSucc(record_id)`

回调参数	类型	说明
record_id	String	服务端录制 ID

  - **恢复服务端录制失败：** `OnResumeRemoteRecordFailed(record_id, err_code, err_msg)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	int	恢复服务端录制失败返回： ▪ -10361：房间状态错误 ▪ -10362：录制状态错误 ▪ -10363：服务端错误
err_msg	String	状态信息

ChangeRemoteRecordWatermark

变更服务端录制水印。

- **参数：**

请求参数	类型	说明
record_id	String	服务端录制 ID
overlaps	Jsonarray	见 附录 recordParam 中的 `overlaps` 。

- **回调接口异步返回：**

- 变更服务端录制水印成功：`OnChangeRemoteRecordWatermarkSucc(record_id)`

回调参数	类型	描述
record_id	String	服务端录制 ID

- 变更服务端录制水印失败：`OnChangeRemoteRecordWatermarkFailed(record_id, err_code, err_msg)`

回调参数	类型	说明
record_id	String	服务端录制 ID
err_code	int	状态码
err_msg	String	状态信息

PauseFile

暂停共享音频或视频播放。

- **参数：**

请求参数	类型	说明
sid	int	流 ID

- **回调接口异步返回：**
  - 暂停共享文件成功：`OnPauseFileSuccess`

    回调参数：无。
  - 暂停共享文件失败：`OnPauseFileFail`

    回调参数：无。

PlayFile

恢复播放共享音频或视频播放。

- **参数：**

请求参数	类型	说明
sid	int	流 ID

- **回调接口异步返回：**

- 恢复播放共享文件成功：`OnPlayFileSuccess`

  回调参数：无。
- 恢复播放共享文件失败：`OnPlayFileFail`

  回调参数：无。

ChangeStreamSize

动态修改视频尺寸。

- **参数：**

请求参数	类型	是否必填	说明
sid	int	-	流 ID
video_profile_type	String	选填	本地视频的清晰度档位： ◦ 1：720P，1280x720，16:9 ◦ 2：VGA，640x480，4:3 ◦ 3：360P，640x360，16:9 ◦ 4：720P（竖），720x1280，9:16 ◦ 5：360P（竖），360x640，9:16 ◦ 6：1080P，1920x1080，16:9 ◦ 7：1080P（竖），1080x1920，9:16 ◦ 8：90P（竖），90x160，9:16 ◦ 9：540P，960x540，16:9 ◦ 10：540P（竖），540x960，9:16 ◦ 100：自定义

- **回调接口异步返回：**
  - 修改视频尺寸成功：`OnChangeStreamSizeSuccess(sid, localVideoWidth, localVideoHeight)`

请求参数	类型	说明
sid	int	流 ID
localVideoWidth	int	切换后的视频宽度
localVideoHeight	int	切换后的视频高度

- **修改视频尺寸失败：** `OnChangeStreamSizeFail(sid, localVideoWidth, localVideoHeight, errCode, errMsg)`

请求参数	类型	说明
sid	int	流 ID
localVideoWidth	int	切换后的视频宽度
localVideoHeight	int	切换后的视频高度
err_code	int	状态码
err_msg	String	状态信息

SendData

通过 DataChannel 发送信息。

参数：

请求参数	类型	说明
data	String（目前仅支持文本）	发送文本信息

6.3.3. 回调接口

本文介绍音视频通话 API 在接入 Web 端时涉及的回调接口。

OnMediaCallSucc(sid)

初始化音视频成功。

OnMediaCallFail(err_code, err_msg)

初始化音视频失败。

参数说明：

回调参数	类型	说明

回调参数	类型	说明
err_code	int	初始化音视频失败返回： • 5101：视频启动失败 • 5102：音频启动失败 • 5103：浏览器不支持 • 5104：房间状态不正确 • 5105：共享桌面启动失败 • 5106：发布获取文件流失败 • 5107：发布文件不存在 • 5108：发布区域不存在 • 5109：背景虚化图片未提供 • 5110：自定义流未提供 • 5111：自定义分辨率、帧率、码率参数未提供
err_msg	String	状态信息

OnInviteRequest(roomId, rtoken, inviter, extra)

被邀请者收到“加入房间”的邀请消息。

参数说明：

回调参数	类型	说明
roomId	String	房间 ID
rtoken	String	房间 token
inviter	String	邀请方的 UID
extra	String	业务透传字段

OnInviteReply(invitee, reply)

邀请者收到“邀请加入房间”的回复消息。

参数说明：

回调参数	类型	说明
invitee	String	被邀请方的 UID

回调参数	类型	说明
reply	int	• 0：接听 • 1：离线 • 2：拒绝

OnRoomAttendanceList(participants)

推送“房间与会者列表”给新加入者。

参数说明：

回调参数	类型	值	值类型	说明
participants	jsonarray	uid	String	房间与会者的 UID
		userType	int	与会者的用户类型： • 0：普通用户 • 101：rtcvod 点播服务器
		publish	jsonarray	发布流
		feedId	String	房间与会者的发布流 ID
		tag	String	feedId 对应的 tag

OnNewJoinerIn(participant)

推送“新加入房间者”给与会者。

参数说明：

回调参数	类型	说明
participant	String	新加入房间者的 UID

OnNewPublish(feed)

推送“有新发布”给与会者。

参数说明：

回调参数	类型	说明
feed	JSON	新发布的流信息： • uid：流所属的用户 • feedId：发布流 • tag：标签 • mediaSource：媒体源

OnNewSubscribe(subscriber,feed)

将“有新订阅”的消息推送给与会者。

参数说明：

回调参数	类型	说明
subscriber	String	新订阅者 ID
feed	JSON	新订阅的流信息： • uid：流所属的用户 ID • feedId：被订阅流 ID

OnUnSubscribe(unsubscriber,feed)

推送“取消订阅”给与会者。

参数说明：

回调参数	类型	说明
unsubscriber	String	取消订阅者 ID
feed	JSON	取消订阅的流信息： • uid：流所属的用户 ID • feedId：被订阅流 ID

OnMediaClose(code，sid)

音视频关闭通知。

参数说明：

回调参数	类型	说明

回调参数	类型	说明
code	int	正常状态： • 5001：取消发布/订阅触发关闭流 • 5002：对端离开房间，被动关闭流 • 5006：对端取消发布，被动关闭流 • 5008：退出房间触发关闭流 异常状态： • 5003：启动流失败（信令阶段） • 5004：流建立连接超时 • 5005：流建立连接成功，但房间状态异常 • 5007：媒体链路连接关闭 • 5009：媒体服务器通过信令通道发出关闭指令 • 5010：媒体链路断开连接超时 • 5011：媒体链路连接失败 • 5012：发布初始化媒体源失败
sid	int	流 ID

OnReceiveTextMsg(uid, msg)

推送文本消息。

参数说明：

回调参数	类型	说明
uid	String	文本消息发送人
msg	String	消息文本

OnParticipantLeaveRoom(participant,exitType)

推送“退出房间者”给与会者。

参数说明：

回调参数	类型	说明
participant	String	退出者 ID

回调参数	类型	说明
exitType	int	退出类型： • 1：正常退出 • 2：异常退出

OnNetworkWeak(bpsSend, bpsRecv,sid)

弱网回调。

参数说明：

回调参数	类型	说明
bpsSend	String	每秒发送数据，单位为 bit
bpsRecv	String	每秒接收数据，单位为 bit
sid	int	流 ID，便于区分哪一路流出现弱网

OnVodOver(vod_id)

语音点播结束通知。

参数说明：

回调参数	类型	说明
vod_id	String	点播 ID

OnVolumeAnalyser(sid, analyser)

实时音量数据回调。

参数说明：

回调参数	类型	说明
sid	String	流 ID
analyser	AnalyserNode	返回 AnalyserNode 对象

OnClientRecordBlob(clientRecordId, blob)

浏览器录制推送录制 blob 数据。

参数说明：

回调参数	类型	说明
clientRecordId	String	浏览器录制 ID
blob	Blob	通过 Blob 对象将数据保存成多媒体文件

OnRemoteRecordOver(record_id)

服务端录制结束通知。

参数说明：

回调参数	类型	说明
record_id	String	录制 ID

OnRemoteRecordingFailed(recordId, feedId, code, msg)

服务端录制失败通知。

参数说明：

回调参数	类型	说明
recordId	String	录制 ID
feedId	String	与事件相关的流 ID
code	int	服务端录制失败通知返回： • -12481：eventCode 值为 1 • -12482：eventCode 值为 2
msg	String	状态信息

OnReceiveData

接收 Channel 数据。

参数说明：

回调参数	类型	说明
data	String	回调参数接收信息

StreamFilterHandler

业务自定义处理流。

> 说明
> StreamFilterHandler 需要是异步的函数或者返回 Promise 对象，SDK 使用 `then()` 接收处理后的 Stream。

参数说明：

回调参数	类型	说明
publish_tag	String	业务自定义的标记内容
stream	MediaStream	需要处理的流
stream_type	String	发布类型：publish（发布流）或者 subscribe（订阅流）
publish_device	int	当 stream_type 的值为 publish 才涉及此参数，参考 InitRoomConfig。
media_type	String	回调参数接收信息

OnDesktopDisplayClosed

共享屏幕关闭回调。

回调参数：无。

OnFileStreamClosed

OnFileStreamClosed 中的共享文件关闭回调。

回调参数：无。

OnParticipantEvent

对端发送事件回调。

参数说明：

回调参数	类型	说明
uid	String	对端用户
eventType	int	事件类型
eventDescription	String	事件描述

回调参数	类型	说明
eventExtra	Object	事件数据（可选）

6.3.4. 错误码

本文介绍音视频通话 Web API 的错误码及其说明。

建立通话相关错误码

错误码	说明
-101	信令连接初始化超时。
-102/1004	信令连接初始化错误。
-201	初始化房间失败：房间状态非法。
-202	初始化房间失败：初始化房间参数非法。
-203	初始化房间失败：初始化房间错误，P2P 模式下才能开启 Data Channel。
-301	创建房间失败：房间未初始化。
-302	创建房间失败：服务器返回错误。
-303	创建房间失败：创建房间超时。
-304	创建房间失败：在创建房间后，自启动发布的情况下，media 初始化超时。
-401	加入房间失败：房间未初始化。
-402	加入房间失败：参数错误。
-403	加入房间失败：服务器返回错误。
-404	加入房间失败：加□房间超时。
-501	邀请对方加入房间失败：房间状态非法。

错误码	说明
-502	邀请对方加入房间失败：服务器返回错误。
-503	邀请对方加入房间失败：参数错误。
-601	用户答复邀请失败：房间状态错误。
-602	用户答复邀请失败：服务器返回错误。

发布/订阅媒体流相关错误码

错误码	说明
-1071	发布媒体流失败：服务器返回错误。
-1072	发布媒体流失败：未设置视频标签。
-1073	发布媒体流失败：超出发布限额。
-1074	发布媒体流失败：发布建立媒体链路连接超时。
-1081	订阅媒体流失败：服务器返回错误。
-1082	订阅媒体流失败：视频窗口的 videoId 已使用。
-1083	订阅媒体流失败：超出订阅限额。
-1084	订阅媒体流失败：订阅建立媒体链路失败。
-1085	订阅媒体流失败：订阅的窗口的 videoId 为空。

切流相关错误码

错误码	说明
-1101	切流失败：当前 sid 未发布。

错误码	说明
-1102	切流失败：切流参数错误。
-1104	切流失败：获取音视频失败。
-1105	切流失败：共享屏幕流为空。
-1106	切流失败：获取视频失败。
-1107	切流失败：获取文件流失败。
-1108	切流失败：获取音频失败。
-1109	切流失败：浏览器不支持音视频。
-1110	切流失败：未提供自定义流。

服务端录制相关错误码

错误码	说明
-10311	服务端录制失败：服务端返回录制启动失败。
-10312	服务端录制失败：开启录制超时。
-10313/-10314	服务端录制失败。
-10291	服务端返回错误。
-10321	切换视频分辨率失败。
-10341	暂停服务端录制：房间状态错误。
-10342	暂停服务端录制：录制状态错误。
-10343	暂停服务端录制：服务端错误。

错误码	说明
-10351	获取服务端录制结果失败：连接状态错误。
-10352	获取服务端录制结果失败：服务端返回错误。
-10353	获取服务端录制结果失败：请求参数错误。
-10361	恢复服务端录制：房间状态错误。
-10362	恢复服务端录制：录制状态错误。
-10363	恢复服务端录制：服务端错误。

语音点播相关错误码

错误码	说明
-10371	开始语音点播失败：服务端返回错误。
-10372	开始语音点播失败：房间状态错误。
-10373	开始语音点播失败：服务端通知开启失败。
-10374	开始语音点播失败：点播参数有误。
-10375	开始语音点播失败：开启点播超时。
-10381	变更服务端录制水印失败。
-10391	停止语音点播失败：服务端返回失败。
-10392	停止语音点播失败：房间状态错误。
-10393	停止语音点播失败：点播 ID 错误。

浏览器录制相关错误码

错误码	说明
-12481/-12482	服务端录制失败。
-20301	开启浏览器录制失败：房间状态错误。
-20302	开启浏览器录制失败：已经在录制中。
-20303	开启浏览器录制失败：录制 ID 已经使用。
-20304	开启浏览器录制失败：未发布或未订阅。
-20305	开启浏览器录制失败：mediaType 参数错误。
-20306	开启浏览器录制失败：stream_list 长度错误。
-20307	开启浏览器录制失败：没有要求录制的视频流或者音频流。
-20308	开启浏览器录制失败：record_resolution 错误。
-20331	停止浏览器录制失败：房间状态错误。
-20371	暂停浏览器录制失败：房间状态错误。
-20372	暂停浏览器录制失败：未开启浏览器录制。
-20391	继续浏览器录制失败：房间状态错误。
-20392	继续浏览器录制失败：未开启浏览器录制。
-20351	下载浏览器录制文件失败：房间状态错误。
-20352	下载浏览器录制文件失败：未开启浏览器录制。
-20353	下载浏览器录制文件失败：未停止浏览器录制。
-30101	获取设备信息失败：浏览器不支持音视频。

错误码	说明
-30102	获取设备信息失败：获取媒体错误。

音视频关闭通知错误码

状态	错误码	说明
正常状态	5001	取消发布/订阅触发关闭流。
	5002	对端离开房间，被动关闭流。
	5006	对端取消发布，被动关闭流。
	5008	退出房间触发关闭流。
异常状态	5003	启动流失败（信令阶段）。
	5004	流建立连接超时。
	5005	流建立连接成功，但房间状态异常。
	5007	媒体链路连接关闭。
	5009	媒体服务器通过信令通道发出关闭指令。
	5010	媒体链路断开连接超时。
	5011	媒体链路连接失败。
	5012	发布初始化媒体源失败。

发布相关错误码

错误码	错误信息
5101	发布启动失败：视频启动失败。

错误码	错误信息
5102	发布启动失败：音频启动失败。
5103	发布启动失败：浏览器不支持媒体。
5104	发布启动失败：房间状态非法。
5105	发布启动失败：共享桌面启动失败。
5106	发布启动失败：获取文件流失败。
5107	发布启动失败：发布文件不存在。
5108	发布启动失败：发布区域不存在。
5109	背景虚化图片未提供。
5110	自定义流未提供。

切换编码格式相关错误码

错误码	说明
-5201	切换编码格式：房间状态不正确。
-5202	切换编码格式：无法找到流信息。
-5203	切换编码格式：此编码格式当前不可用。
-5204	切换编码格式：setRemoteDescription 失败。
-5205	切换编码格式：setLocalDescription 失败。
-5206	切换编码格式：createOffer 失败。

6.4. Linux API

6.4.1. 概述

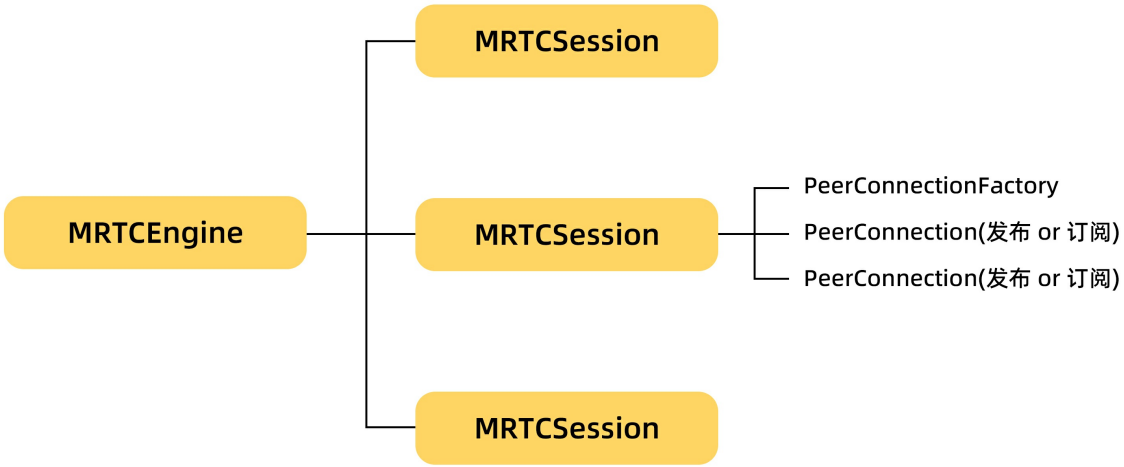
在某些可以智能化处理的场景，比如针对某些业务场景打造的智能音视频客服（虚拟人），需要提供基于 Linux 平台的 WebRTC-SDK，它可以帮助您的服务端接入 WebRTC 服务与 Android、iOS、Web 端的真实用户进行实时的音视频通话。

目前 SDK 主要包括以下三个方面：

- 在音频方面，支持输入和输出都为 16k 采样率的单声道 PCM，同时也支持 48k 双声道的输入和输出。
- 在视频方面，支持输入和输出 I420P YUV 格式的数据。
- 在某些硬编平台，考虑到性能，支持 H264-NALU 输入和输出，此场景下，H264 编解码由您来完成。

接口设计

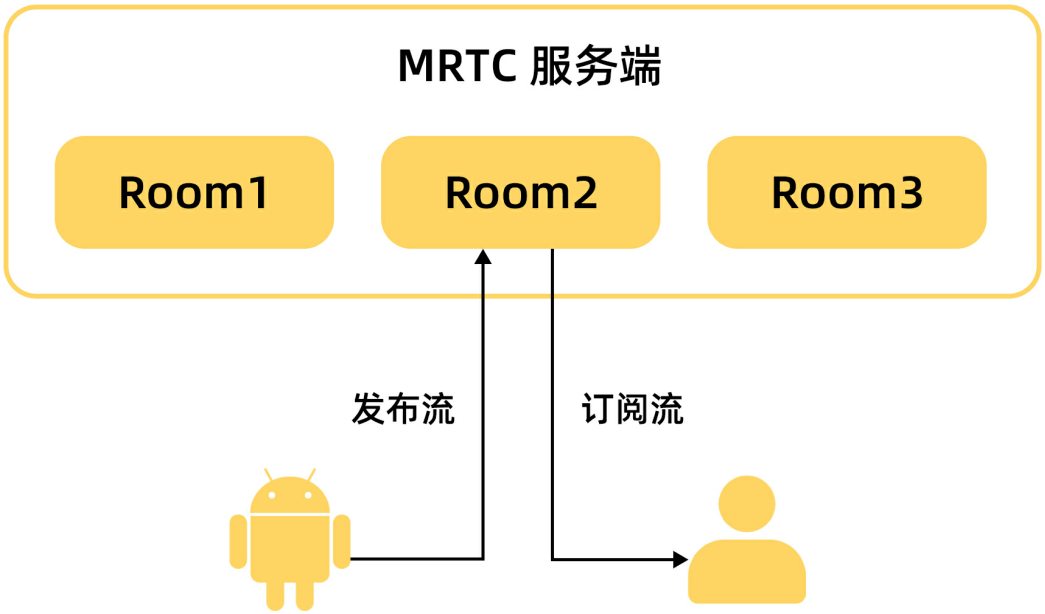
MRTCEngine 单例负责 MRTCSession 的创建。每个 MRTCSession 代表在某个房间里面的□次通话过程。



重要概念

参数	说明
uid	用户 ID，标识房间里面的用户，全局唯□性。
roomId	房间 ID，通话双方必须加入同□个房间。
sessionId	会话 ID，客户端创建或者加入房间以后，服务端会为客户端的这次通话分配唯□的 ID，整个通话过程中，此 ID 维持不变。
streamId	流 ID，标识会话过程中用户的某次发布或者订阅操作，比如您发布□路音频流，发布□路视频流，服务器会为您分配两个流 ID，如果您还订阅了□路来自其他用户的流，针对此次订阅操作，系统也会分配□个流 ID。
recordId	录制 ID，如果业务需要录制能力，那么创建房间成功后，除了返回相应的 roomId，还会返回□个 recordId，有了这个录制 ID，您可以从 OSS 上把这个录制文件下载下来。

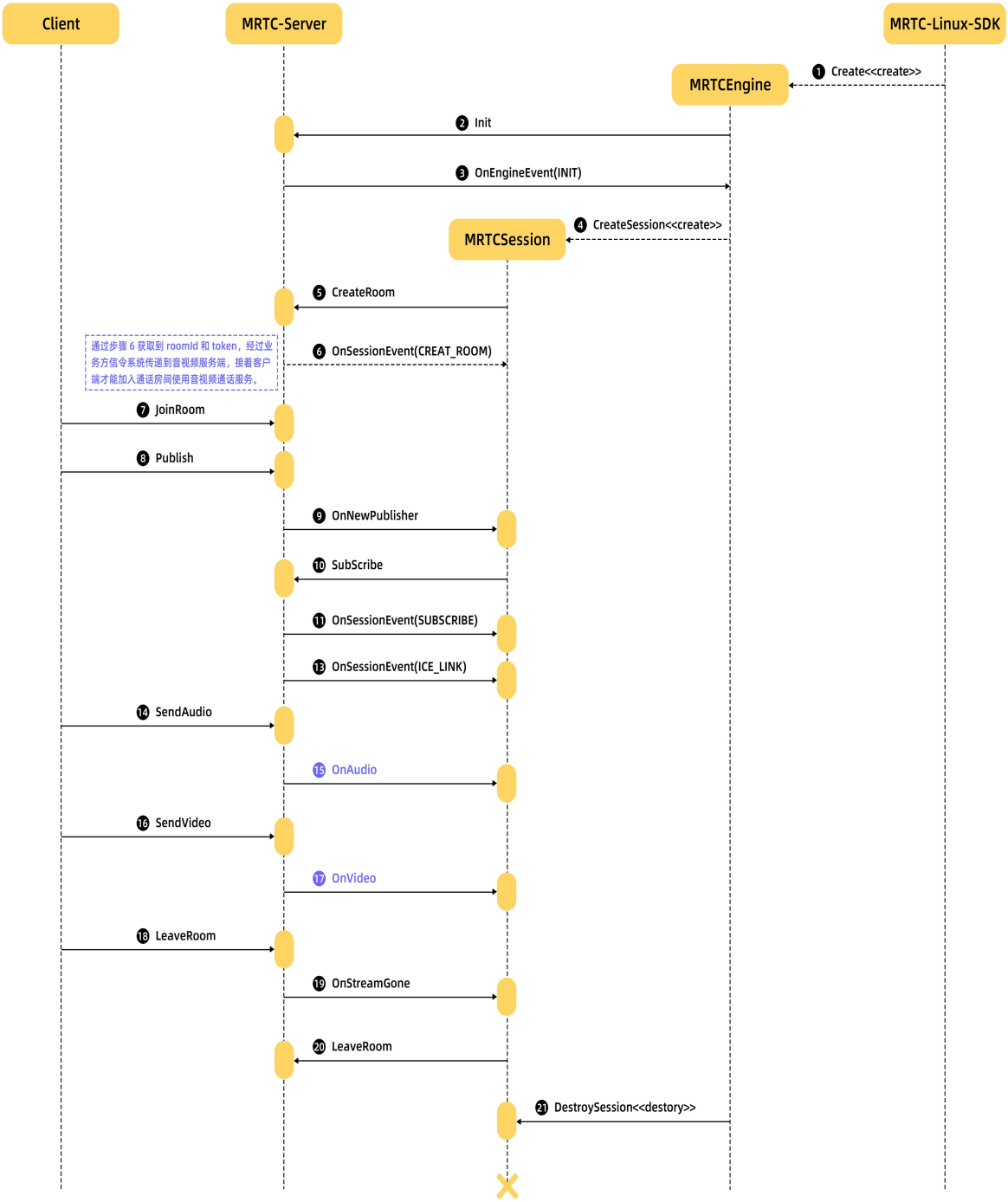
音视频流的管理模式



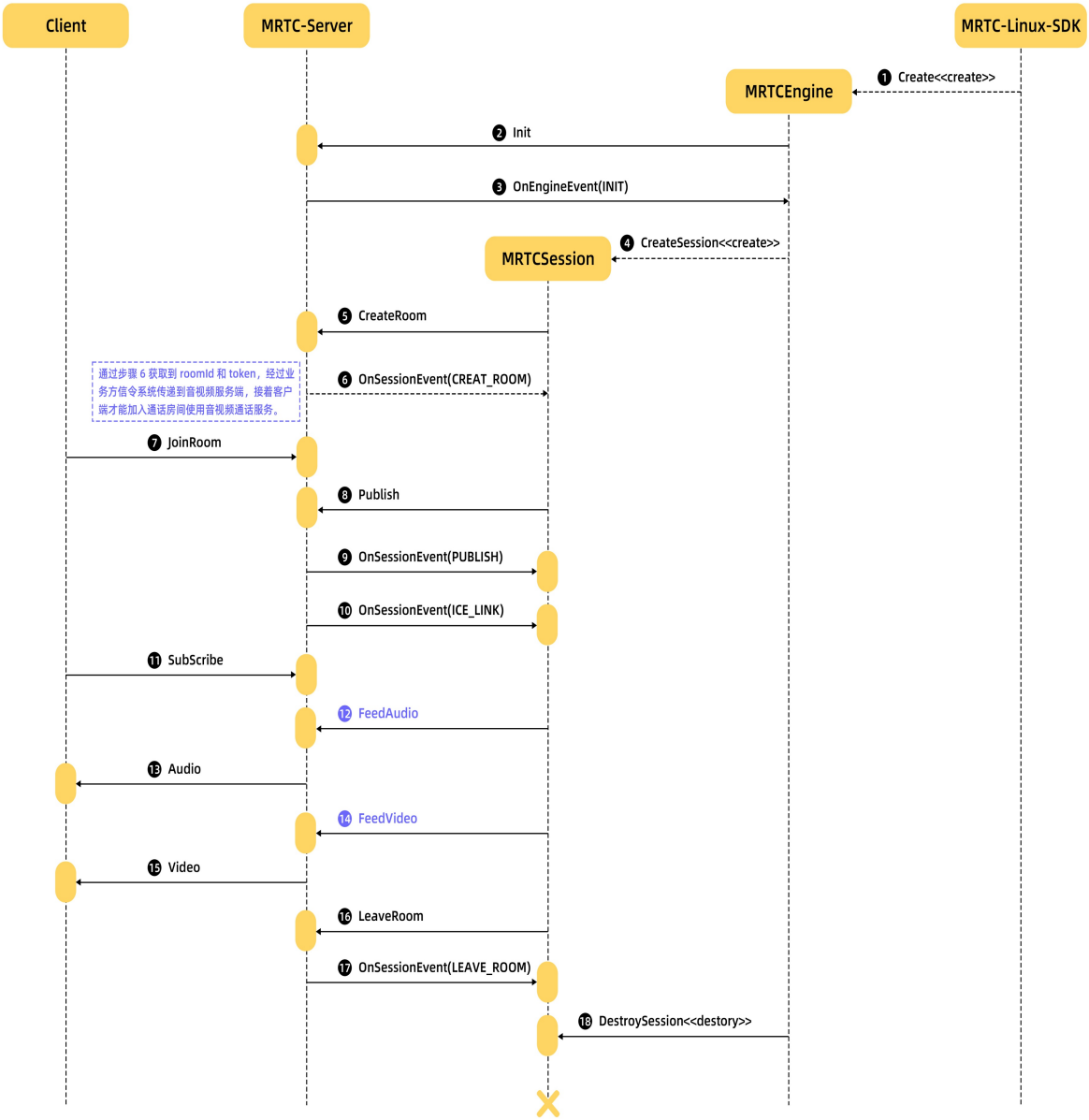
上图简要描述了 MRTC 系统在客户端和服务器之间的音视频流管理模式。整体的框架是基于发布和订阅的模式进行设计的。客户端创建或者加入某个房间后通过发布操作，发布□路音频或者视频流到达服务器，接着服务器生成唯□的流 ID 对这路流进行标识，那么订阅方加入房间后，就可以通过订阅操作对这路流进行订阅。

调用流程

下图中紫色标识的步骤表示音视频服务端 **接收** 来自客户端的音视频数据。



下图中紫色标识的步骤表示音视频服务端 **发送** 音视频数据到客户端。



6.4.2. 接口定义

本文介绍创建引擎、初始化引擎、引擎创建/销毁会话、获取引擎版本号和监听引擎事件的相关接口，以及会话相关的接口，包括创建一个房间、加入已有的房间、发布和订阅音视频流等。

引擎相关

创建引擎

```
enum EventType {
    INIT = 0,                          //引擎类：初始化结果
    WSS_LINK,                          //引擎类：和房间服务器的 WebSocket 的连接状态
    CREATE_ROOM,                       //会话类：创建房间
    JOIN_ROOM,                         //会话类：加入房间
    PUBLISH,                           //会话类：发布
    SUBSCRIBE,                         //会话类：订阅
    EXIT_ROOM,                         //会话类：退出房间
    ICE_LINK,                          //会话类：音视频数据通道联调状态
    START_RECORD                       //会话类：启动录制的结果
};
struct RtcEvent {
    std::string uid;                   //事件关联的用户 ID,目前 uid 只支持英文字母、数字、下划线
的组合，且长度不超过 128 个字符。
    EventType type;                    //事件类型
    int32_t code;                      //事件结果，小于 0 表示错误，等于 0 表示正常
    std::string ext = "";              //扩展字段，格式和事件类型相关，参考问题列表章节
};
struct MRtcEngineListener {
    MRtcEngineListener() {};
    virtual ~MRtcEngineListener(){};
    virtual void OnEngineEvent(RtcEvent event) = 0;
};
static MRtcEngine* Create(MRtcEngineListener* engineListener);
```

初始化引擎

```
enum RtcVideoCodec {
    VIDEO_CODEC_YUV420P,          //视频输入输出是未编码的 YUV 数据（I420P）
    VIDEO_CODEC_H264,             //视频输入输出是编码好的 H264 数据
};
enum RtcAudioCodec {
    AUDIO_CODEC_PCM,              //音频当前只支持 PCM 格式的输入和输出
};
struct RtcVideoData {
    const uint8_t *dataY;       //引擎初始化为 VIDEO_CODEC_YUV420P 时为 Y 分量缓存，其他情况为
编码后视频帧缓存地址
    const uint8_t *dataU;       //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    const uint8_t *dataV;       //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideY;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideU;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideV;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t length;             //视频数据长度，引擎初始化为 VIDEO_CODEC_H264 必填，其他情况无
需关注
    int32_t width;              //视频宽
    int32_t height;             //视频高
    int64_t timestamp;          //视频时间戳，必填
};
struct RtcAudioFormat {
    RtcAudioFormat():channels(1),sampleRate(16000),bytesPerSample(2),audioCodec(AUDIO_COD
_PCM){}
    int32_t            channels;               //音频声道数
    int32_t            sampleRate;             //音频采样率
    int32_t            bytesPerSample;         //单个样本点字节数
    RtcAudioCodec      audioCodec;
};
struct MRtcEngineInitParam {
    std::string roomUrl;              //房间服务器的接入地址
    RtcVideoFormat videoFormat;       //视频输入输出格式
    RtcAudioFormat audioFormat;       //音频输入输出格式
    std::string logLevel = "debug";   //日志级别，none|error|warn|info|debug 中的某一个，non
e 表示不打印日志
    int32_t maxSessionNum = 20;       //运行并发的最大会话数量,大于 0 时配置生效
    bool enableRelay = false;         //如果服务器没有公网 IP，可以考虑打开该开关，使用数据中转模
式
    bool enableDataChannel = false;   //是否开启 DataChannel 功能
};
int32_t Init(const MRtcEngineInitParam& param);
```

引擎创建□个会话

```
/**
 *  会话监听器，可以侦听会话中产生的一些事件，比如发布成功，订阅成功等
 *  当有新的发布者加入或者离开会话，会得到通知，告知该发布者的一些信息，包括使用的流 ID，用户 ID 等
 *  如果订阅了某个发布者的流数据，那么还会收到其相应的音视频数据
**/
struct MRtcSessionListener {
    MRtcSessionListener() {};
    virtual ~MRtcSessionListener(){};
    //会话事件回调
    virtual void OnSessionEvent(RtcEvent event) = 0;
    //房间里面有人发布了一条新流，可供订阅
    virtual void OnNewPublisher(StreamInfo info) = 0;
    //房间里面某条流取消了发布，离开了房间
    virtual void OnStreamGone(StreamInfo info) = 0;
    //有新的订阅者订阅了某条流
    virtual void OnNewSubscriber(StreamInfo info) = 0;
    //混音后的音频数据,该接口保留,两人场景下没有用到
    virtual void OnMixedAudio(const RtcAudioData& audio) = 0;
    //收到某条流的音频数据
    virtual void OnAudio(const RtcAudioData& audio,const std::string& streamId) = 0;
    //收到某条流的视频数据
    virtual void OnVideo(const RtcVideoData& video,const std::string& streamId) = 0;

    //H264-NALU 对接的场景下，需要反馈一些信息 给到编码器做实时调节
    //请求本地编码器下一帧编码关键帧
    virtual void OnKeyFrameRequest() {}
    //实时控制编码器的一些编码参数,比如输出帧率和码率
    virtual void OnEncoderQosRequest(const MRtcEncoderQosParams& params) {}
};
/**
 *  根据底层网络状态的反馈情况以及订阅方视频解码的情况，
 *  实时调节上次业务方编码器的输出帧率和码率
 *  业务方可以根据编码器的特性，尽可能的做出对应的调整
**/
struct MRtcEncoderQosParams {
    int target_kbps;
    int target_fps;
};
MRtcSession* CreateSession(MRtcSessionListener* sessionListener);
```

引擎销毁一个会话

```
void DestroySession(MRtcSession* session);
```

获取引擎的版本号

```
std::string GetVersion();
```

监听引擎事件

```
void OnEngineEvent(RtcEvent event);
```

会话相关

创建一个房间

```
struct CreatRoomParam {
    std::string uid;                    //用户 ID，作为房间里某个用户的唯一标识
    std::string sign;                   //房间服务器需要验证该通话业务的合法性
    std::string bizName;                //业务类型，比如 Bank（银行类业务）、Stock（证券类业务
）
    std::string subBiz;                 //子业务类型，比如上海银行，平安证券等
    std::string workspaceId;            //MPaaS 类业务需要填写，其他业务不需要该字段
    bool autoSubscribe;                 //当房间里面有新人进入，发布了新的流，SDK 是否自动订阅
    EngineType engine = ENGINE_ALIPAY;  //媒体数据使用 P2P 模式还是 SFU 模式，默认使用 SFU 模
式（中转）
    std::string ext;                    //留作扩展，可用来配置录制相关的特性，参见问题列表
};
void CreateRoom(const CreatRoomParam& createParam);
```

加入已有房间

```
struct JoinRoomParam {
    std::string roomId;
    std::string uid;
    std::string sign;
    std::string bizName;
    std::string subBiz;
    std::string token;                  //加入房间的凭证
    std::string workspaceId;            //MPaaS 类业务需要填写，其他业务不需要该字段
    bool autoSubscribe;                 //是否自动订阅房间里面发布者发布的音视频流
    EngineType engine = ENGINE_ALIPAY; //媒体数据使用 P2P 模式还是 SFU 模式，默认使用 SFU 模式
（中转）
    std::string ext;                    //留作扩展，暂时未用
};
void JoinRoom(const JoinRoomParam& joinParam);
```

发布音视频流

```
struct PublishParam {
    PublishParam():enableVideo(false),enableAudio(true){}
    bool enableVideo;                 //是否发布本地视频到视频服务器
    bool enableAudio;                 //是否发布本地音频到视频服务器
};
void Publish(const PublishParam& pubParam);
```

订阅音视频流

```
struct SubscribeParam {
    SubscribeParam():enableVideo(false),enableAudio(true){}
    bool enableVideo;                //手动订阅的场景下，是否订阅视频
    bool enableAudio;                //手动订阅的场景下，是否订阅音频
    std::string streamId;            //房间里面的每个发布者都有唯一的一个流标识，这个标识指明去订
阅哪个参与者的音视频流
};
void Subscribe(const SubscribeParam& subParam);
```

从服务端输入音频数据发送到客户端

```
struct RtcAudioData {
    const uint8_t *data;            //必填，音频数据缓存
    int32_t length;                 //必填，音频数据长度
    int64_t timestamp;              //必填，音频时间戳
};
void FeedAudio(const RtcAudioData& audio);
```

从服务端输入视频数据发送到客户端

```
struct RtcVideoData {
    const uint8_t *dataY;       //引擎初始化为 VIDEO_CODEC_YUV420P 时为 Y 分量缓存，其他情况为
编码后视频帧缓存地址
    const uint8_t *dataU;       //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    const uint8_t *dataV;       //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideY;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideU;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t strideV;            //引擎初始化为 VIDEO_CODEC_YUV420P 时必填，其他情况无需关注
    int32_t length;             //视频数据长度，引擎初始化为 VIDEO_CODEC_H264 必填，其他情况无
需关注
    int32_t width;              //视频宽
    int32_t height;             //视频高
    int64_t timestamp;          //视频时间戳，必填
};
void FeedVideo(const RtcVideoData& video);
```

发送文本消息

peers 存放接收端的 UID，文本消息底层使用 WSS 协议通过 Room 中转。

```
void SendText(const std::string& text,const std::list<std::string> peers);
```

发送文本消息

与 SendText 的区别在于，底层是采用的 WebRtc 的 DataChannel 传输通道。

```
void SendData(const std::string& data);
```

离开房间

```
void LeaveRoom();
```

清除音频发送队列

这些数据将不会发送到客户端，实现打断的功能。

```
void ClearAudio();
```

音频发送队列大小

获取音频发送队列当前剩余的数据量，用于判断当前所有后台产生的语音包是否已经发送到端。

```
int32_t AudioQueueSize();
```

6.4.3. 重要参数

本文介绍的是音视频引擎与会话接口的相关参数。

参数	说明
MRtcEngineInitParam	引擎初始化。
MRtcEncoderQosParams	音视频编码服务质量。
CreatRoomParam	创建一个房间。
JoinRoomParam	加入已有房间。
PublishParam	发布音视频流。
SubscribeParam	订阅音视频流。

MRtcEngineInitParam

引擎初始化。作为引擎初始化的参数，只有该参数填写正确，才能顺利创建会话。

参数	类型	说明	是否必填
roomUrl	String	房间服务器的接入地址。	是
logLevel	String	日志级别，none|error|warn|info|debug 中的某一个，none 表示不打印日志。	是
maxSessionNum	Int	运行并发的最大会话数量,大于 0 时配置生效。	是

参数	类型	说明	是否必填
enableRelay	Bool	如果服务器没有公网 IP，可以考虑打开该开关，走数据中转模式。	否
enableDataChannel	Bool	是否开启 DataChannel 功能。	否

MRtcEncoderQosParams

保证音视频服务提供的质量。实时调节编码器的输出帧率和码率。

参数	类型	说明	是否必填
target_kbps	Int	编码器的输出码率。	是
target_fps	Int	编码器的输出帧率。	是

CreatRoomParam

创建房间。作为创建房间的参数，只有该参数填写正确，才能顺利创建音视频房间。

参数	说明
uid	用户 ID，唯一标识房间里的某个用户。
sign	房间服务器需要验证该通话业务的合法性。
bizName	业务类型，比如 Bank（银行类业务）、Stock（证券类业务）。
subBiz	子业务类型，比如上海银行，平安证券等。
workspaceId	mPaaS 类业务需要填写，其他业务不需要该字段。
autoSubscribe	当房间里面有新人进入，发布了新的流，SDK 是否自动订阅。
ext	留作扩展，可用来配置录制相关的特性，参见 常见问题。

JoinRoomParam

加入房间。作为进房参数，只有该参数填写正确，才能顺利进入 roomId 所指定的音视频房间。

参数	说明
roomId	创建的房间 ID，唯一标识。
uid	用户 ID，唯一标识房间里的某个用户。
sign	房间服务器需要验证该通话业务的合法性。
bizName	业务类型，比如 Bank（银行类业务）、Stock（证券类业务）。
subBiz	子业务类型，比如上海银行，平安证券等。
token	加入房间的凭证。
workspaceId	mPaaS 类业务需要填写，其他业务不需要该字段。
autoSubscribe	是否自动订阅房间里面发布者发布的音视频流。
ext	留作扩展，可用来配置录制相关的特性，参见 常见问题。

PublishParam

发布音视频流。是否发布本地音视频到视频服务器。

参数	说明
enableVideo	是否发布本地视频到视频服务器。
enableAudio	是否发布本地音频到视频服务器。

SubscribeParam

订阅音视频流。手动订阅的场景下，是否订阅音视频。

参数	说明
enableVideo	手动订阅的场景下，是否订阅视频。
enableAudio	手动订阅的场景下，是否订阅音频。

参数	说明
streamId	房间里面的每个发布者都有唯一的一个流标识，这个标识指明去订阅哪个参与者的音视频流。

6.4.4. 主调函数

接口主要分为引擎与会话两类，会话由引擎创建。

主调引擎函数对应会话控制回调函数。

```
/**
 *  音视频引擎，用于创建，销毁会话
 **/
struct MRtcEngine {
    static MRtcEngine* Create(MRtcEngineListener* engineListener);
    virtual int32_t Init(const MRtcEngineInitParam& param) = 0;
    virtual void Destroy() = 0;
    virtual MRtcSession* CreateSession(MRtcSessionListener* sessionListener) = 0;
    virtual void DestroySession(MRtcSession* session) = 0;
    virtual std::string GetVersion() = 0;
};
```

主调引擎监听函数对应会话监听回调函数。

```
/**
 *  引擎监听器，监听一些引擎相关的事件，比如初始化成功，失败等等
 **/
struct MRtcEngineListener {
    MRtcEngineListener() {};
    virtual ~MRtcEngineListener(){};
    virtual void OnEngineEvent(RtcEvent event) = 0;
};
```

6.4.5. 回调函数

本文对引擎创建的会话相关的回调函数进行说明。

会话控制

其中包含发起人创建房间、发布音视频流、订阅方加入房间等回调通知。

```
    /**
     *  会话控制，每次通话对应一个会话
     *  用于创建房间，加入房间、发布、订阅等操作，只有加入或者创建房间成功，才会分配合法的 sessionId
     *  对于房间创建者，典型流程是：CreateRoom -> Publish -> FeedAudio/FeedVideo -> LeaveRoo
m，后续有人加入再做 Subscribe
     *  对于房间加入者，典型流程是：JoinRoom -> Subscribe -> LeaveRoom，当然，JoinRoom 以后也
可以做 Publish
     *  另外还支持通过 SendMessage 接口发送文本消息到对端，peers 存放接收方的 uid 列表
     **/
    struct MRtcSession {
        MRtcSession() {};
        virtual ~MRtcSession() {};
        virtual void CreateRoom(const CreatRoomParam& createParam) = 0;
        virtual void JoinRoom(const JoinRoomParam& joinParam) = 0;
        virtual void Publish(const PublishParam& pubParam) = 0;
        virtual void Subscribe(const SubscribeParam& subParam) = 0;
        virtual void FeedAudio(const RtcAudioData& audio) = 0;
        virtual void FeedVideo(const RtcVideoData& video) = 0;
        virtual void SendText(const std::string& text,const std::list<std::string> peers)
 0;
        virtual void LeaveRoom() = 0;
        virtual std::string GetSessionId() = 0;
        virtual void ClearAudio() = 0;              //清除音频队列数据，实现打断机器人说话的功
能
        virtual int32_t AudioQueueSize() = 0;       //查看音频队列大小，一定程度上可以判断机器
人说的话是不是发送完毕了
     virtual void StartRecord() = 0;
    };
```

会话监听器

音视频流发布成功、加入方订阅成功等事件均会触发相应的回调通知。

```
    /**
     *  会话监听器，可以侦听会话中产生的一些事件，比如发布成功，订阅成功等
     *  当有新的发布者加入或者离开会话，会得到通知，告知该发布者的一些信息，包括使用的流 ID、用户 ID
等
     *  如果订阅了某个发布者的流数据，那么还会收到其相应的音视频数据
     **/
    struct MRtcSessionListener {
        MRtcSessionListener() {};
        virtual ~MRtcSessionListener(){};
        virtual void OnSessionEvent(RtcEvent event) = 0;
        virtual void OnNewPublisher(StreamInfo info) = 0;
        virtual void OnStreamGone(StreamInfo info) = 0;
        virtual void OnNewSubscriber(StreamInfo info) = 0;
        virtual void OnMixedAudio(const RtcAudioData& audio)
{}                          //混音后的音频数据，该接口保留，两人场景下没有用到
        virtual void OnAudio(const RtcAudioData& audio,const std::string& streamId) = 0;

        virtual void OnVideo(const RtcVideoData& video,const std::string& streamId) = 0;

        virtual void OnText(const RtcTextData& text) = 0;

        //H264-NALU 对接的场景下，需要反馈一些信息 给到编码器做实时调节
        virtual void OnKeyFrameRequest() {}
        virtual void OnEncoderQosRequest(const MRtcEncoderQosParams& params) {}

        //通知有人进入房间，主要是确保 2 人都在房间以后，再启动录制，业务不需要录制则不需要关注该回调
        virtual void OnNewJoiner(UserInfo user) {}
    };
```

6.4.6. 错误码

错误码都是通过回调的形式提供的，分别体现在 OnEngineEvent 以及 OnSessionEvent 回调函数里面。

大类	子类（EventType）	错误码类别
引擎相关	INIT	0：引擎初始化成功。 -1：引擎初始化不成功。
	WSS_LINK	0：信令通道建立或重连成功 。 -1：信令通道连接失败。
	CREATE_ROOM	0：创建房间成功。 其他：创建房间失败。
	JOIN_ROOM	0：加入房间成功。 其他：加入房间失败。

大类	子类（EventType）	错误码类别
会话相关	PUBLISH	0：发布成功。 其他：发布失败。
	SUBSCRIBE	0：订阅成功。 其他：订阅失败。
	ICE_LINK	0：数据通道建立成功。 其他：数据通道建立失败。
	START_RECORD	0：启动录制成功。 其他：启动录制失败。
	EXIT_ROOM	0：退出房间成功。 其他：退出房间失败。

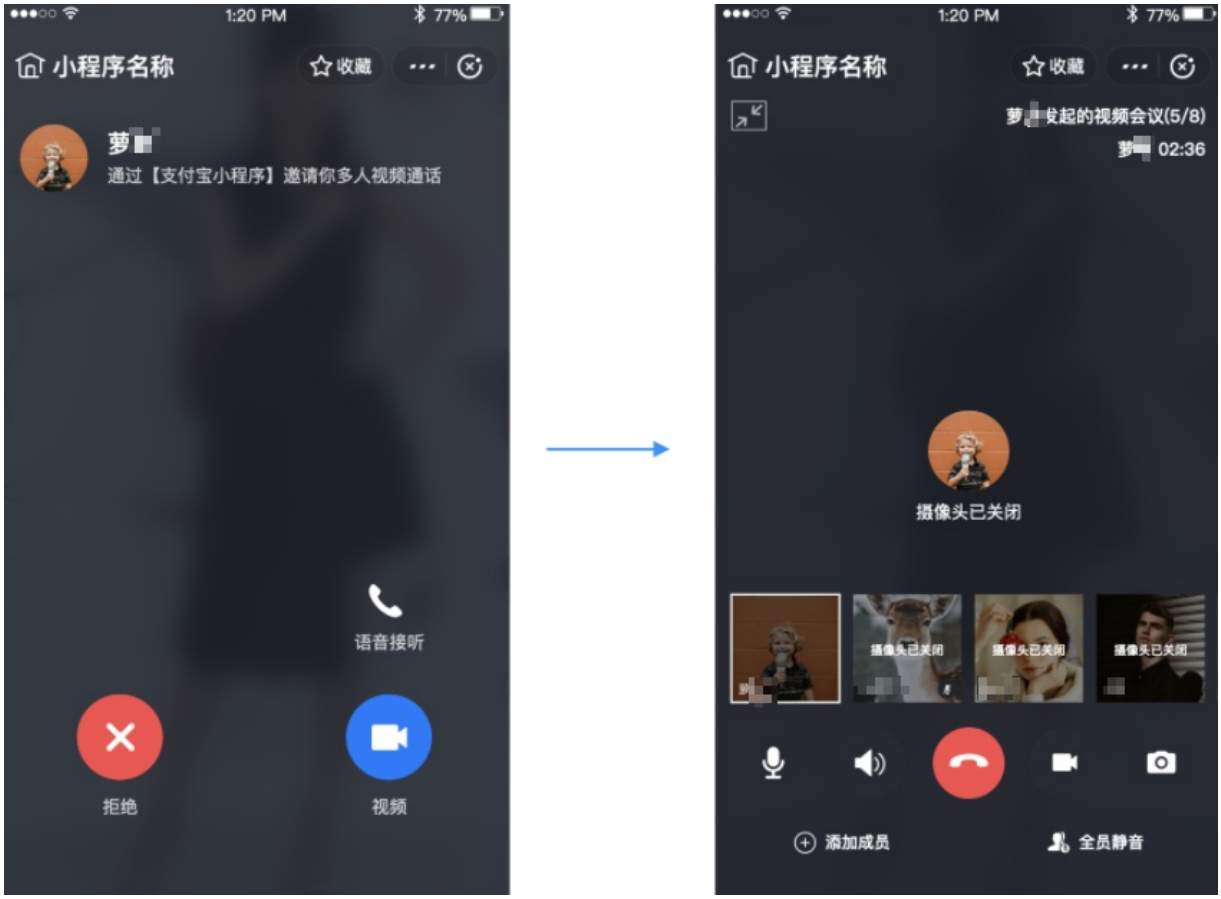
6.5. 支付宝小程序 API

6.5.1. 概述

实时音视频通话插件通过集成阿里云 mPaaS 音视频通话服务，可以在支付宝小程序之间，以及支付宝小程序与其他应用之间实现一对一和多对多的实时音视频通话功能。

案例介绍

实时音视频通话插件可以用于互联网医疗问诊、互联网审案、公益诉讼和实时客服等场景。



6.5.2. 重要参数

本文主要介绍发起视频通话（Start） 和加入视频通话（Join）及组件需要传入的 Config 参数。

基础参数

参数名称	参数类型	是否必填	说明
userId	String	是	用户 ID。
appId	String	是	小程序 ID。
roomId	String	否	房间 ID。
token	String	否	房间 token。
resolution	int	否	视频的清晰度，0(640 * 360), 1(960 * 540), 2(1280 * 720)，默认为 2。
fps	int	否	视频帧率，15 / 30，默认为 30。

streamInfo

使用不同通道时需传入不同的参数，目前只支持 mPaaS 通道，下表列出了在 mPaaS 通道下的相关参数。

通道	参数名称	参数类型	是否必填	说明
mPaaS	signature	String	是	蚂蚁域内和 mPaaS 必选。
	serverUrl	String	是	服务器 URL。 mPaaS 通道下，serverUrl 地址不变：`wss://cn-hangzhou-mrtc.cloud.alipay.com/ws`。
	workspaceId	String	是	使用 mPaaS 时必选。
	bizName	String	是	使用 mPaaS 时必选。
	subBiz	String	是	使用 mPaaS 时必选。

界面显示

参数名称	参数类型	是否必填	说明
displayInfo	String	否	界面显示用到的配置，可选。
calling_information	String	否	接听界面显示的文本。
end_information	String	否	挂断界面显示的文本。
pusher_visible	boolean	否	是否显示（配合 `enableCamera` 用于只想推音频的场景）。
pusher_isMini	boolean	否	是否以小窗显示（默认 false，与 `player.isMini` 互斥，只能一个为 true）。
player_visible	boolean	否	是否显示。
player_isMini	boolean	否	是否以小窗显示（默认 true，注意与 `pusher.isMini` 互斥，只能一个为 true）。

参数名称	参数类型	是否必填	说明
hangup_visible	boolean	否	是否显示关闭按钮（默认 true，只有值为 false 时隐藏）。
mute_visible	boolean	否	是否显示静音按钮（默认 true，只有值为 false 时隐藏）。
screenshot_visible	boolean	否	是否显示截屏按钮（默认 true，只有值为 false 时隐藏）。
enableCamera_visible	boolean	否	是否显示开启摄像头按钮（默认 true，只有值为 false 时隐藏）。
swtichCamera_visible	boolean	否	是否显示切换摄像头按钮（默认 true，只有值为 false 时隐藏）。
users_visible	boolean	否	是否显示用户列表（默认 true，只有值为 false 时隐藏）。
users_name	string	是	界面上显示的用户名称（比外层的 users 优先级高）。
users_avatar	string	是	界面上显示的用户头像（比外层的 users 优先级高）。
users_name	string	否	界面上显示的用户名称。
users_avatar	string	否	界面上显示的用户头像。
controlInfo_enableCamera	boolean	否	启动时是否打开摄像头。

Config 参数

```
{
  userId,   // 用户 ID
  appId,    // 小程序 ID

  roomId?,  // 房间 ID，创建不需要，加入可选
  token?,   // 房间 token，创建不需要，加入必选
```

```
  resolution?: 2, // 视频的清晰度，0(640 * 360), 1(960 * 540), 2(1280 * 720)，默认为 2
  fps?: 30,       // 视频帧率，15 / 30，默认为 30

  streamInfo: {
    // 根据不同通道传入不同的参数，目前只支持 mPaaS 通道
    //  - mPaaS
    //    - signature
    //    - serverUrl (wss://cn-hangzhou-mrtc.cloud.alipay.com/ws)
    //    - workspaceId
    //    - bizName
    //    - subBiz

    // 签名
    signature,            // 蚂蚁域内和 mPaaS 必选

    // mPaaS
    bizName: '',          // 使用 mPaaS 时必选
    subBiz: '',           // 使用 mPaaS 时必选
    workspaceId: '',      // 使用 mPaaS 时必选
    serverUrl: 'wss://cn-hangzhou-mrtc.cloud.alipay.com/ws'
    // mPaaS 通道下，上面这个 serverUrl 地址不变

  },

  displayInfo?: { // 界面显示用到的配置，可选
    calling?: {
      information?: string // 接听界面显示的文本
    },

    end?: {
      information?: string // 挂断界面显示的文本
    },

    answer?: {
      pusher?: {
        visible?: boolean  // 是否显示（配合 enableCamera 用于只想推音频的场景）
        isMini?: boolean   // 是否以小窗显示（默认 false，与 player.isMini 互斥，只能一个为 tr
ue）
      },

      player?: {
        visible?: boolean  // 是否显示
        isMini?: boolean   // 是否以小窗显示（默认 true，注意与 pusher.isMini 互斥，只能一个为
true）
      },

      actions?: {
        hangup?: {
          visible?: boolean // 是否显示关闭按钮（默认 true，只有值为 false 时隐藏）
        },

        mute?: {
          // ! 表示的是采集端的声音采集，不是手机的音量
          visible?: boolean // 是否显示静音按钮（默认 true，只有值为 false 时隐藏）
```

```
          visible?: boolean // 是否显示静音按钮（默认 true，只有值为 false 时隐藏）
        },

        screenshot?: {
          visible?: boolean // 是否显示截屏按钮（默认 true，只有值为 false 时隐藏）
        },

        enableCamera?: {
          visible?: boolean // 是否显示开启摄像头按钮（默认 true，只有值为 false 时隐藏）
        },

        swtichCamera?: {
          visible?: boolean // 是否显示切换摄像头按钮（默认 true，只有值为 false 时隐藏）
        }
      },

      users?: {
        visible?: boolean, // 是否显示用户列表（默认 true，只有值为 false 时隐藏）

        // 用户配置（比外层的 users 优先级高）
        config: [userId]: {
          name: string   // 界面上显示的用户名称
          avatar: stirng // 界面上显示的用户头像
        }
      }
    },

    users?: {
      [userId]: {
        name: string   // 界面上显示的用户名称
        avatar: stirng // 界面上显示的用户头像
      }
    }
  },

  controlInfo?: {
    enableCamera?: boolean  // 启动时是否打开摄像头
  },

  extraInfo?: {}
}
```

6.5.3. 状态码和错误码

本文提供音视频通话状态、网络状态和接入操作错误所返回的信息。

通话状态码

状态码	说明
1101	打开摄像头成功

状态码	说明
1102	打开麦克风成功
1003	直播已开始
1004	直播已暂停
1005	直播已停止

网络状态码

状态码	说明
1000	RTMP 推流拥塞
1001	RTMP 推流拥塞结束
1100	当前非 Wi-Fi 状态
1101	切换到 Wi-Fi
1102	当前无网络

错误码

错误码	说明
-1300	用户禁用相机
-1301	用户禁用麦克风
-1302	相机打开失败
-1303	麦克风打开失败
-1500	网络出错

错误码	说明
-1600	传入的参数错误
-1000	未定义内部错误

6.6. 微信小程序 API

6.6.1. 概述

实时音视频通话 WX-SDK 通过集成阿里云 mPaaS 音视频通话服务，可以在微信小程序之间，以及微信小程序与其他移动端或者 Web 端之间实现一对一和多对多的实时音视频通话功能。

主要功能

WX-SDK 主要提供微信小程序端和其他移动端以及 Web 端进行音视频通话的能力。

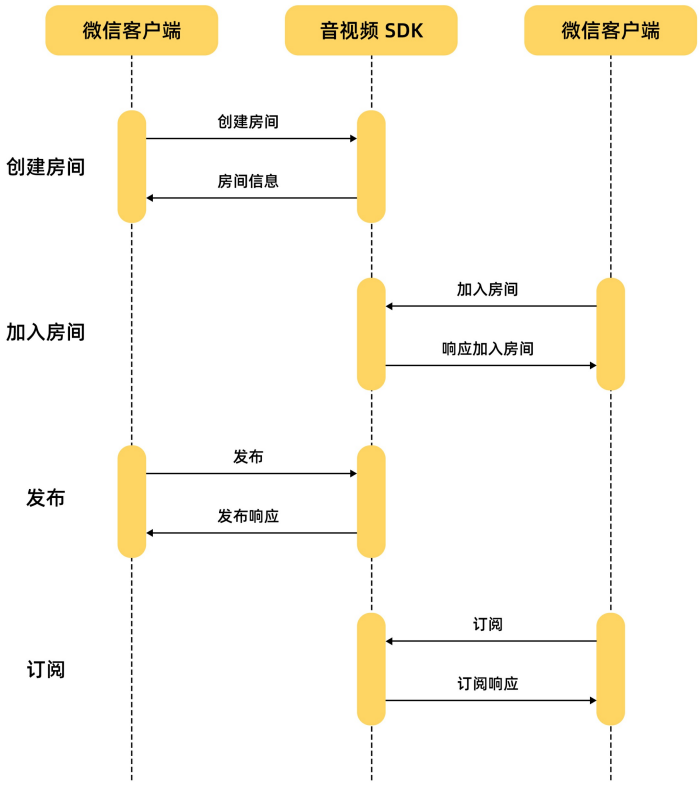
> **说明**
> 当前 SDK 的版本号为 1.1.0。

接口协议

整个 API 封装在 ArtvcRoom 的类中，接口分为两类：

- 主调接口
  - 供业务主动调用
- 回调接口
  - 主调接口的反馈回调
  - 返回业务上的通知

使用流程图



6.6.2. 主调接口

主调接口是供业务主动调用的接口，封装于 ArtvcRoom 类中。

> **重要**
> 目前 uid 只支持使用英文字母、数字、下划线的组合，且长度不超过 128 个字符。

Connect

建立通话连接。

- 参数：

请求参数	值	类型	说明
	uid	String	用户 ID 标识
	biz_name	String	业务标识
	appId	String	业务标识
config（JSON）			

请求参数	值	类型	说明
	workspaceId	String	业务标识
	server_url	String	房间服务器的地址

- **回调接口异步返回：**
  - OnConnect(data)
  - OnError(data)

CreateRoom

创建通话房间。

- **参数：**

请求参数	值	类型	说明
config（JSON）	roomId	String	房间号
	rtoken	String	房间密码

- **回调接口异步返回：**
  - OnCreateRoom(data)
  - OnError(data)

JoinRoom

加入通话房间。

- **参数：**

请求参数	值	类型	说明
config（JSON）	roomId	String	房间号
	rtoken	String	房间密码

- **回调接口异步返回：**
  - OnJoinRoom(data)
  - OnError(data)

Publish

推送通话信息。

- **参数：**

请求参数	值	类型	说明
config（JSON）	resolution	String	分辨率
	fps	int	帧率
	maxBitrate	int	最大码率
	tag	String	自定义标识

- 回调接口异步返回：
  - OnPublish(data)
  - OnError(data)

Subscribe

订阅通话信息。

- 参数：无。
- 回调接口异步返回：
  - OnSubscribe(data)
  - OnError(data)

LeaveRoom

退出通话房间

- 参数：无。
- 回调接口异步返回：
  - OnLeaveRoom(data)
  - OnError(data)

Disconnect

断开通话连接。

参数：无。

SendTxtMessage

发送文本信息。

参数：

请求参数	值	类型	说明
	uids	list	接收信息的所有用户 ID
	timestamp	number	时间戳

config（JSON） 请求参数	值	类型	说明
	msg	String	文本信息的内容

ReportClientEvents

> 说明
>
> 此接口的相关逻辑已在 demo 中实现，由于事件是跟 UI 相关，逻辑是写在业务 UI 层的，因此需要业务侧把该上报逻辑复制到自己的项目中去，方便日常排查及功能扩展。

上报通话事件相关状态码。请求参数（code），类型（int）。

参数：

请求参数	类型	说明
code	int	事件类型： 301：{ desc:"ACTIVITY_PAUSE" },// 小程序后台运行 302：{ desc:"ACTIVITY_RESUME" },// 小程序恢复前台 321：{ desc:"DISABLE_LOCAL_VIDEO" },// 关闭摄像头 322：{ desc:"ENABLE_LOCAL_VIDEO" },// 开启摄像头 323：{ desc:"DISABLE_LOCAL_AUDIO" },// 关闭麦克风 324：{ desc:"ENABLE_LOCAL_AUDIO" },// 开启麦克风 329：{ desc:"CHANGE_TO_EARPIECE" },// 转听筒 330：{ desc:"CHANGE_TO_SPEAKER_PHONE" },// 转外放 363：{ desc:"ERROR_CAMERA_PERMISSION" },// 相机权限错误 364：{ desc:"ERROR_MIC_PERMISSION" },// mic 权限错误 366：{ desc:"ERROR_OPEN_CAMERA" },// 打开摄像头错误-1007 367：{ desc:"ERROR_OPEN_MIC" },// 打开麦克风错误/麦克风采集异常 503：{ desc:"VIDEO_FIRST_FRAME", extra:true }// 订阅到视频首帧
extra	Object	上述事件类型包含 extra 字段，需要携带额外的信息。

6.6.3. 回调接口

回调接口主要是作为主调接口的回调反馈和返回业务上通知的接口，封装在 ArtvcRoom 的类中。

> 说明
> 目前 uid 只支持英文字母、数字、下划线的组合，且长度不超过 128 个字符。

OnConnect(data)

建立连接回调。

回调参数	类型	说明
data	JSON	• code：状态码 • msg：状态说明

OnGetSign(bizName, appId, uid,workspaceId)

> 说明
> 此接口的值需要返回给 SDK。

获取签名。

回调参数	类型	说明
bizName	String	对应 Connect 的 biz_name
appId	String	对应 Connect 的 appId
uid	String	对应 Connect 的 UID
workspaceId	String	对应 Connect 的 workspaceId

OnError(data)

错误信息回调通知。

回调参数	类型	说明
data	JSON	• code：状态码 • msg：状态说明

OnCreateRoom(data)

创建房间成功回调。

回调参数	类型	说明
data	JSON	• code：状态码 • msg：状态说明

OnJoinRoom(data)

加入房间成功回调。

回调参数	类型	说明
data	JSON	• code：状态码 • msg：状态说明

OnPublish(data)

发布媒体流成功回调。

回调参数	类型	说明
data	JSON	• pushUrl：推流 URL

OnSubscribe(data)

订阅媒体流成功回调。

回调参数	类型	说明
data	JSON	• liveUrl：推流 URL

OnLeaveRoom(data)

离开房间回调。

回调参数	类型	说明

data	JSON	• leaveType
		  ◦ 1：正常退出
		  ◦ 2：异常退出

OnParticipantLeaveRoom(data)

通知对端离开房间回调。

回调参数	类型	说明

data	JSON	• participant：用户 UID
		• exitType
		  ◦ 1：正常退出
		  ◦ 2：异常退出

OnSendTxtMessageSucc(data)

发送文本信息成功回调。

回调参数	类型	说明
msgId	number	已发送文本信息的 ID

OnGetTxtMessage(data)

获取文本信息成功回调。

回调参数	类型	说明
mgsId	int	文本信息的 ID
uid	int	发送文本信息的 UID
msg	String	文本信息的内容

OnGetFeedIdsD(Array)

获取当前房间的所有用户信息成功回调。

回调参数	类型	值	值类型	说明
Array	Array	feedId	String	订阅流需要的 ID
		uid	String	用户 ID
		tag	String	自定义标签

OnClientJoin(Data)

通知有用户加入房间回调。

回调参数	类型	值	值类型	说明
Data	JSON	feedId	String	订阅流需要的 ID
		uid	String	用户 ID
		tag	String	自定义标签

6.6.4. 状态码

本文介绍相关的状态码信息。

状态码（code）	说明（msg）
0	成功
-10411	创建房间超时
-10412	创建房间失败
-10431	连接已关闭
-10432	加入房间超时
-10433	加入房间失败
-10451	系统错误
-10471	系统错误

7.附录

本文介绍录制参数 `recordParam` 中包含的具体配置项。

配置项	类型	说明	默认值
silentRecord	bool	静默录制模式	false
width	int	录制视频的宽度	640
height	int	录制视频的高度	360
recordTotalStream	int	最小成功开启录制流的条数	0
startTimeout	int	录制开启超时时间	10s
splitType	int	额外单独分离音视频： • 0：不分离 • 1：额外单独分离音频 • 2：额外单独分离视频	0
splitFilePath	String	录制音视频分离文件的存储路径	后端□动□成
endType	int	结束类型： • 0：调□者退出时 • 1：房间销毁时	0
layoutParam	Array	布局参数：[tag1，tag2，tag3...]	-

tagPositions	JsonArray	JSON 内容如下：
		• 流 tag，不能为空
		• position
		  ◦ xPosition
		  ◦ yPosition
		  ◦ width
		  ◦ height
			-

配置项	类型	说明	默认值

overlaps	JsonArray	JSON 内容如下：
		• enable：bool 值，true 为开启，false 为关闭
		• type:
		  ◦ 1 为时间戳水印
		  ◦ 2 为文字水印
		  ◦ 3 为图片水印
		• id：水印 ID，不设置为空
		• tag：流 tag，如果不设置或为空，则为全局水印
		• xPosition：x 轴位置
		• yPosition：y 轴位置
		• text: 水印文字
		• fontSize：字体大小
		• url：水印图片 HTTP 地址
			-

tagFilter	String	匹配上发布端的 tag 前缀才能订阅录制	-
crf	int	录制视频的清晰度	-