百度智能云

RTC 文档

【版权声明】

版权所有©百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司。 未经本公司书面许可，任何单位和个人不得擅自摘抄、复制、传播本文档内容，否则本公司有权依法追究法律责任。

【商标声明】

和其他百度系商标，均为百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司的商标。 本文档涉及的第三方商标，依法由相关权利人所有。未经商标权利人书面许可，不得擅自对其商标进行使用、复制、修改、传播等行为。

【免责声明】

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导。 如您购买本文档介绍的产品、服务，您的权利与义务将依据百度智能云产品服务合同条款予以具体约定。本文档内容不作任何明示或暗示的保证。

目录

大模型实时互动

动态与公告

功能发布记录

发布时间	功能分类	功能描述
2025-05-15	新功能	iOS v1.2.8版本发布： • 新增地图导航互动功能 • 新增发送文本到TTS直接播报接口 Websocket API发布： • 支持通过Websocket API方式接入大模型实时互动能力
2025-04-25	新功能	大模型实时互动功能全新上线！ • 新增大模型实时互动产品简介，介绍了核心概念、功能特性、应用场景、产品优势和计费模式。 • 新增大模型实时互动快速入门指导，帮助开发者快速集成。 • 新增大模型实时互动SDK和服务端API，支持Android、iOS、H5、小程序、RTOS等不同客户端的接入。

产品简介

概述

大模型实时互动以多模态互动、超低延时、全链路语音增强和丰富生态为核心，为开发者提供了一站式的大模型智能交互服务，支持语音互动、视频互动、数字人互动、任务交互等场景，可以广泛应用于智能眼镜、AI玩具、智慧屏、教育硬件等各类智能设备，也可以用于AI口语老师、AI律师、心理咨询、差旅报销、旅游规划等各类智能应用场景。

核心概念

大模型实时互动

- 互动应用：大模型实时互动基础业务单元，每个互动应用有唯一的AppID，并且对应一种互动类型，包括语音互动、视频互动、数字人互动，每个互动应用下可以创建多个互动实例。
- ASR：语音转文字，在大模型互动时将用户的语音流转化成文字，然后输出给大模型。
- TTS：文字转语音，在大模型互动时将大模型输出的文字转化成语音，然后进行播放。
- 声音复刻：支持通过30s-1min的音频文件进行快速音色复刻。
- LLM：大语言模型，使用大量文本数据训练的深度学习模型，使得该模型可以生成自然语言文本或理解语言文本的含义。
- 智能体：可以自主思考、决策，并执行复杂任务的Agent。
- function call：通过大模型调用预先定义好的函数，以完成特定任务，如“调大音量”、”拍照“等。
- 话题：在大模型互动中，客户可自定义的大模型，将一类或几类话题的意图，转发到自定义的大模型服务。例如：育儿话题、佛学佛经话题。
- 场景与角色：通过场景及角色功能，您可以设定大模型的作用范围，包括指定大模型扮演的角色、具备的能力、输出结果的格式与风格等，角色如太乙真人、哪吒，场景如成语接龙、猜谜语等，支持设置多个场景与角色。

- 云渲染：云渲染是指将语音、视频、图文、网页等内容在云端渲染处理，渲染完成后将结果传回本地设备显示，适用于低算力设备使用。
- 语音互动：互动应用类型之一，支持用户通过语音通话的方式与大模型进行实时互动。
- 视频互动：互动应用类型之一，支持用户通过音/视频通话的方式与大模型进行实时互动。
- 数字人互动：互动应用类型之一，支持用户通过数字人通话的方式与大模型进行实时互动。
- 大模型互动框架：提供大模型实时互动全链路服务，包括实时音视频、语音增强、语音转文字、大模型、文字转语音服务，其中大模型、文字转语音服务支持客户接入第三方服务。
- License授权：License是大模型互动服务的一种计费模式，在智能硬件场景中，百度通过License对单个设备进行收费，客户购买License后，需要将License烧录在设备中激活SDK使用，在License有效期内，设备使用服务不再进行收费。

功能特性

大模型实时互动功能

产品能力	功能	功能说明	Android	iOS	Web	RTOS	小程序	WebSocket
互动方式	语音对话	用户通过语音通话的方式与大模型互动，大模型理解用户意图，给出语音答复。	✓	✓	✓	✓	✓	✓
	视频理解	用户开启摄像头持续推送视频流与大模型互动，大模型对视觉内容进行理解，输出理解内容，支持视频流问答，以及指导用户完成复杂任务，例如下棋、做饭等。	✓					
	数字人对话	用户与数字人进行互动对话，大模型输出的内容通过数字人的讲话、动作等传递给用户，为用户提供更加真实的互动体验。	✓	✓	✓		✓	
	图片理解	用户通过发送图片、提问自动上传图片方式与大模型互动，大模型采用语音、文本回答。	✓		✓			
ASR	VAD时长	当用户停止说话时，ASR将等待一段时间再将文本内容提供给大模型，确保短暂的停顿不会结束用户讲话，支持自定义VAD时长	✓	✓	✓	✓	✓	✓
	输入语言	支持中英文输入	✓	✓	✓	✓	✓	✓
	语音字幕	将用户的语音转文字后，可以将文字内容实时传给客户端，展示用户说话的语音字幕	✓	✓	✓	✓	✓	✓
大模型	开场白	支持设置开场白，进入通话后会主动与用户打招呼	✓	✓	✓	✓	✓	✓
	角色人设	支持设置大模型角色人设	✓	✓	✓	✓	✓	✓
	用户位置	支持设置用户位置，在沟通语境中会使用设置的位置	✓	✓	✓	✓	✓	✓
	多轮对话	默认记忆15轮历史对话。大模型根据对话内容持续跟进话题，并准确把握对话上下文语境，理解隐含意思，给出恰当回复，顺畅与用户进行多轮交互，保证对话连贯。	✓	✓	✓	✓	✓	✓
	大模型回复字幕	大模型输出文字内容后可以实时传给客户端，展示大模型回复的语音字幕	✓	✓	✓	✓	✓	✓

	暴							
	Function Call	用户与智能体互动的过程中，可以说出特定指令以调用特定功能，比如【拨打电话】、【调大音量】等，支持根据业务需求自定义配置Function Call	✓	✓	✓	✓	✓	✓
	互动指令	已预置一些互动指令，如暂【停止讲话】/【恢复讲话】、【关闭自动打断】/【开发自动打断】、【设置人设】、【设置输出语言】等，提升用户体验	✓	✓	✓	✓	✓	✓
	接入第三方大模型	系统已预置大模型服务，也支持切换第三方大模型服务，包括百度千帆、阿里百炼、通义千问、豆包大模型、腾讯云LKEApp等	✓	✓	✓	✓	✓	✓
	有声资源	支持音乐、故事、相声	✓	✓	✓	✓	✓	✓
	成语接龙	用成语的最后一个字作为下一个成语的第一个字，进行成语接龙的游戏。	✓	✓	✓	✓	✓	✓
	脑筋急转弯	用户出脑筋急转弯让大模型猜，或者大模型出脑筋急转弯用户猜	✓	✓	✓	✓	✓	✓
	猜谜语	支持与大模型一起玩猜谜语	✓	✓	✓	✓	✓	✓
	飞花令	支持与大模型一起玩飞花令，用户与大模型轮流说出含有指定关键字的诗句，且关键字在诗句中的位置逐轮推进，无法及时接出合规诗句就算输	✓	✓	✓	✓	✓	✓
TTS	输出语言	支持中文或英文输出	✓	✓	✓	✓	✓	✓
	音色	支持切换不同的音色	✓	✓	✓	✓	✓	✓
	接入第三方TTS	系统已预置TTS服务，支持切换成第三方TTS服务，包括百度、火山、讯飞	✓	✓	✓	✓	✓	✓
音频增强	智能打断	可以通过语音快速打断语音播报，并对新一轮的问题进行回复，支持关闭	✓	✓	✓	✓	✓	✓
	AI降噪	通过端侧和服务端的AI降噪算法，有效识别并消除常见的尖锐声、键盘声等非人声噪声，有效提升ASR识别准确率，避免噪音误打断	✓	✓	✓	✓	✓	✓
	声音增益	智能识别人声，对人声做自动增益，人声更清晰	✓	✓	✓	✓	✓	✓
	回声消除	有效杜绝回声、啸叫问题	✓	✓	✓	✓	✓	✓
	VAD增强	检测到用户停止说话后，快速将ASR转出的文本发送给大模型，有效降低端到端延时	✓	✓	✓	✓	✓	✓
多平台支持	互动SDK	支持多个端SDK：Andriod、iOS、Web、小程序、Linux、RTOS，其中RTOS已适配：乐鑫、杰理、ASR、移远	✓	✓	✓	✓	✓	✓

性能指标

性能指标	性能说明
端到端语音延时	1.4s
端到端视频延时	2.5s
打断延时	0.8s

应用场景

大模型实时互动

智能眼镜

通过语音指令、手势控制和视觉反馈，满足百科问答、地图导航、实时翻译、图像识别、会议纪要等场景需求，提升智能眼镜的实用性和便捷性。

AI玩具

为玩具提供更智能的交互能力，通过识别儿童的语音、手势等，给予生动有趣的回应，陪伴儿童成长，激发儿童的想象力和创造力。

智慧屏

智慧屏结合数字人形象，用户通过智慧屏与数字人进行实时语音对话，应用于AI律师、AI医生、心理咨询、研学等场景。

智慧应用

赋能智能应用AI大模型实时对话能力，打造个性化人设，模拟真实场景下的多种对话情境，适用于AI口语练习、AI数学解题、知识答疑、财务报销、旅游指导、数字人互动等场景。

产品优势

大模型实时互动

超低延时互动

语音互动端到端响应延时1.4s，语音打断响应延时0.8s以内，超低延时互动，交互更流畅

丰富应用资源

打通百度丰富的生态资源，提供30+应用和资源，如百度百科、音乐、经典故事、有声读物、翻译、导航等，降低开发者对接成本

全链路音频增强

支持AI降噪、声音增益、回声消除、人声分离、声纹识别、VAD检测、智能打断，7大音频增强，云+端协同，有效提升互动体验性

开放式LUI互动框架

核心服务组件（LLM/TTS）可替换，支持Function Call自定义配置，满足满足个性化需求

全平台支持

提供开箱即用的互动SDK，只需要调用几个接口就可以快速集成多模态互动功能，支持Android、iOS、Web、小程序、Linux、RTOS等多个端，RTOS已适配杰理、乐鑫、ASR、移远、BK、RK等多款主流芯片

产品计费

大模型实时互动计费

本文介绍 大模型实时互动 计费规则。

大模型实时互动支持多种计费方式，分为按license、通话时长、通话并发计费，适用于不同的场景。

每种计费方式的具体价格请咨询商务。

按license计费

计费项：license授权数

付费方式：预付费

计费公式：大模型实时互动费用 = license授权数 x 个数

按通话时长计费

计费项：通话时长

付费方式：后付费

付费周期：按小时扣费，即北京时间整点扣费并生成账单。出账单时间是当前计费周期结束后1小时内。例如，10:00-11:00的账单会在12:00之前生成，具体以系统出账时间为准

计费公式：大模型实时互动费用 = 音频通话时长 x 音频单价 + 视频通话时长 x 视频单价

按通话并发计费

计费项：通话并发

付费方式：预付费

计费公式：大模型实时互动费用 = 通话并发 x 单价

快速入门

开通服务

您的应用接入大模型实时互动服务，您必须先开通实时音视频RTC产品服务。

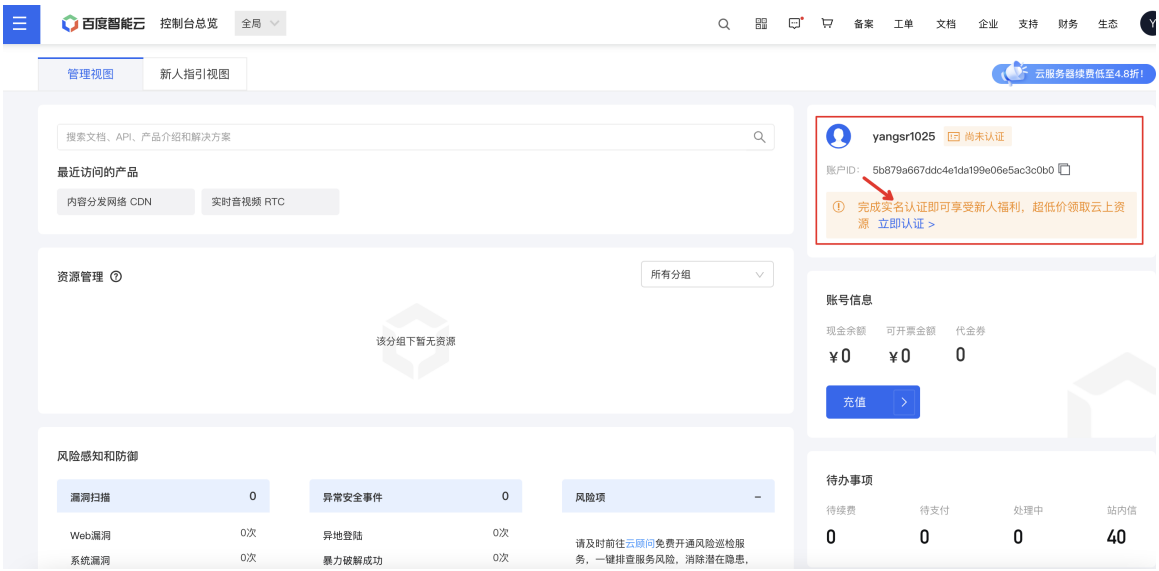
要开通实时音视频RTC产品服务，您需要遵循以下步骤：

1、登录百度智能云控制台

1. 进入百度智能云控制台，如果您是首次登录，请先注册账号，参看账号注册。
2. 如果您已拥有百度智能云账号，请先登录，参看登录。

2、实名认证

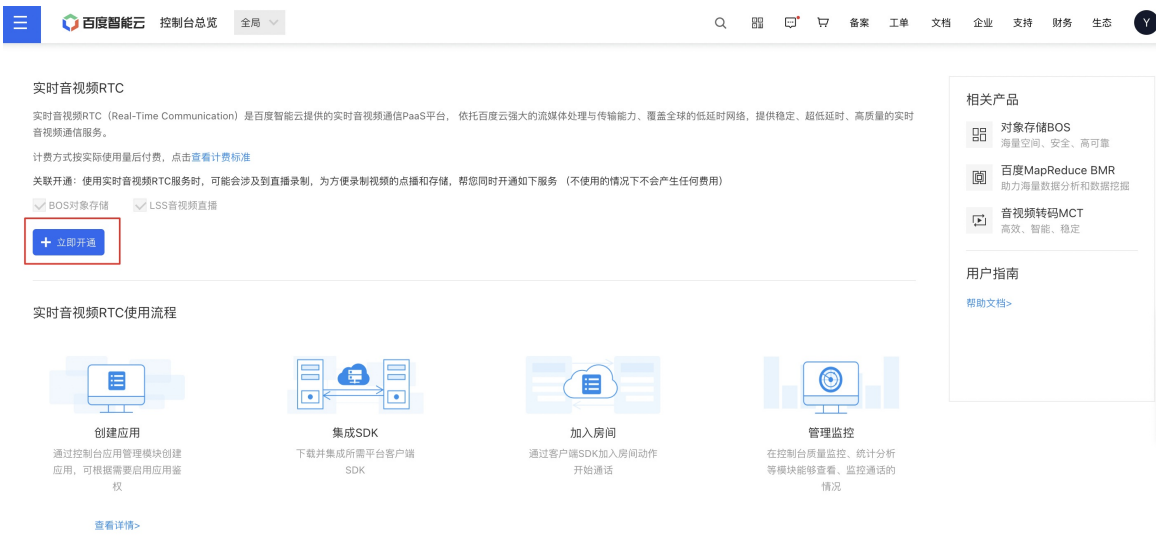
登录成功后，您必须先进行实名认证，参看实名认证：



3、申请开通 RTC 服务

您可以在总览页选择 「**实时音视频 RTC**」 进入并**申请开通**。

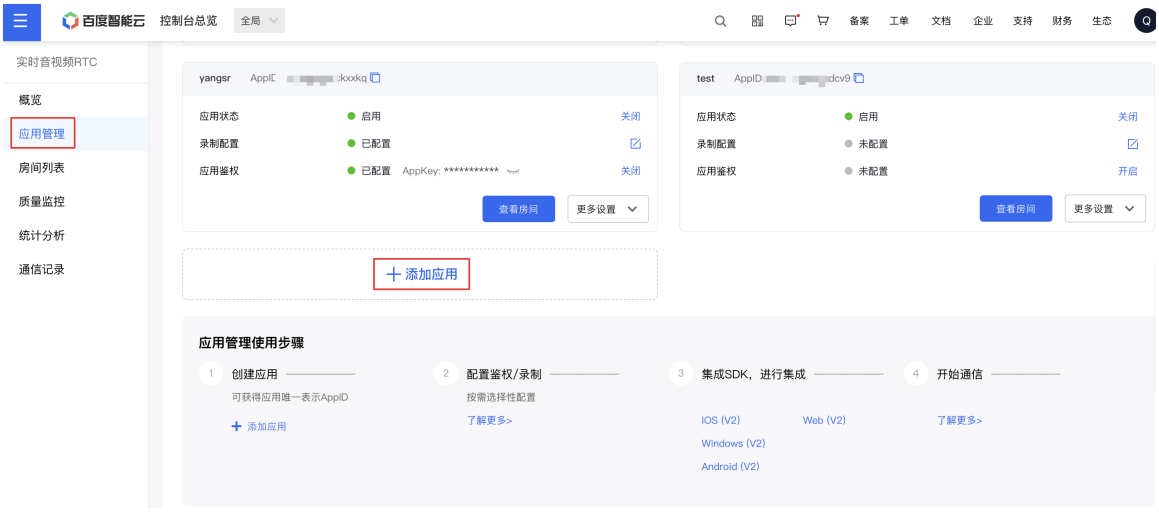
也可以通过 **实时音视频 RTC** 产品详情页 点击 **立即使用** 申请开通。



4、创建 RTC 应用，获取 AppId

进入 RTC 产品控制台后，您可以在 「**应用管理**」 中管理您的应用（包括修改应用名称、应用状态、查看 AppID 、 AppKey 等）。

如果您需要创建新的应用，可以点击**添加应用**。



注释

1. AppId 是每个应用的唯一标识符，在调用 RTC SDK 的 API 接口实现功能时，您必须填入您获取到的 AppId 。
2. AppKey 是每个应用对应的密钥，用于生成 Token 鉴权，请妥善保管。

接口鉴权

服务地址

BRTC服务端 API 通过域名接入，服务域名为 **rtc-aiagent.baidubce.com**。

通讯协议

BRTC 服务端 API 支持 HTTP 和 HTTPS 两种协议。为了提升数据的安全性，建议使用HTTPS协议。

数据格式

数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

API认证机制

API 认证机制的详细内容请参见鉴权认证。
如何获取 AK/SK 参见获取AK/SK。
使用常用工具快速生成签名请求，可以选择在线签名工具。
API 鉴权示例代码参见Sample-Code。
详细的签名报错及处理方式请参考常见签名认证错误排查。

公共请求头与响应头参数，如非必要，在每个接口单独的接口文档中不再进行说明，但每次请求均需要携带这些参数。

公共请求头

头域	说明	是否必须
host	服务域名，目前固定为rtc-aiagent.baidubce.com	是
authorization	包含Access Key与请求签名。具体请参考鉴权认证	是
content-Type	application/json	是
x-bce-request-id	请求ID	否
x-bce-date	表示日期的字符串，符合API规范	否

公共响应头

头域	说明
content-type	application/json;charset=UTF-8
x-bce-request-id	对应请求ID

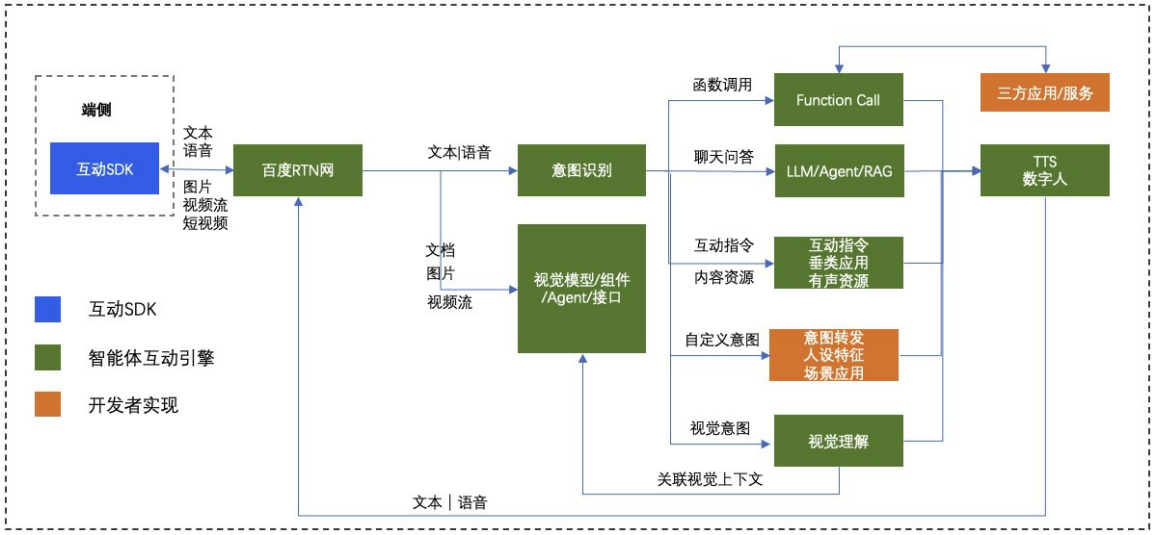
集成步骤指引

音视频互动集成步骤指引

文档作用

本文档目的在介绍如何快速了解和集成多模态大模型互动，以及如何定制功能。实现语音互动、视频互动、数字人互动、第三方模型互动。

架构介绍



上图为音视频互动整体架构，主要由端 + 云服务组成。端侧提供互动SDK供应用集成，云上提供大模型互动能力供SDK在线调用。主要如下：

- 大模型互动SDK
  - 采集或者数据输入各类媒体：文本、音频流、视频流、图片文件等，然后采用RTC协议将数据低延迟传送到云端大模型互动服务。
  - 采用RTC协议低延迟接收到云服务返送的文本、音频流、视频流、卡片多媒体内容，播放或接口输出到端应用。
  - 音频支持SDK直采麦克风设备，或者数据接口接受APP的数据输入两种方式。支持音频PCM、G711、G722、Opus格式，采样率8K、16K、48K。
  - 支持硬件或者软件的音频回声消除、降噪、增益功能，与云上音频增强处理相配合。
  - 支持图片、视频流的自动降采样处理，压缩数据尺寸；支持JPEG、H264、H265编码格式。
  - 支持RTC超低延迟传输、抗弱网、就近调度功能，支持NACK、FEC、ARQ、网络自适应功能。
  - SDK提供大模型互动语意的专用接口，简单，快捷集成。
  - 提供**智能打断**、**实时双工**、**VAD检测**、**低延迟交互**、播放有声资源功能。

大模型互动云服务

云服务主要包括3个阶段：音视频前处理、模型服务、音视频后处理。音视频前处理是在接收到原始音视频后，进行语音、视频增强，按需ASR语音转文本服务；模型服务区分系统内置的多模态交互大模型服务和第三方模型对接；音视频后处理是处理模型返回的文本或语音，驱动TTS或数字人生成音视频，发送给SDK。

- 音视频前处理
  - 提供云端音频增强功能，包括：**大模型AI降噪**、**人声分离**、**声纹识别**、**VAD增强**。
  - 提供云端视频增强功能，包括：视频抽帧、图像检测、图像增强、视频分段等。
  - 提供ASR语音转写、情绪识别、多国语言识别、方言识别。
- 模型服务

  用户可以使用系统内置的多模态互动大模型服务，直接调用专为实时交互构建的一揽子方案，包括工具应用、内容资源、视觉多模态等功能。用户也可以只使用大模型互动框架，对接第三方模型或智能体，不使用系统内置模型功能，这种方式获得最大灵活性，同时也获得互动框架带来的智能打断、语音双工、低延迟互动、音视频增强等能力。

多模态互动大模型服务 （内置）

- 意图识别：提供多轮的用户意图识别能力，将用户意图分类为：函数调用、知识问答、闲聊陪伴、数学计算、互动指令、垂类应用、视觉理解等十余种，并支持意图可配置、意图可定制能力。
- 聊天问答：支持闲聊闲谈、知识问答、联网实搜、数学计算等功能。
- 语音指令：暂停聊天、继续聊天、开启或关闭自动打断、设定人设、设定输出语言等。
- 垂类应用，例如天气、限号、百科、彩票等30多种。
- 可选服务：地图导航、附近推荐、电话呼叫、有声资源播放、实时翻译等。
- 视觉理解：支持上传图片、视频流、问题触发的视觉理解服务，能够对动物、植物、场景、路标等图像问答。
- 定制Function Calling：可定制函数意图、函数参数、执行结果TTS播报内容；函数触发事件可发送到SDK或者在云端调用HTTP接口执行。
- 定制话题：支持将特定话题意图识别出来，转发到某智能体提供服务，例如把佛学宗教、育儿教育等相关话题，转发到第三方智能体或模型。
- 定制人设：支持设置复杂人设，定制人设prompt；系统也内置了若干人设供选择，可通过API或语音指令切换。
- 定制场景：允许用户定制场景化prompt。例如针对成语接龙游戏定制prompt，针对模拟面试定制prompt等；可通过API、语音指令切换不同的场景。

对接第三方大模型服务

系统架构支持良好的开发性，允许使用第三方模型、智能体平台服务。详细参见对接第三方模型说明（链接）

- 大模型支持除百度文心系列之外，已支持了阿里千问、豆包、OpenAI大模型，以及支持符合OpenAI LLM协议的任意三方大模型（协议规范链接）。
- 多模态模型对接：支持视觉模型包括：文心多模态模型、豆包多模态、OpenAI gpt-4o、千问视觉模型等。
- 智能体平台已对接：百度AppBuilder平台、阿里百练平台、Coze平台、腾讯混元平台、Dify。

- 音视频后处理
  - 支持TTS文本转语音功能，提供丰富的发音人、音色供选择，也支持一句话声音克隆，支持多国语言。
  - TTS服务支持多厂商对接：包括百度云TTS、讯飞云TTS、火山云TTS等。
  - 支持数字人对接，当前对接了百度曦灵数字人系统。（链接） **注意：**该场景要求采用第三方模型方式集成。

注意：上述功能，可能受不同平台端SDK不同而不同。服务区分国内、海外版，国内与海外服务不互通。

集成步骤

- 1.在百度云上注册账号后，开通RTC多模态实时交互服务(开通链接 介绍链接)。
- 2.在RTC多模态实时交互服务中，创建APP应用，获得APPID后，可通过创建工单、邮件、服务控制台（即将上线）三种方式开通大模型互动服务，服务开通后账号内有免费额度供试用，免费用完或到期需购买服务授权。如果采用工单、邮件方式开通服务，申请中需携带：用户账号、APPID、业务概述、服务区域（国内、欧美、东南亚等）、服务内容（第3步的服务内容项）、联系方式内容，申请提交后等待开通。邮件地址：liuzhen29@baidu.com,youli01@baidu.com,keyugang@baidu.com 。
- 3.大模型互动服务开通后，可在服务控制台（即将上线）、邮件、创建工单3种方式配置或者修改服务内容。邮件地址：liuzhen29@baidu.com,youli01@baidu.com,keyugang@baidu.com 。 服务内容配置项主要包括：
  - 互动类型：音频互动、视频互动、数字人互动（链接）、第三方模型互动(链接)
  - ASR参数：VAD时长（默认：1500ms）

- LLM参数：FunctionCalling、话题、人设、场景等
- TTS参数：发音人、倍速等
- 可选功能：**有声读物（音乐、故事、相声等）**、**地图导航（即将上线）**、同声翻译、**电话呼叫**等。注意：标粗为单独收费项

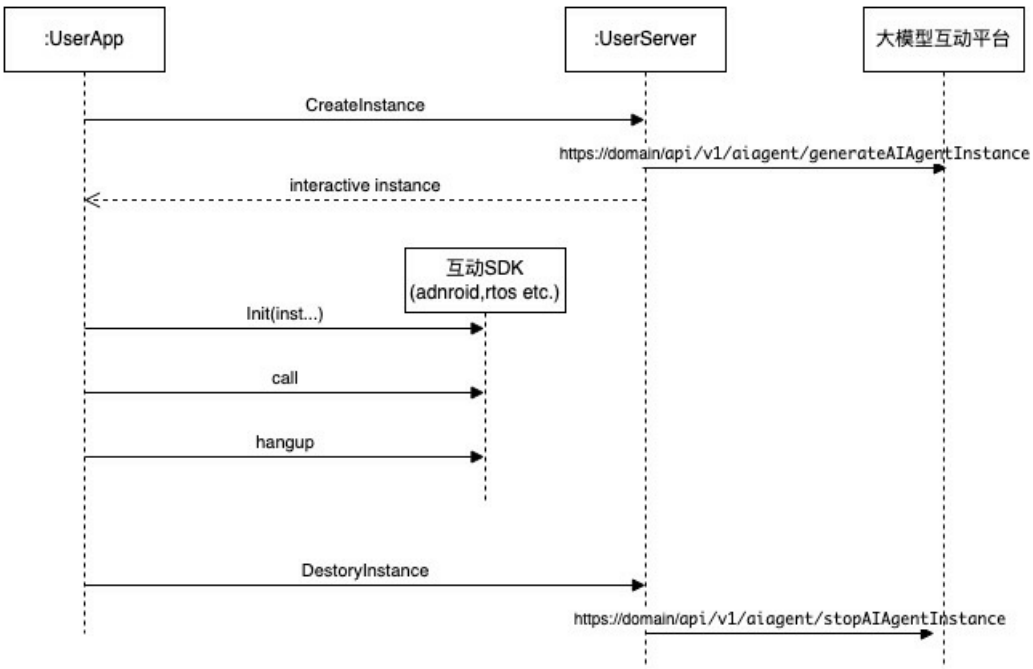
- 4.集成端SDK或者WebsocketAPI开始使用大模型互动服务。 SDK类型包括有：
  - Android SDK 集成指南（链接） SDK说明（链接） SDK下载（链接）
  - IOS SDK 集成指南（链接） SDK说明（链接） SDK下载（链接）
  - H5 SDK 集成指南（链接） SDK说明（链接） SDK下载（链接）
  - 微信小程序SDK 集成指南（链接） SDK说明（链接） SDK下载（链接）
  - RTOS SDK 集成指南（链接） SDK说明（链接） SDK下载（链接）
  - Websocket API 集成指南（链接） SDK下载（链接）
  - Linux SDK (即将上线)

  注意：为了配合端SDK使用，需要先调用云上generateAIAgentCall或generateAIAgentInstance（即将上线）创建互动实例接口（链接）,获得互动服务的Token和服务上下文。该接口说明如下：
  - 该接口作用是在云上创建互动服务实例，加载大模型互动服务配置，返回token给端，供端SDK合法连接到云服务。
  - 云API接口访问需采用百度云统一的AK/SK鉴权方式（链接）。
  - 可通过该接口配置参数，如：LLM参数、TTS发音人、动态访问参数等。

  注意：在SDK断开云端服务时，需主动调用stopAIAgentInstance销毁互动实例接口，否则可能导致云端资源泄漏，造成多计费。

- 5.SDK集成完成，使用账号中的免费额度测试业务。免费额度用完或者业务正式上线，可通过服务控制台（即将上线）、邮件、创建工单3种方式购买服务授权。邮件地址：liuzhen29@baidu.com,youli01@baidu.com,keyugang@baidu.com 。

接口调用时序



- UserAPP是在端上集成大模型互动SDK的应用，例如Android APP、Rtos系统的任务。
- UserServer是应用的服务端，由于访问百度云API需要AK/SK，存放在端侧存在安全隐患，推荐在服务侧调用ggenerateAIAgentCall或generateAIAgentInstance（即将上线）创建互动实例接口（链接），接口返回token和实例上下文返送到端侧，用于SDK初始化。
- UserAPP调用SDK的init初始化后，再call方法用于与互动服务实例建立连接，成功后可以大模型实时互动。
- 互动结束，UserAPP调用SDK的hangup方法断开与云服务连接。UserAPP通知UserServer，调用stopAIAgentInstance方法销毁互动实例，释放云资源。

  注意：上述流程中提到的接口在不同端SDK可能存在差异，以端SDK接口为准。

第三方服务集成步骤指引

第三方服务集成

第三方LLM及智能体集成配置 LLM 支持内置和第三方大模型，同时支持百度千帆智能体、阿里百炼应用、 腾讯混元智能体、扣子等。

默认设置
llm、llm_url不传则使用服务端默认LLM配置

接入第三方LLM服务配置：
一. 使用兼容OPENAI 协议的接入：
{"llm":"OPENAI", "llm_url":"" , "llm_cfg":"$MODEL", "llm_token": "$API-KEY"}

llm_url填入模型服务地址; llm_cfg填入模型名称， llm_token填入API-KEY
1.）百度千帆模型服务地址： https://qianfan.baidubce.com/v2/chat/completions
2.）阿里百炼模型服务地址： https://dashscope.aliyuncs.com/compatible-mode/v1/chat/completions
3.）腾讯混元模型服务地址： https://api.hunyuan.cloud.tencent.com/v1/chat/completions
4.）豆包模型服务地址： https://ark.cn-beijing.volces.com/api/v3/chat/completions
5.）豆包自定义应用地址： https://ark.cn-beijing.volces.com/api/v3/bots/chat/completions
6.）DeepSeek模型服务地址：https://api.deepseek.com/chat/completions
7.）使用客户自己的OPENAI 兼容的服务器接入： 主要配置类别OPENAI， 访问的模型地址， 访问的密钥， 以及具体的模型名称。

二. 百度AppBuilder接入：
{"llm":"LLMAppBuilder", "llm_cfg":"app_id[,conversation_id]", "llm_token": "$API-KEY"}

三. 阿里百炼应用接入：
{"llm":"LLMDashScopeApp","llm_url":"https://dashscope.aliyuncs.com/api/v1/apps/$YOUR_APPID/completion" , "llm_cfg":"[session_id]", "llm_token": "$API-KEY"}

四. 腾讯元宝智能体接入：
{"llm":"LLMYuanbao", "llm_cfg":"assistant_id[,user_id]", "llm_token": "$API-KEY"}

五. 扣子接入：
{"llm":"LLMCoze", "llm_cfg":"bot_id[,user_id][,conversation_id]", "llm_token": "$API-KEY"}

六. Dify接入
{"llm":"LLMDify", "llm_url":"https://api.dify.ai/v1/chat-messages", "llm_cfg":"[user],[conversation_id]", "llm_token": "$API-KEY"}

TTS 设置 服务端API的TTS设置

数字人互动集成步骤指引

构建数字人互动实例

前置条件：

1. 在曦灵管理平台创建数字人应用， 购买云渲染交互组件, 并绑定云渲染交互组件。

https://xiling.cloud.baidu.com/open/widgetStore/list

2. 获得曦灵数字人的AppID 和AppKey后使用如下的 Token 生成工具生成Token（根据业务需要选择token有效期，建议为900000小时）：

https://open.xiling.baidu.com/token-gen-tool.html

配置数字人示例

数字人互动实例是一类特别互动实例。

音频互动示例配置请参考“互动智能体指引”一节。

改变cfg和必要的参数，就可以配置数字人互动实例了， 具体说明如下：

```
 // H5 上创建数字人互动示例代码
var cfg = {
        xiling_auth: 'xiling-token-XXX',
        xiling_bgimg: '',
        fid: 'A2A_V2-muqing', // 数字人形象， 可选形象 https://cloud.baidu.com/doc/AI_DH/s/2lyzilgsg
        tts_url：'s?per=5132', // 数字人音色， 可选音色 https://cloud.baidu.com/doc/AI_DH/s/Slywt3fxy
        tts: 'DHV2', // 固定值，指定数字人服务类别为DHV2
    };
    Agent.Start({
        instance_type: 'DigitalHuman', // 固定值，指明是数字人智能体
        appid: 'AppIdxxx',
        cfg: cfg,
        remotevideoviewid: 'therevideo',
        localvideoviewid: 'herevideo',
        success: function () {
        },
        error: function (error) {
        },
        onmessage: function (msg) {
            console.log('onmessage id: ' + msg.id + ' data: ' + msg.data);
        }
    });
```

参数详解

参数	类型	描述	默认值
instance_type	string	互动智能体实例类别	固定值'DigitalHuman' 为数字人智能体
appid	string	百度 派发的数字 ID, 开发者的唯一标识	
cfg.xiling_auth	string	曦灵数字人的token字符串	
cfg.xiling_bgimg	string	数字人的背景图	
cfg.fid	string	数字人的形象id	可选形象 https://cloud.baidu.com/doc/AI_DH/s/2lyzilgsg
cfg.tts_url	string	数字人音色配置， 格式是：'s?per=5132'	可选音色 https://cloud.baidu.com/doc/AI_DH/s/Slywt3fxy
cfg.tts	string	固定只配置为数字人用	强制设置为： 'DHV2'
error	function(error)	Start()失败，或运行过程中出现了错误	
destroyed	function(error)	运行过程中出现错误被销毁的回调	

快速集成大模型实时互动Demo

Android

Android快速跑通大模型互动DEMO

本文介绍如何通过少量代码集成百度 大模型互动SDK ，快速实现高质量、低延时的大模型互动功能。

实现大模型互动的步骤如下：

- 调用创建互动实例接口（为了快速跑通本DEMO，该接口在Android调用，正式使用必须在服务器上调用）
- 创建并且初始化 `AIAgentEngine#init`
- 开始通话 `AIAgentEngine#call`
- 结束通话 `AIAgentEngine#hangup`
- 销毁资源 `AIAgentEngine#destroy`
- 调用销毁互动实例接口

步骤

1.前期准备 在实现 Android 大模型互动demo之前，你需要有以下准备：

Android Studio 本文档基于2022.1.1 RC 1版本 Android SDK gradle

本文档基于 gradle版本： `com.android.tools.build:gradle:7.1.3`
https://services.gradle.org/distributions/gradle-7.2-all.zip

申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html 下载SDK
https://cloud.baidu.com/doc/RTC/s/wm8y592lc
一台运行 Android 4.4 或以上版本的移动设备。 可以访问互联网的计算机。确保你的网络环境没有部署防火墙，否则无法正常使用百度服务。
快速Demo 下载地址：https://brtc-sdk.cdn.bcebos.com/ai/agent/android/Android.ChatAgent.v1.0.26.zip

2.创建项目

可以使用官网下载的Demo进行参考，也可自行创建项目导入SDK。下面是快速实现大模型互动能力的几个步骤：

（1）创建Android项目 按照以下步骤准备开发环境：如需创建新项目，在 Android Studio 里，依次选择 Phone and Tablet > Empty Activity，创建 Android 项目。 创建项目后，Android Studio 会自动开始同步 gradle, 稍等片刻至同步成功后再进行下一步操作。

(2)导入SDK

maven 依赖

```
implementation 'com.baidubce.mediasdk:lib_agent:1.0.1.x'
implementation 'com.baidubce.mediasdk:brtc:3.4.0.xx'
```

(3)使用SDK

- 初始化SDK

```
AIAgentEngine.AIAgentEngineParams mAiAgentParams;
AIAgentEngine mAiAgent;

mAiAgentParams = new AIAgentEngine.AIAgentEngineParams();
mAiAgentParams.appId = mClientData.mAppId;
mAiAgentParams.context = mClientData.mContext;
/** 调试日志 */
mAiAgentParams.verbose = true;
/** 更改音频编码器 */
if (!TextUtils.isEmpty(mClientData.mAudioCodec)) {
    mAiAgentParams.audioCodec = mClientData.mAudioCodec;
}
//      /** 外部音频采集 */
//      mAiAgentParams.enableExternalAudioInput = true;
//      /** 外部音频渲染 */
//      mAiAgentParams.enableExternalAudioOutput = true;

// 初始化
mAiAgent = AIAgentEngine.init(mActivity, mAiAgentParams);
```

- 开始通话

```
mAiAgent.call(mClientData.mTokenStr, JavaTypesHelper.toLong(mClientData.mInstanceId, 1));


/** 用户端音频ASR结果 */
@Override
public void onUserAsrSubtitle(String text, boolean isFinal) {
    Log.d(TAG, " onUserAsrSubtitle text " + text + " isFinal: " + isFinal);
}

/** AI智能体结果 */
@Override
public void onAIAgentSubtitle(String text, boolean isFinal) {
    Log.d(TAG, " onAIAgentSubtitle text " + text + " isFinal: " + isFinal);
}

@Override
public void onFunctionCall(String id, String funcCallName, String params) {

}
```

- 结束通话&销毁资源

```
    if (mAiAgent != null) {
        mAiAgent.hangup();
        mAiAgent.destroy();
    }
```

iOS

说明

- Xcode 13.0 为例，如何快速接入跑通IOS大模型互动DEMO **准备环境**
- Xcode 9.0 或以上版本
- 支持 iOS 9.0 或以上版本的 iOS 设备
- 支持音视频功能的模拟器或真机

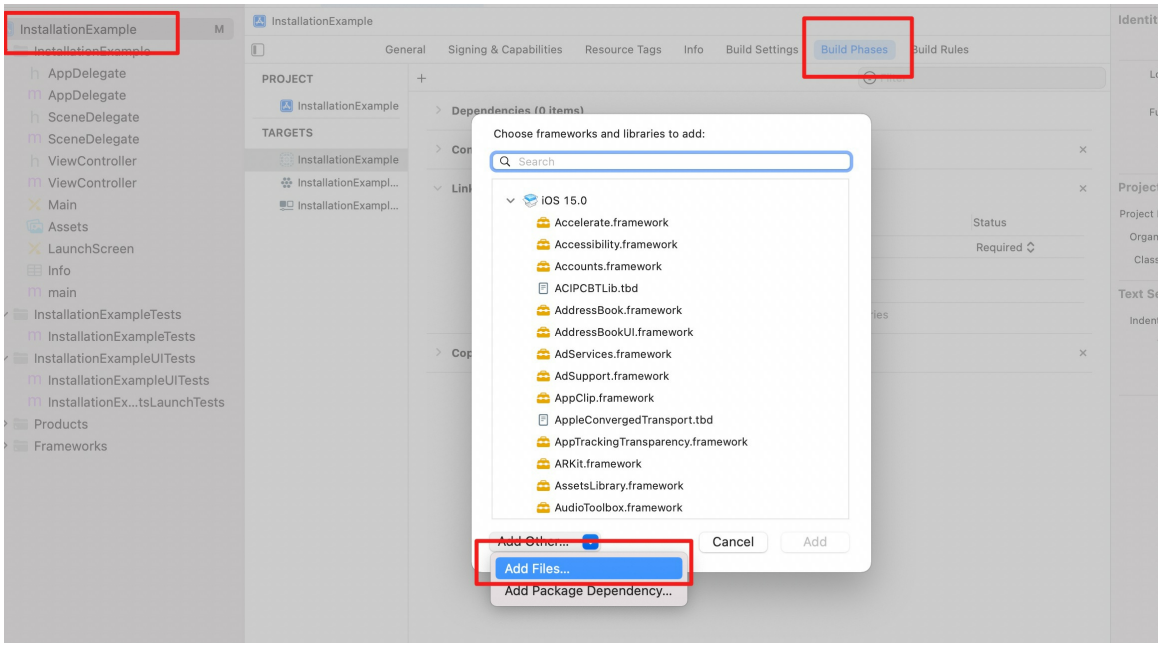
下载SDK

进入智能体文档中心，点击“下载专区>SDK&Demo下载”，即可下载iOS客户端。

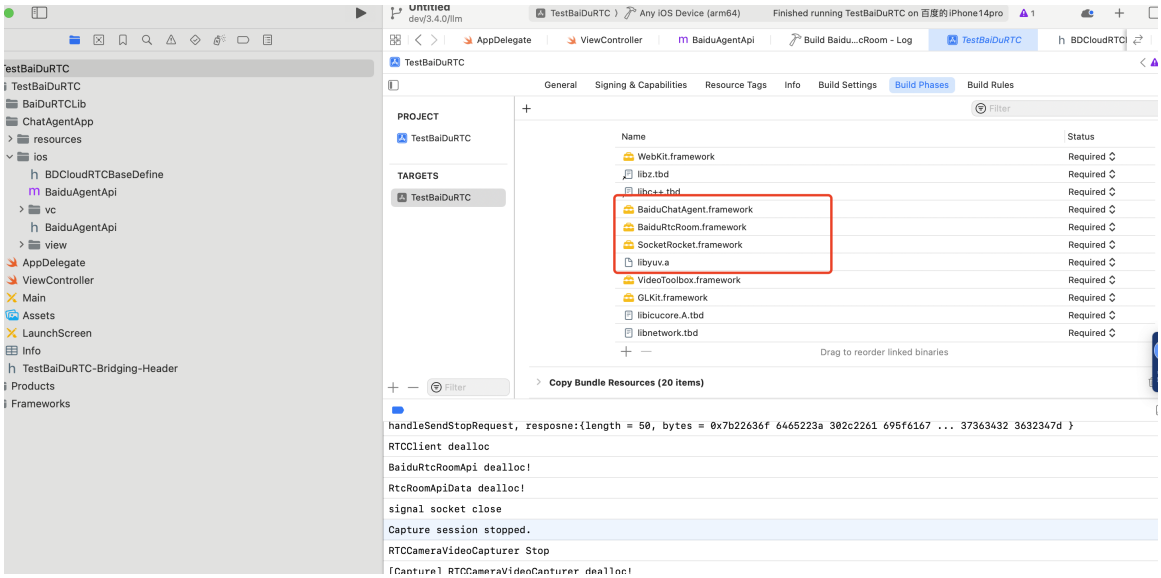
集成SDK 创建一个iOS项目，若已有 iOS 项目，可以直接集成 SDK。此版本SDK提供静态库及动态库形式。

添加 SDK 库文件

1. 将下载的SDK中lib文件夹内的 BaiduChatAgent.framework BaiduRtcRoom.framework SocketRocket.framework文件复制到项目文件夹下
2. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，添加 SDK 静态库文件到项目。



3. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，将libyuv.a添加到项目中
4. 添加依赖库



其中 ： BaiduChatAgent智能体SDK用户直接调用使用的库

BaiduRtcRoom 为 BRTC SDK 主依赖库，必须添加；

SocketRocket 为三方依赖库，必须添加；

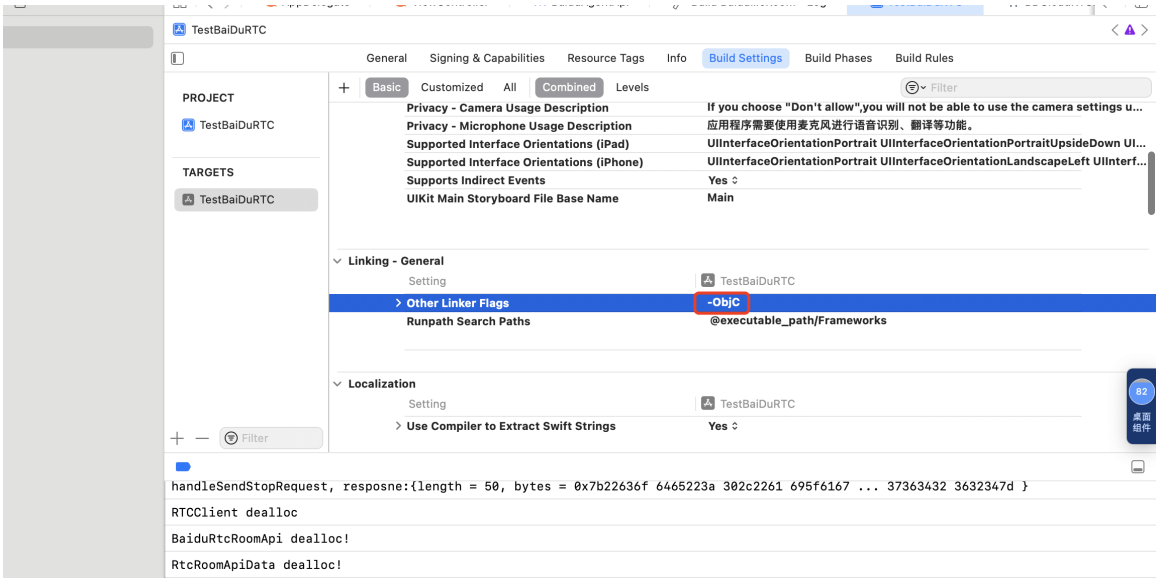
libyuv.a为百度内部依赖库，必须添加；

其他 BaiduRtc 前缀 framework 为 BRTC SDK 扩展模块依赖库，按需添加；

其余为系统依赖库，必须添加

项目设置

1. 打开 Xcode，选择：项目 TARGET ->Build Settings->Other Linker Flags，设置支持objC。



快速接入说明 本节介绍如何实现快速接入智能体。 分为两部分：启动远端智能体和本地智能体

启动远端智能体

1.代码实例http方式启动，获取远端启动参数参数信息 以下是DEMO的POST方式启动实例，调试不需要鉴权。注意，正式使用必须通过服务器上调用该接口，以及防AK/SK泄漏。

```
    NSString * url = [NSString stringWithFormat:@"%@%@", self.Server_host_env, self.Server_path_start];
    NSURL* urlRequest = [NSURL URLWithString:url];
    NSMutableURLRequest* request = [NSMutableURLRequest requestWithURL:urlRequest];
    NSMutableDictionary *dict  = [[NSMutableDictionary alloc] init];
    [dict setValue:params.appId forKey:@"app_id"];
    [dict setValue:@(params.quicStart) forKey:@"quick_start"];
    if (params.config && [params.config length] > 0) {
        [dict setValue:params.config forKey:@"config"];
    }

    NSError *error;
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dict options:NSJSONWritingPrettyPrinted error:&error];
    request.HTTPBody = jsonData;
    request.HTTPMethod = @"POST";
    [request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];

    NSURLSession *session = [NSURLSession sharedSession];
    [[session dataTaskWithRequest:request
                completionHandler:^(NSData * _Nullable data, NSURLResponse * _Nullable response, NSError * _Nullable
error) {
        NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse *) response;
        if ([httpResponse statusCode] != 200) {
            NSLog(@"url %@, statusCode:%ld", urlRequest, [httpResponse statusCode]);
            [self.delegate  BaiduAgentApiConnectFailed];
            return;
        }
        if (error) {
            NSLog(@"sendStartRequest fail, error: %@", error.description);
            return;
        }
        [self handleSendStartRequest:data block:block];

        }] resume];
```

2. 启动本地智能体，设置回调事件参数

```
 self.baiduChatAgent = [[BaiduChatAgent alloc] initWithParams:self.baiduChatAgentParms delegate:self];//实例生成

[self.baiduChatAgent call:self.tokenAgent instanceId:self.baiduChatAgentParms.instanceId]; //启动本地智能体终端
```

3. 结束智能体聊天

```
    [self.baiduChatAgent hangup];
    [self.baiduChatAgent destroy];
```

H5

H5快速实现大模型互动 本文介绍如何通过少量代码跑通大模型互动H5 DEMO，快速实现高质量、低延时的大模型互动功能。

实现大模型互动的步骤如下：

创建并且初始化 new BaiduRtcAgentClient(); 开始通话 Agent.Start(); 结束通话 Agent.Stop();

步骤：
1.前期准备 在实现 H5大模型互动demo之前，你需要有以下准备：

申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html

快速Demo 下载地址：https://brtc-sdk.cdn.bcebos.com/ai/agent/h5/BRTC.Agent.H5.SDK.V1.0.4.zip

2.使用SDK

```
 1). 初始化
var Agent = new BaiduRtcAgentClient();

2). 开始通话
var cfg = {
    llm: 'LLMxxx', // 私有LLM 可以填入'LLMOpenAPI', llm_url传入私有化地址
    llm_url: ''
};
Agent.Start({
    appid: 'AppIdxxx',
    cfg: cfg,
    remotevideoviewid: 'therevideo',
    localvideoviewid: 'herevideo',
    success: function () {
    },
    error: function (error) {
    },
    onmessage: function (msg) {
        console.log('onmessage id: ' + msg.id + ' data: ' + msg.data);
    }

3). 发送文本消息
Agent.sendMessageToUser('[T]:你好');

4). 结束通话
Agent.Stop();
```

微信小程序

微信小程序快速实现大模型互动 本文介绍如何通过少量代码快速跑通大模型互动微信小程序DEMO，实现高质量、低延时的大模型互动功能。

实现大模型互动的步骤如下：

创建并且初始化 require('./agent.js');
开始通话 Agent.Start();
结束通话 Agent.Stop();

步骤： 1.前期准备 在实现 微信小程序大模型互动demo之前，你需要有以下准备：

申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html

快速Demo 下载地址：https://brtc-sdk.cdn.bcebos.com/ai/agent/miniapp/BRTC.Agent.MiniApp.SDK.V1.0.5.zip

使用微信小程序开发工具建立工程：

详细要求可参考：BRTC微信小程序SDK集成

2.使用SDK

```
 1). 初始化
const brtc_agent = require('./agent.js');
var mAgent = brtc_agent.Agent;

2). 开始通话

var cfg = {
    llm: 'LLMxxx',
    llm_url: ''
};
mAgent.Start({
    appid: 'AppIdxxx',
    cfg: cfg,
    success: function (pushurl, agentId) {
    },
    error: function (error) {
    },
    onmessage: function (msg) {
        console.log('onmessage id: ' + msg.id + ' data: ' + msg.data);
    },
    remotevideocoming: function (id, display, attribute, pullurl) {
    },
});

3). 发送文本消息
mAgent.sendMessageToUser('[T]:你好');

4). 结束通话
mAgent.Stop();
```

集成大模型实时互动SDK

集成Android SDK

准备环境

本节将介绍如何创建项目，将BRTC SDK集成进你的项目中。

Android Studio 2022 或以上版本，Gradle 7.2或以上版本，编译环境请选择支持java8 Android KK（4.4）及以上的设备

*注：经过验证的开发环境如下：

```
 ------------------------------------------------------------
Gradle 7.2
------------------------------------------------------------

Build time:   2021-08-17 09:59:03 UTC
Revision:     a773786b58bb28710e3dc96c4d1a7063628952ad

Kotlin:       1.5.21
Groovy:       3.0.8
Ant:          Apache Ant(TM) version 1.10.9 compiled on September 27 2020
JVM:          11.0.26 (Homebrew 11.0.26+0)
OS:           Mac OS X 15.1 aarch64
```

下载SDK

大模型实时互动SDK必须依赖 RTCSDK

maven方式 推荐使用 Maven 在项目中接入，步骤如下：

RTC SDK maven 版本列表： https://repo1.maven.org/maven2/com/baidubce/mediasdk/brtc/

Agent SDK maven版本列表: https://repo1.maven.org/maven2/com/baidubce/mediasdk/lib_agent/

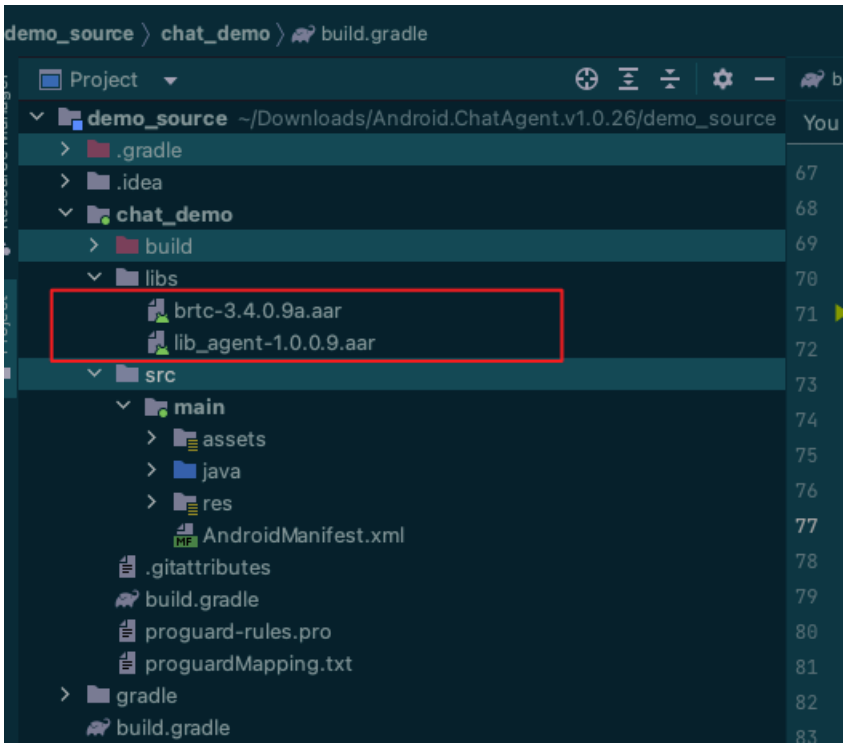
```
 implementation 'com.baidubce.mediasdk:brtc:2.xx.xx'
implementation 'com.baidubce.mediasdk:lib_agent:1.0.xx.xx'
```

离线版本

进入RTC文档中心，点击“下载专区>SDK&Demo下载”，即可下载客户端SDK。下载后请校验下载的包md5值与SDK中心里记录的是否一致。

创建Android项目，若已有 Android 项目，可以直接集成 SDK

将SDK包内 brtc-xxx.aar, lib-agent-xxxx.aar 拷贝到项目的libs目录.



依赖权限

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

<uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-permission
   android:name="android.permission.INTERNET" />

<uses-permission
   android:name="android.permission.CAMERA" />

<uses-permission
   android:name="android.permission.RECORD_AUDIO" />

<uses-permission
   android:name="android.permission.FOREGROUND_SERVICE" />

<uses-permission
   android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

<uses-permission
   android:name="android.permission.CHANGE_NETWORK_STATE" />

<uses-permission
   android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission
   android:name="android.permission.BLUETOOTH" />

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

代码防混淆

```
 -keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-keep interface okhttp3.Interceptor$* { *; }

-keep class com.baidu.rtc.BaiduRtcRoom {*;}
-keep class com.webrtc.** {*;}
-keep class com.baidu.rtc.** {*;}
```

项目配置

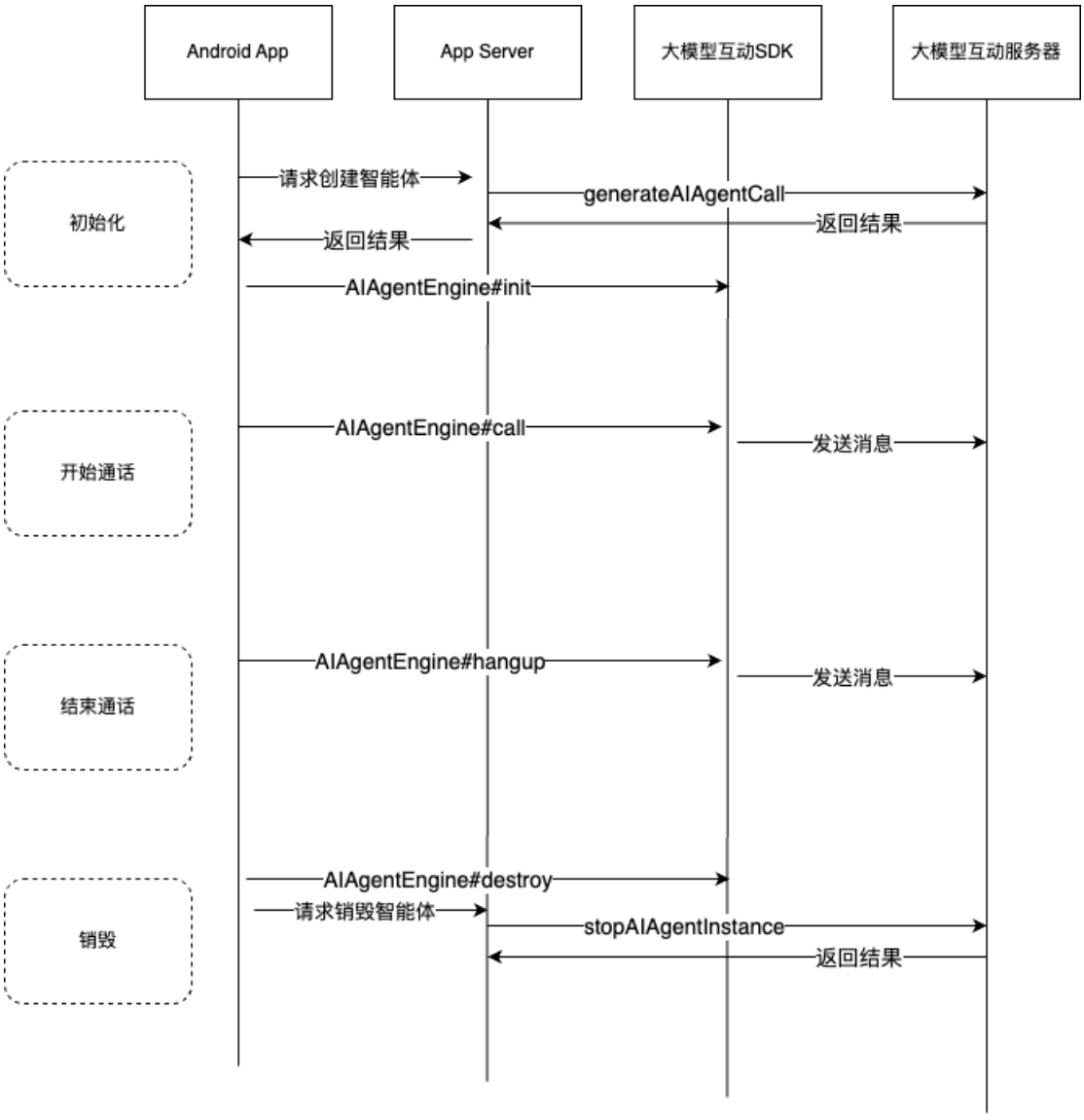
```
  android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    packagingOptions {
        doNotStrip "**/*.so"
        pickFirst "**/*.so"
        resources.excludes.add("META-INF/*")
    }

    sourceSets.main {
        jniLibs.srcDir 'libs'
        jni.srcDirs = []
    }
}
```

集成大模型互动AndroidSDK

本节介绍如何实现大模型实时互动。调用时序见下图:



关键代码段

- 创建大模型实时互动Engine

```
  private AIAgentEngine.AIAgentEngineParams mAiAgentParams;
private AIAgentEngine mAiAgent;

    mAiAgentParams = new AIAgentEngine.AIAgentEngineParams();
    mAiAgentParams.appId = mClientData.mAppId;
    mAiAgentParams.context = mClientData.mContext;
    mAiAgent = AIAgentEngine.init(mActivity, mAiAgentParams);
    mAiAgent.setCallback(this);
```

- 开始通话

```
    mAiAgent.call(mClientData.mTokenStr, JavaTypesHelper.toLong(mClientData.mInstanceId, 1));
```

- 结束通话

```
    mAiAgent.hangup();
```

- 销毁

```
    if (mAiAgent != null) {
      mAiAgent.destroy();
  }
```

集成RTOS SDK

准备环境

本节将介绍如何创建项目，将BRTC RTOS SDK集成进你们的项目中。

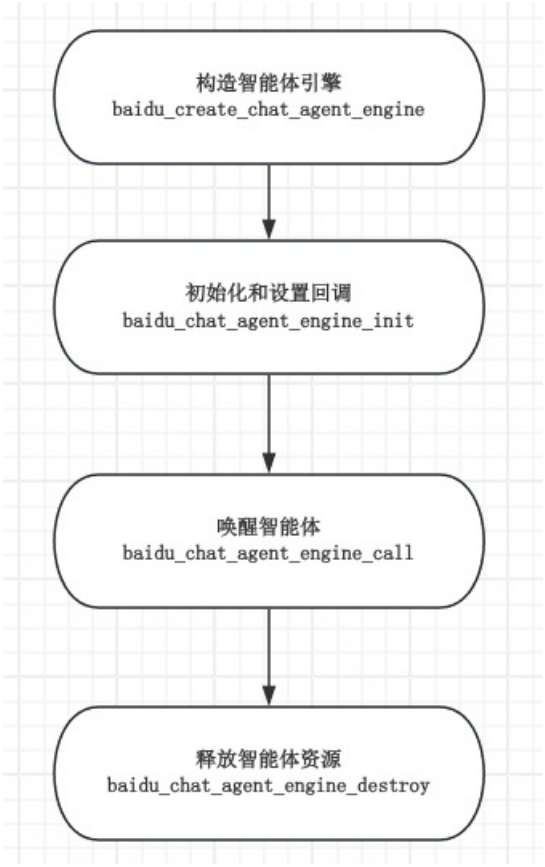
针对不同的芯片平台集成RTOS SDK需要注意芯片类型和芯片的基线版本的差异。

*注：下面是经过验证的开发环境如下：

芯片平台	依赖的软件版本	描述
乐鑫ESP32S3	esp-adf版本：v2.7-59-g8301b97d esp-idf版本：v5.3.1	集成时建议基线版本不要低于该版本
杰里AC79	AC79NN_SDK_V1.2.0	集成时建议基线版本不要低于该版本

SDK调用流程图

下面是大模型实时互动RTOS SDK的调用流程图。



实例代码

1. 创建大模型实时互动Engine

```
 BaiduChatAgentEvent events = {
   .onError = onErrorCallback,
   .onCallStateChange = onCallStateChangeCallback,
   .onConnectionStateChange = onConnectionStateChangeCallback,
   .onUserAsrSubtitle = onUserAsrSubtitleCallback,
   .onAIAgentSubtitle = onAIAgentSubtitleCallback,
   .onAIAgentSpeaking = onAIAgentSpeaking,
   .onFunctionCall = onFunctionCall,
   .onAudioPlayerOp  = onAudioPlayerOp,
   .onMediaSetup = onMediaSetup,
   .onAudioData = onAudioData,
   .onVideoData = onVideoData,
   .onLicenseResult = onLicenseResult,
};
BaiduChatAgentEngine* engine = baidu_create_chat_agent_engine(&events);
```

2. 创建智能体和初始化SDK的参数 2.1 服务端通过http创建智能体，客户端获取服务端返回的参数

```
###### define JSON_CONFIG_TEMPLATE "{\"app_id\": \"%s\", \"config\" : \"{\\\"llm\\\" : \\\"%s\\\", \\\"llm_token\\\" : \\\"no\\\", \\\"rtc_ac\\\": \\\"g722\\\", \\\"lang\\\" : \\\"%s\\\"}\", \"quick_start\": true}"
//客户自己的服务端通过https按照上面格式发送请求创建智能体
```

2.2 客户端初始化SDK参数

```
  const char* json_str =
"{"
    "\"ai_agent_instance_id\": 2230595646193664,"
    "\"context\": {"
        "\"cid\": 1,"
        "\"token\": \"00415f2bd1c3fe5dbc02cc49af726b18d6c6bfc56dc174238426178be806a1742470661\""
    "}"
"}"; //这是服务端返回的参数，需要将参数给客户端

void setUserParameters(AgentEngineParams *params) {
    snprintf(params->agent_platform_url, sizeof(params->agent_platform_url), "%s", SERVER_HOST_ONLINE);
    snprintf(params->appid, sizeof(params->appid), "%s", BDCloudDefaultRTCAppID);
    snprintf(params->userId, sizeof(params->userId), "%s", "12345678");//用户ID，可以设置指定人设
    snprintf(params->cer, sizeof(params->cer), "%s", "./a.cer");
    snprintf(params->workflow, sizeof(params->workflow), "%s", "VoiceChat");
    snprintf(params->license_key, sizeof(params->license_key), "%s", "abcdxxx");//向百度购买设备license可以获取到key

    params->instance_id = 10373;
    params->verbose = false;
    params->enable_internal_device =false;
    params->enable_local_agent =false;
    params->enable_voice_interrupt = true;
    params->level_voice_interrupt = 80;
    strncpy(params->llm, "LLMRacing", sizeof(params->llm) - 1);
    strncpy(params->lang, "zh", sizeof(params->lang) - 1);
    params->AudioInChannel = 1;
    params->AudioInFrequency = 16000;
}

    AgentEngineParams agentParams;
    memset(&agentParams, 0, sizeof(agentParams));
    setUserParameters(&agentParams);
    int result = baidu_chat_agent_engine_init(engine, &agentParams);
```

3. 开始通话

```
  baidu_chat_agent_engine_call(engine);
```

4. 结束通话

```
  baidu_chat_agent_engine_destroy(engine);
```

客户端SDK

Android SDK

初始化接口

SDK初始化

```
    /** 初始化Engine */
    public static AIAgentEnginImpl init(Context context, AIAgentEngineParams params)

    public static class AIAgentEngineParams {
        /** appid */
        public String appId = "";
        /** 调试开关 */
        public boolean verbose = false;
        /** 开启终端侧TTS */
        public boolean enableTerminalTTS = false;
        /** 说话音频打断 */
        public boolean enableVoiceInterrupt = true;
        /** 实例id  服务器端下发，当前调试默认即可 */
        public long aiAgentInstanceId = 0;
        /**容许外部输入音频，通过接口 pushAudioFrame 输入音频PCM数据 */
        public boolean enableExternalAudioInput = false;
        /**容许外部输出音频，通过接口回调 onPlaybackAudioFrame 返回音频PCM数据 */
        public boolean enableExternalAudioOutput = false;
        /** dump 采集音频 */
        public boolean dumpAudioInput = false;
        /** 音频source */
        public int inputAudioSource = -1;
        /** 工作流类型，默认用VoiceChat */
        public String workflow = "VoiceChat";
        /** 音频编码器 默认opus */
        public String audioCodec = RtcParameterSettings.AudioCodecId.OPUS;
        /** 服务器端接口下发 cid, token， 调试模式传入空即可 */
        public String context = "{ "cid": 1, "token": "xxx-aa-bb" }";
    }
```

SDK初始化。

参数

参数	类型	描述
context	Context	Android上下文环境
params	AIAgentEngineParams	Engine初始化参数

返回

AIAgentEnginImpl 实例对象：成功； null：失败；

设置回调

```
    public abstract void setCallback(AIAgentEngineCallback callback);
```

参数

参数	类型	描述
callback	AIAgentEngineCallback	事件回调接口

返回

无

Agent相关接口

开启通话

```
public abstract void call(String token， long instanceId);
```

开启通话，开始音频采集和音频播放；

参数

参数	类型	描述
token	String	鉴权token
instanceId	long	实例id

返回

无

结束通话

```
public abstract void hangup();
```

结束通话，停止音频采集和音频播放；

参数

参数	类型	描述

返回

无

发送文本

```
public abstract void setTextToAIAgent(String text);
```

发送文本消息给智能体，作为query向大模型进行提问；

参数

参数	类型	描述
text	String	文本query

返回

无

发送文本并且打断

```
public abstract void sendTextToAIAgentAndInterrupt(String text);
```

发送文本消息给智能体，作为query向大模型进行提问，并且立即打断当前播报；

参数

参数	类型	描述
text	String	文本query

返回

无

发送文本直接播报

```
public abstract void sendTextToTTS(String text);
```

发送文本消息给TTS模块直接进行播报；

参数

参数	类型	描述
text	String	TTS播报文本

返回

无

主动打断

```
public abstract void interrupt();
```

打断当前播报内容，停止播放；

参数

参数	类型	描述

返回

无

销毁Engine

```
public abstract void destroy();
```

销毁Engine，释放资源；

返回

无

发送FunctionCall结果

```
    public abstract void sendFunctionCallResult(String id, String result);
```

发送FunctionCall结果；

参数

参数	类型	描述
id	String	唯一表示
result	String	结果 例如： `{"result":"ok"}`

返回

无

设置数字人View

```
    public abstract void setDigitalDisplay(ViewGroup viewGroup, ViewGroup.LayoutParams params,
RTCVideoView.ScalingType type);
```

设置数字人视图；

参数

参数	类型	描述
viewGroup	ViewGroup	父容器
params	ViewGroup.LayoutParams	布局参数

type	RTCVideoView.ScalingType	缩放模式, 支持模式：
		`RTCVideoView.ScalingType.SCALE_ASPECT_FIT`
		`RTCVideoView.ScalingType.SCALE_ASPECT_FILL`

返回

无

上传文件

```
    public abstract boolean uploadFile(String path, int expire);
```

上传文件会收到 `onUploadFileStatus` 事件回调

注意 暂时仅仅支持 JPEG/PNG/JPG 三种数据格式，并且大小不超过7MB;

参数

参数	类型	描述
path	String	文件路径，必须具有可读权限
expire	int	过期时间 单位秒, 最大值默认180秒 （图片支持多轮次会话引用时间），最小值0 （图片仅仅支持一轮会话引用）

返回

无

音频相关接口

静音播放

```
public abstract void mutePlayback(boolean isMute);
```

音频播放声音控制；

参数

参数	类型	描述
isMute	boolean	是否静音

返回

无

静音麦克风

```
public abstract void muteMic(boolean isMute);
```

音频采集声音控制；

参数

参数	类型	描述
isMute	boolean	是否静音

返回

无

扬声器播放

```
public abstract void switchToSpeaker(boolean speaker);
```

声音使用扬声器播放控制；

参数

参数	类型	描述
speaker	boolean	是否使用扬声器播放

返回

无

注册外部TTS

```
    public abstract void registerExternalTTS(Constants.TTSService service);

/** TTS 服务 */
public static interface TTSService {
    /** 停止播放TTS，打断语音TTS播报时候调用,建议mute speaks 1000ms */
    public void onStop();

    /** 开始TTS */
    public void onStart(String msg);
}
```

注册外部TTS服务

参数

参数	类型	描述
service	Constants.TTSService	外部tts服务

返回

无

外部音频采集（1）

```
        public abstract void pushAudioFrame(byte[] data, long timestamp, int sampleRate, int channels);
```

外部音频采集

注意 接入方负责将音频输入PCM 输入进入 （当前要求16K音频采样，单声道, AudioFormat.ENCODING_PCM_16BIT），每次输入10ms音频数据长度；

参数

参数	类型	描述
data	byte数组	PCM音频数据, 单声道, AudioFormat.ENCODING_PCM_16BIT, 10ms 音频数据长度 320字节
timestamp	long	时间戳
sampleRate	int	音频采样率 使用16K音频采样率
channels	int	音频声道数

返回

无

外部音频采集（2）

```
public abstract void pushAudioFrame(ByteBuffer data, long timestamp, int sampleRate, int channels);
```

外部音频采集

注意 接入方负责将音频输入PCM 输入进入 （当前要求16K音频采样，单声道, AudioFormat.ENCODING_PCM_16BIT），每次输入10ms音频数据长度；

参数

参数	类型	描述
data	ByteBuffer	PCM音频数据, 单声道, AudioFormat.ENCODING_PCM_16BIT, 10ms 音频数据长度 320字节
timestamp	long	时间戳
sampleRate	int	音频采样率 使用16K音频采样率
channels	int	音频声道数

返回

无

事件回调

AIAgent 回调

错误回调

```
public void onError(int error, String msg, Bundle bundle)
```

出错回调，出错后需要结束通话；

参数

参数	类型	描述
error	int	错误码
msg	String	错误信息
bundle	Bundle	错误描述详细信息

通话状态变化

```
public void onCallStateChange(int state);
```

通话状态变化；

参数

参数	类型	描述
state	int	开始通话 {@link Constants.CallState#ON_CALL_BEGIN } 结束通话 {@link Constants.CallState#ON_CALL_END }

链接状态变化

```
public void onConnectionStateChange(int state);
```

通话状态变化；

参数

参数	类型	描述
state	int	链接断开 {@link Constants.ConnectionState#CONNECTION_STATE_DISCONNECTED } 重连接 {@link Constants.ConnectionState#CONNECTION_STATE_RECONNECTING } 链接成功 {@link Constants.ConnectionState#CONNECTION_STATE_CONNECTED }

用户端ASR结果

```
public void onUserAsrSubtitle(String text, boolean isFinal);
```

用户端ASR结果；

参数

参数	类型	描述
text	String	ASR识别结果
isFinal	boolean	标记是否最终结果

AI智能体结果

```
public void onAIAgentSubtitle(String text, boolean isFinal);
```

AI智能体结果；

参数

参数	类型	描述
text	String	智能体结果
isFinal	boolean	标记是否最终结果

AI智能体音频状态变化

```
    public void onAIAgentAudioStateChange(@Constants.AIAgentAudioStateType int newState);

/** 智能体音频状态 */
@Retention(RetentionPolicy.SOURCE)
public @interface AIAgentAudioStateType {
    /** 大模型停止说话 */
    int STOPPED = 1;
    /** 大模型说话 */
    int SPEAKING = 2;
}
```

AI智能体音频状态变化；

参数

参数	类型	描述
newState	@Constants.AIAgentAudioStateType int	音频状态

用户音频状态变化

```
    public void onUserAudioStateChange(@Constants.UserAudioStateType int newState);

/** 用户音频状态 */
@Retention(RetentionPolicy.SOURCE)
public @interface UserAudioStateType {
    /** 用户停止说话*/
    int STOPPED = 1;
    /** 用户说话*/
    int SPEAKING = 2;
}
```

用户声音状态变化；

参数

参数	类型	描述
newState	@Constants.UserAudioStateType int	音频状态

智能体音频能量回调

```
    public void onAIAgentAudioLevel(int level);
```

智能体音频能量回调， 间隔100ms回调一次；

参数

参数	类型	描述
level	int	音频能量

用户音频能量回调

```
public void onUserAudioLevel(int level);
```

用户音频能量回调， 间隔100ms回调一次；

参数

参数	类型	描述
level	int	音频能量

functionCall回调

```
public void onFunctionCall(String id, String params);
```

用户收到function call, 处理完成后需要调用 `sendFunctionCallResult`

参数

参数	类型	描述
id	String	单次function call唯一标识
params	String	参数 一般是json string 例如: `{"function_name":"phone_call","parameter_list":[{"called_number":"1891017000"},{"called_name":"我的父亲"}`

文件上传状态

```
public void onUploadFileStatus(int code, String msg)
```

文件上传后状态回调

参数

参数	类型	描述
code	int	成功 {@link Constants#NO_ERROR } 失败 其他错误码
msg	String	成功时候返回文件名，失败时候返回错误消息

外部音频播放回调

```
public void onPlaybackAudioFrame(byte[] data, int sampleRate, int channelCount)
```

当打开外音频播放 `AIAgentEngine.AIAgentEngineParams#enableExternalAudioOutput` ，有该回调事件；

注意 预期这里直接将音频输入给AudioTrack，如果不输入到AudioTrack，这里需要 TimeUnit.MILLISECONDS.sleep(10);

参数

参数	类型	描述
data	byte数组	PCM音频数据
sampleRate	int	采样率
channelCount	int	声道数

agent意图

```
public void onAgentIntent(String type, Bundle bundle)
```

意图回调

当前仅仅支持 Constants.AgentIntentType#IMAGE_UPLOAD和Constants.AgentIntentType#AGENT_EVENT_PLAY_AUDIO;

参数

参数	类型	描述
type	String	图片上传: {@link Constants.AgentIntentType#IMAGE_UPLOAD } 播放音频: {@link Constants.AgentIntentType#AGENT_EVENT_PLAY_AUDIO }
bundle	Bundle	其他附加信息

iOS SDK

智能体开启/关闭

SDK初始化

```
- (instancetype)initSDKWithAppID:(NSString *)appId tokenStr:(NSString *)tokenStr delegate:
(id<BaiduRtcRoomDelegate>)delegate;
```

初始化SDK。

初始化SDK时调用。设置 AppID、日志等参数

参数

参数	类型	描述
appId	NSString*	RTC 基础业务单元的唯一标识
config	NSString*	服务器端接口下发数据 { "cid": 1, "token": "xxx-aa-bb" }
instanceId	NSInteger	实例id 服务器端下发，当前调试默认即可
verbose	BOOL	日志开关
enableVoiceInterrupt	BOOL	是否允许自动打断
userAudioLevelInterrupt	NSInteger	本地音频打断能量值 默认80

返回

返回 SDK 实例，nil 表示初始化失败

启动本地智能体

```
- (void)call:(NSString *)token instanceId:(NSInteger)instanceId;
```

该函数在 initSDKWithAppID后调用，启动本地的智能体本地端

参数

参数	类型	描述
token	NSSTring*	通过url获取远端的智能体的返回的参数，token
instanceId	NSString*	instanceId 实例id

返回

无

挂断本地退出

```
- (void)hangup;
```

主动退出远端和本地智能体

参数

无

返回

资源销毁

```
- (void)destroy;
```

销毁本地资源

返回

无

智能体交互控制

打断智能体会话

```
- (void)interrupt;
```

打断智能体的对话指令，开启新的回话。

参数 无

返回

静音播放

```
- (void)mutePlayback:(BOOL)isMute;
```

开启和关闭静音智能体的播放

参数

参数	类型	描述
isMute	BOOL	YES/NO

返回

void

静音mic

```
- (void)muteMic:(BOOL)isMute;
```

采集接口静音操作

参数

参数	类型	描述
isMute	BOOL	YES/NO

返回

无

切换听筒和扬声器

```
-- (void)switchToSpeaker:(BOOL)speaker;
```

听筒和外放切换

参数

参数	类型	描述
speaker	BOOL	YES 扬声器 NO 听筒

返回

无

预置扬声器状态

```
- (void)presetLoudSpeaker:(BOOL)isSpeaker;
```

预置听筒/扬声器播放语音，在 initWithParams 之后，call 之前调用

参数

参数	类型	描述
isSpeaker	BOOL	YES 扬声器 NO 听筒

返回

无

发送文本给智能体

```
- (void)sendTextToAIAgent:(NSString *)text;
```

发送文本信息

参数

参数	类型	描述
text	NSString *	发送文本信息

返回

无

发送文本给智能体，打断播报

```
- (void)sendTextToAIAgentAndInterrupt:(NSString *)text;
```

发送文本给智能体，并且打断当前播报

参数

参数	类型	描述
text	NSString *	发送文本信息

返回

无

发送文本给TTS - (void)sendTextToTTS:(NSString *)text;

发送文本给TTS，直接进行播报

参数

参数	类型	描述
text	NSString *	发送文本信息

返回

无

FunctionCall处理结果返回

```
- (void)sendFunctionCallResult:(NSString*) functionID result:(NSString *)msg;
```

处理结果返回

参数

参数	类型	描述
functionID	NSString*	fuction的默认ID
msg	NSString*	返回结果

返回

无

支持数字人功能

```
-(void)setDigitalDisplay:(RTCRemoteVideoView *) remoteVideoView;
```

开启设置数字人的显示的View

参数

参数	类型	描述
remoteVideoView	RTCRemoteVideoView	RTC里面的通用显示view

RTCRemoteVideoView 使用BRTC的View

返回

无

事件回调

错误信息回调

```
-(void)onError:(NSInteger)errCode errMsg:(NSString *)errMsg extInfo:(nullable NSDictionary*)extInfo;
```

错误信息回调

参数

参数	类型	描述
errCode	NSInteger	errno
errMsg	NSString *	错误文本
extInfo	NSDictionary*	错误的json串返回

智能体登录状态改变通知

```
- (void)onCallStateChange:(AGentCallState)state;
```

错误信息回调

参数

参数	类型	描述
state	NSInteger	状态信息 AGentCallState

回调事件

```
typedef NS_ENUM(NSInteger, AGentCallState) {
    AGENT_CALL_BEGIN = 1,
     AGENT_CALL_END
};
```

链路状态信息回调

```
- (void)onConnectionStateChange:(AGentConnectState)state;
```

智能体链路信息回调

参数	类型	描述
state	NSInteger	状态信息 AGentCallState

回调事件

```
typedef NS_ENUM(NSInteger, AGentCallState) {
    AGENT_CONNECTION_STATE_DISCONNECTED = 1,
    AGENT_CONNECTION_STATE_CONNECTING,
    AGENT_CONNECTION_STATE_CONNECTED,
    AGENT_CONNECTION_STATE_RECONNECTING
};
```

本地语音Asr文本识别返回

```
- (void)onUserAsrSubtitle:(NSString *)text isFinal:(BOOL)isFinal;
```

链接状态变更回调

参数

参数	类型	描述
text	NSString *	ASR 返回内容
isFinal	BOOL	是否是结尾

远端语音TTS文本识别返回

```
- (void)onAIAgentSubtitle:(NSString *)text isFinal:(BOOL)isFial;
```

视频渲染首帧回调

参数

参数	类型	描述
text	NSString *	TTS返回文本内容
isFinal	BOOL	是否是结尾

AI智能体音频状态变化

```
- (void)onAIAgentAudioStateChange:(AGentAudioState)newState;
```

智能体音频状态变化

参数

参数	类型	描述
newState	AGentAudioState	用户uid

```
typedef NS_ENUM(NSInteger, AGentAudioState) {
    AGENT_STOPPED = 1,
    AGENT_SPEAKING
};
```

用户声音状态变化

```
- (void)onUserAudioStateChange:(AGentAudioState)newState;
```

远端用户离开事件

参数

参数	类型	描述
newState	AGentAudioState	用户uid

```
typedef NS_ENUM(NSInteger, AGentAudioState) {
    AGENT_STOPPED = 1,
    AGENT_SPEAKING
};
```

智能体音频能量

```
- (void)(void)onAIAgentAudioLevel:(NSInteger)level;
```

远端智能体的音量返回

参数

参数	类型	描述
level	NSInteger	返回智能体的音量

用户音频能量

```
- (void)onUserAudioLevel:(NSInteger)level;
```

本地用户音量返回

参数

参数	类型	描述
level	NSInteger	返回本地的音量

回调functionCall事件

```
- (void)onFunctionCall:(NSString*) functionID param:(NSString*) params;
```

本地用户音量返回

参数

参数	类型	描述
functionID	NSString*	返回function的ID
params	NSString*	返回的function的详细json串

H5 SDK

初始化Agent实例

var Agent = new BaiduRtcAgentClient();

介绍

初始化时使用。

参数

无

返回

Agent对象

开始通话

Agent.Start()

介绍

使用配置参数进行通话。

参数

无

返回

无

```
  var cfg = {

};
Agent.Start({
    appid: 'AppIdxxx',
    cfg: cfg,
    remotevideoviewid: 'therevideo',
    localvideoviewid: 'herevideo',
    success: function () {
    },
    error: function (error) {
    },
    onmessage: function (msg) {
        console.log('onmessage id: ' + msg.id + ' data: ' + msg.data);
    }
```

参数详解

参数	类型	描述	默认值
appid	string	百度派发的AppID, 开发者的唯一标识	
cfg	json	互动智能体配置， 数字人智能体详细配置参见“数字人智能体”一节	
remotevideoviewid	string	显示远端视频，来自html的DOM对象的ID名称	
localvideoviewid	string	显示本地摄像头视频，来自html的DOM对象的ID名称	
onmessage	function(msg)	消息事件回调{msg.id,msg.data}	
success	function()	Start()成功	
error	function(error)	Start()失败，或运行过程中出现了错误	

停止通话

Agent.Stop();

介绍

停止智能体通话

参数

无

返回

无

麦克风静音

```
BRTC_MuteMicphone(muted);
```

参数

参数	类型	描述
muted	bool	静音标识， true表示要静音， false 是取消静音

返回

无

发送消息给智能体

Agent.sendMessageToUser(msg);

介绍

本函数用来给特定ID用户发送消息或者向房间内发送广播消息。 消息在接收端的onmessage回调函数中接收。发送用户消息的频率应小于10次/秒，超出的话用户消息可能会被丢弃。

前置条件: 在调用Start登录成功后才能调用。

参数

msg, 需要发送的消息内容， 为一个字符串， 比如: '[T]:你好'

返回

无

接收智能体消息

onmessage: function (msg)

介绍

本函数用来接收智能体消息。

参数

msg.data为一个字符串， 比如: '[A]:我是文小言'

更详细的消息定义见“客户端与服务端消息” 一节。

返回

无

微信小程序 SDK

获取SDK实例

```
const brtc_agent = require('./agent.js');
var mAgent = brtc_agent.Agent;
```

实例的各API函数说明如下:

开始通话

```
mAgent.Start()
```

介绍

开始通话时使用。

参数

参数	类型	描述
server	string	百度的RTC 服务器， 使用默认值即可
appid	string	百度 派发的数字 ID, 开发者的唯一标识

该接口参数数量较多，请参考下面的参数详解进行了解。

返回

无

参数详解

```
 // 启动Agent例子代码
var cfg = {
    llm: 'LLMxxx',
    llm_url: ''
};
mAgent.Start({
    appid: 'AppIdxxx',
    cfg: cfg,
    success: function (pushurl, agentId) {
    },
    error: function (error) {
    },
    onmessage: function (msg) {
        console.log('onmessage id: ' + msg.id + ' data: ' + msg.data);
    },
    remotevideocoming: function (id, display, attribute, pullurl) {
    },
});
```

参数	类型	描述	默认值
appid	string	百度 派发的数字 ID, 开发者的唯一标识	
cfg	json	智能体配置	cfg配置LLM，TTS等，互动数字人的配置见“互动数字人”一节
remotevideocoming	function(id,display,attribute, pullurl)	远端用户流上线的回调	pullurl 是拉流的RTMP地址, 可以用liveplayer进行播放
remotevideoleaving	function(id)	远端用户流离开的回调	
onmessage	function(msg)	消息事件回调{msg.id,msg.data}	
userevent_joindroom	function(id,display,attribute)	用户加入房间的事件，此时用户还没有发布流	
userevent_leavingroom	function(id,display)	用户离开房间	
success	function(pushurl，agentid)	Start()成功	pushurl是服务器分配的用于推流的RTMP地址, 可以用livepusher进行推流
error	function(error)	Start()失败，或运行过程中出现了错误	
destroyed	function(error)	运行过程中出现错误被销毁的回调	
debuglevel	bool/array	是否打印调试信息	默认值为false， 可取值为: true, false, 'all', ['debug','log','error']

结束通话

```
 mAgent.Stop();
```

介绍

结束通话

参数

无

返回

无

发送文本消息

```
mAgent.sendMessageToUser
```

介绍

本函数用来发送广播消息。 发送用户消息的频率应小于100次/秒，超出的话用户消息可能会被丢弃。

参数

msg, 需要发送的消息内容， 为一个字符串， 比如: '[T]:你好'

返回

无

接收消息回调

onmessage: function (msg) { }

介绍

本函数用来接收消息。发送用户消息的频率应小于100次/秒，超出的话用户消息可能会被丢弃。

参数

msg, 消息值 消息的消息见“客户端和服务端消息” 一节。

返回

无

RTOS SDK

初始化接口

SDK初始化

```
typedef struct AgentEngineParams {
    char agent_platform_url[256];       // 登陆agent智能体中心的地址
    char config[256];                   // 配置文件路径
    char appid[64];                     // 应用ID
    char userId[256];                   //用户id，根据id可以指定人设
    char cer[256];                      // 默认证书路径
    char token[256];                    // 鉴权token
    char workflow[64];                  // 工作流类型，默认VoiceChat
    char llm[64];                       // 大模型类型
    char lang[16];                      // 语言类型，默认zh
    char cid[64];                       // 通讯id
    long long instance_id;              // 实例ID，服务器下发，当前默认即可
    int level_voice_interrupt;          // 本地音频打断能量值，默认80
    int AudioInChannel;                 // 音频输入通道数，默认1
    int AudioInFrequency;               // 音频输入采样率，默认16000
    char AudioIncodecType[16];          // 音频输入的格式，默认是pcm
    char AudioOutcodecType[16];         // 音频播放的格式，默认是g711u
    int VideoWidth;                     // 视频宽度
    int VideoHeight;                    // 视频高度
    bool verbose;                       // 是否开启详细日志，默认关闭
    bool enable_voice_interrupt;        // 是否开启本地音频打断，默认关闭
    bool enable_internal_device;        // true：内部采集数据模式，flase: 外部采集数据模式
    bool enable_local_agent;            // true：本地请求创建智能体，flase: 服务端请求创建智能体
    char remote_params[1024];           //获取远端的服务的参数，替代用户自己解析，例如instance_id，token;
    char license_key[256];              //客户需要向百度购买设备license获取对应的key
} AgentEngineParams;

BaiduChatAgentEngine* baidu_create_chat_agent_engine(const BaiduChatAgentEvent* events);
int baidu_chat_agent_engine_init(BaiduChatAgentEngine* engine, const AgentEngineParams* param);
```

介绍

创建brtc chat agent engine并初始化，首先通过baidu_create_chat_agent_engine创建engine, 然后通过baidu_chat_agent_engine_init初始化。

参数

BaiduChatAgentEvent句柄

返回

BaiduChatAgentEngine句柄

介绍

创建brtc chat agent engine，并设置事件回调。

参数

参数	类型	描述
events	const BaiduChatAgentEvent*	BaiduChatAgentEvent句柄

返回

BaiduChatAgentEvent句柄

baidu_chat_agent_engine_init

```
int baidu_chat_agent_engine_init(BaiduChatAgentEngine* engine, const AgentEngineParams* param);
```

介绍

初始化SDK。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄
param	const AgentEngineParams*	AgentEngineParams初始化参数

返回

200 成功， 其他值 失败

Agent相关接口

开启对话

baidu_chat_agent_engine_call

```
void baidu_chat_agent_engine_call(BaiduChatAgentEngine* engine);
```

介绍

开启对话。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄

返回

无

baidu_chat_agent_engine_send_audio

```
void baidu_chat_agent_engine_send_audio(BaiduChatAgentEngine* engine, const uint8_t* data, size_t len);
```

介绍

发送音频。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄
data	const uint8_t*	发送的音频数据
len	size_t	发送的音频数据长度

返回

无

baidu_chat_agent_engine_send_text

```
void baidu_chat_agent_engine_send_text(BaiduChatAgentEngine* engine, const char* text);
```

介绍

发送文本给AI大模型。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄
text	const char*	发送的文本

返回

无

baidu_chat_agent_engine_send_text_to_TTS

```
void baidu_chat_agent_engine_send_text_to_TTS(BaiduChatAgentEngine* engine, const char* text);
```

介绍

发送文本给TTS，直接进行播报。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄
text	const char*	发送的文本

返回

无

baidu_chat_agent_engine_send_functioncall_result

```
void baidu_chat_agent_engine_send_functioncall_result(BaiduChatAgentEngine* engine, const char* id, const char* result);
```

介绍

发送FunctionCall结果。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄
id	const char*	唯一标识
result	const char*	结果 例如： {"result":"ok"}

返回

无

baidu_chat_agent_engine_interrupt

```
void baidu_chat_agent_engine_interrupt(BaiduChatAgentEngine* engine);
```

介绍

主动打断。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄

返回

无

baidu_chat_agent_engine_destroy

```
void baidu_chat_agent_engine_destroy(BaiduChatAgentEngine* engine);
```

介绍

结束对话和销毁。

参数

参数	类型	描述
engine	BaiduChatAgentEngine*	BaiduChatAgentEngine句柄

返回

无

事件回调

```
void (*onError)(int errCode, const char* errMsg);
```

介绍

错误回调。

参数

参数	类型	描述
errCode	int	错误码
errMsg	const char*	错误信息

返回

无

```
void (*onCallStateChange)(AGentCallState state);
```

介绍

通话状态变化。

参数

参数	类型	描述
state	AGentCallState	开始通话 {@link Constants.CallState#ON_CALL_BEGIN } 结束通话 {@link Constants.CallState#ON_CALL_END }

返回

无

```
void (*onConnectionStateChange)(AGentConnectState state);
```

介绍

链接状态变化。

参数

参数	类型	描述
state	AGentConnectState	链接断开 {@link Constants.ConnectionState#CONNECTION_STATE_DISCONNECTED} 重连接 {@link Constants.ConnectionState#CONNECTION_STATE_RECONNECTING} 链接成功 {@link Constants.ConnectionState#CONNECTION_STATE_CONNECTED}

返回

无

```
void (*onUserAsrSubtitle)(const char* text,  bool final);
```

介绍

用户端ASR结果。

参数

参数	类型	描述
text	const char*	ASR识别结果
final	bool	标记是否最终结果

返回

无

```
void (*onAIAgentSubtitle)(const char* text,  bool final);
```

介绍

AI智能体结果。

参数

参数	类型	描述
text	const char*	智能体结果
final	bool	标记是否最终结果

返回

无

```
void (*onAIAgentSpeaking)(bool speeking);
```

介绍

AI智能体TTS播放状态。

参数

参数	类型	描述
speeking	bool	true 远端音频播放 false 远端停止播放

返回

无

```
void (*onFunctionCall)(const char* id, const char* params);
```

介绍

functionCall回调。

参数

参数	类型	描述
id	const char*	单次function call唯一标识
params	const char*	参数 一般是json string 例如: {"function_name":"phone_call","parameter_list":[{"called_number":"1891017000"},{"called_name":"我的父亲"}

返回

无

```
void (*onAudioPlayerOp)(const char* path, bool start);
```

介绍

返回播放音频资源url。例如：小度音乐/故事

参数

参数	类型	描述
path	const char*	小度音频资源url
start	bool	true 开始播放， false 停止播放

返回

无

```
void (*onMediaSetup)(void);
```

介绍

智能体媒体链路已调通，会语音提示：我来了

参数

无

返回

无

```
void (*onAudioData)(const uint8_t* data, size_t len);
```

介绍

回调给应用层的TTS音频数据。

参数

参数	类型	描述
data	const uint8_t*	TTS音频数据
len	size_t	TTS音频数据长度

返回

无

```
void (*onVideoData)(const uint8_t* data, size_t len);
```

介绍

回调给应用层的视频数据。

参数

参数	类型	描述
data	const uint8_t*	视频数据
len	size_t	视频数据长度

返回

无

```
void (*onLicenseResult)(bool result);
```

介绍

设备license鉴权结果。

参数

参数	类型	描述
result	bool	true鉴权通过 false鉴权失败

返回

无

Websocket API

WebAPI 接口描述

大模型互动客户端API接口采用websocket协议的连接方式， 直接进行语音聊天。

主要流程

1. 连接
2. 连接成功后发送数据， 接收数据
3. 关闭连接
4. 名词解释：

- 连接： 这里指TCP连接及握手（Opening Handshake）， 一般WebSocket库已经封装，用户不必关心
- 发送数据帧：Sending Data Frame， 类似包的概念，指一次发送的内容 。从客户端到服务端。
  - 文本帧：Opcode 0x1 (Text)， 文本指令等
  - 二进制帧：Opcode 0x2 (Binary), 音频数据帧
- 接收数据帧： Receiving Data Frame， 类似包的概念，指一次发送的内容 。从服务端到客户端。
  - 文本帧：Opcode 0x1 (Text)， 识别结果和事件
  - 二进制帧：Opcode 0x2 (Binary)， 实时语音的回答数据
- 关闭连接：Closing Handshake。 关闭连接后就停止了对话

通常WebSocket库用需要用户自己定义下面的3个回调函数实现自己的业务逻辑。

```
 连接成功后的回调函数：
{
   #通常需要开启一个新线程，以避免阻塞无法接收数据
   1.1 实时发送音频数据帧
   1.2 发送文本
}

接收数据的回调函数
{
   2.1 实时返回音频数据帧
   2.2 实时返回文本消息
}

服务端关闭连接的回调函数
{
   3. 关闭客户端连接。
}
```

在线调试&示例代码

目前提供如下demo及演示功能.

语言	收发音频	采集音频	播放音频
javascript	Y	Y	Y
Go	Y		
C	Y		

webAPI SDK（javascript）

webAPI SDK（Go）

webAPI SDK（C）

请求说明

连接地址：wss://rtc-aiotgw.exp.bcelive.com/v1/realtime?a=XXXX&id=YYYY&t=ZZZZ&ac=pcmu

如果连接成功，一般WebSocket库会发起回调。

连接具体参数说明

参数	类型	选项	说明
a	string	必填	控制台上的大模型互动应用ID
id	int	必填	大模型互动实例 ai_agent_instance_id 值， 创建大模型互动实例API
t	string	必填	连接大模型互动实例的context.token值， 创建大模型互动实例API
ac	string	可选	音频格式，取值有raw，raw16k，pcmu, g722，opus; 默认是raw16k

发送音频数据帧

注意帧的类型（Opcode）是Binary

内容是二进制的音频内容。 除最后一个音频数据帧， 每个帧的音频数据长度为20-200ms。 建议最佳160ms一个帧，有限制的也建议80ms。

计算方式：

16000采样率： 1s音频 16000采样点

16bits： 一个采样点 16bits = 2 bytes

1s ： = 1000ms

即 160ms *16000* 2bytes / 1000ms = 5120bytes

实时语音识别api 建议实时发送音频，即每个160ms的帧之后，下一个音频数据帧需要间隔160ms。即：文件，此处需要sleep(160ms)。

发送文本对话信息

注意帧的类型（Opcode）是Text

示例：

[T]:你好！

返回说明

接收数据帧

正常音频数据

注意帧的类型（Opcode）是Binary

WebSocket opcode：binary

WebSocket消息体：音频二进制数据， 具体格式是由连接参数ac确定的

正常文本响应数据

```
 // 互动智能体事件
[E]:[AGENTID]:2233343116640256
[E]:[VOICE_COMING]
[E]:[TTS_BEGIN_SPEAKING]
[E]:[TTS_END_SPEAKING]

// ASR的返回结果
[Q]:[M]:上海
[Q]:[M]:上海有多少
[Q]:[M]:上海有多少人
[Q]:上海有多少人？

// 大模型回答结果
[A]:[M]:人口数量啊。
[A]:[M]:上海有
[A]:[M]:2487.4万人。
[A]:人口数量啊。上海有2487.4万人。
```

更多消息格式

客户端和智能体之间的消息格式

消息格式定义

数据来源	指令名称	前缀标识	示例	备注
服务端ASR	ASR结果 【question】	[Q]:	[Q]:你是谁？	
	ASR中间结果	[Q]:[M]:		
	ASR接续前面的完整问题	[Q]:[C]:		
	ASR中间识别结果 ，接续前面问题	[Q]:[M]:[C]:		
服务端LLM	LLM结果【answer】	[A]:	[A]:我是文小言	最终结果
	LLM中间结果	[A]:[M]:		中间结果
	LLM引导语（提示）	[A]:[H]:		引导语
事件消息	TTS开始播放【event】	[E]:	[E]:[TTS_BEGIN_SPEAKING]	
	TTS结束播放	[E]:	[E]:[TTS_END_SPEAKING]	
	声音到来	[E]:	[E]:[VOICE_COMING]	
	声音结束	[E]:	[E]:[VOICE_DISAPPER]	
	请求上传图片	[E]:	[E]:[UPLOAD_IMAGE]	请求上传图片
	播放音频文件	[E]:	[E]:[PLAY_AUDIO]	服务端下发音频文件 url，客户端播放该 url。格式如：[E]:[PLAY_AUDIO]:http://123.com/a.mp3
	请求上传图片	[E]:	[E]:[UPLOAD_IMAGE]	请求上传图片
	返回智能体应用的会话ID（AppBuilder,百炼,扣子,元宝,Dify等）	[E]:	[E]:[CONID]:c9c5ab24-269f-4c7d-9ce5-5f27fbe0d9b5	新建会话返回conversation_id

	设备授权事件	[E]:[LIC]:[MUST]:	[E]:[LIC]:[MUST]:设备必须鉴权	服务器发送给客户端
	设备激活消息	[E]:[LIC]:[ACTIVE]:	[E]:[LIC]:[ACTIVE]:{"devId":"DEVID-4000437447602844","uId":"4000902424829962","licKey":"xxx"}	客户端发送给服务器
	设备激活结果	[E]:[LIC]:[RES]:	[E]:[LIC]:[RES]:[PASS]:	授权成功，服务器发送给客户端
	设备激活结果	[E]:[LIC]:[RES]:	[E]:[LIC]:[RES]:[FAILED]:	授权失败，服务器发送给客户端
客户端	发送打断指令 【break】	[B]:		打断播报的内容
客户端	发送文本请求【text】	[T]:	[T]:你是谁？	发送文本请求
客户端	发送给智能体的TTS 直接播报	[TTS]:	[TTS]:欢迎光临。	不经过大模型， 直接播报出来
客户端	关闭智能体的自动打断功能	[SET]:[AUTO_INT]:[FALSE]		关闭自动打断功能
客户端	开启智能体的自动打断功能	[SET]:[AUTO_INT]:[TRUE]		开启自动打断功能，默认是开启的

HarmonyOS NEXT SDK

初始化接口

SDK初始化

```
let aiAgent = new AIAgentEngine(AIAgentEngineOption)
```

```
interface AIAgentEngineOption {
/** appid */
appId: string;
/** 实例id  服务器端下发，当前调试默认即可 */
aiAgentInstanceId: number;
}
```

SDK初始化

参数

参数	类型	描述
config	AIAgentEngineOption	初始化参数

返回

AIAgentEngine 实例对象：成功； null：失败；

AIAgent相关接口 开启通话

```
call(): void;
```

开启通话，开始音频采集和音频播放；

参数

无

返回

结束通话

```
hangup(): void;
```

结束通话，停止音频采集和音频播放；

参数

无

返回

无

发送文本

```
sendTextToAIAgent(text: string): void;
```

发送文本消息给智能体，作为query向大模型进行提问；

参数

参数	类型	描述
text	string	文本query

返回

无

发送文本并且打断

```
sendTextToAIAgentAndInterrupt(text: string): void;
```

发送文本消息给智能体，作为query向大模型进行提问，并且立即打断当前播报

参数

参数	类型	描述
text	string	文本query

返回

无

发送文本直接播报

```
sendTextToTTS(text: string): void;
```

发送文本消息给TTS模块直接进行播报

参数

参数	类型	描述
text	string	TTS播报文本

返回

无

主动打断

```
interrupt(): void;
```

打断当前播报内容，停止播放；

参数

无

返回

无

上传文件

```
uploadFile(path: string, expire: number): void;
```

参数

参数	类型	描述
path	string	文件路径，必须具有可读权限
expire	number	过期时间

返回

无

注意: 暂时仅仅支持 JPEG/PNG/JPG 三种数据格式，并且大小不超过7MB; *当发送视频流数据时,要在之前设置相机打开指令

相机开关指令

```
//如果上传视频流数据时,需要将相机指令打开
handleCamera(status: boolean): void;
```

参数

参数	类型	描述
status	boolean	true 开, false 关

返回

无

音频相关接口 静音播放

```
mutePlayback(isMute: boolean): void;
```

音频播放声音控制；

参数

参数	类型	描述
isMute	boolean	是否静音

返回

无

静音麦克风

```
muteMic(isMute: boolean): void;
```

音频采集声音控制；

参数

参数	类型	描述
isMute	boolean	是否静音

返回

无

事件回调 用户端监听事件

```
on(event: string, callback: (info: AIAgentResponse) => void): void;
```

用户端监听事件, 包含用户端ASR结果回调, AI智能体结果回调,functionCall回调

参数

参数	类型	描述
event	string	AIAgentEvent.UserAsrSubtitle 返回用户端ASR结果; AIAgentEvent.AIAgentSubtitle 返回AI智能体结果; AIAgentEvent.FunctionCall functionCall回调; AIAgentEvent.DataChannelMessage 服务端响应

返回

AIAgentResponse(用户端ASR结果 | AI智能体结果); ESObject(FunctionCall 结果 | AIAgentEvent.DataChannelMessage 服务端响应)

```
interface AIAgentResponse {
  text: string;
  isFinal: boolean;
  type: AIAgentType;
  mediaType?: AIAgentMediaTypeStatus;
  time?: string;
  id: string;
}
```

用户端ASR结果回调事例:

```
  this.aiAgent.on(AIAgentEvent.UserAsrSubtitle, (res: AIAgentResponse) => {
    console.log(`AIAgentResponse UserAsrSubtitle is ${JSON.stringify(res)}`)
  })
```

AI智能体结果回调事例:

```
  this.aiAgent.on(AIAgentEvent.AIAgentSubtitle, (res: AIAgentResponse) => {
    console.log(`AIAgentResponse instantId: ${this.instanceId} AIAgentSubtitle is ${JSON.stringify(res)}`)
  })
```

FunctionCall回调事例:

```
  this.aiAgent.on(AIAgentEvent.FunctionCall,(info:ESObject)=>{
    console.log(`FunctionCall is ${info}`)
  })
```

服务端响应事例

```
  this.aiAgent.on(AIAgentEvent.DataChannelMessage,(info:ESObject)=>{
      console.log(`DataChannelMessage is ${JSON.stringify(info)}`)
  })
```

服务端API

服务端API

总览 本章节介绍了百度云大模型互动OpenAPI的的REST风格接口，包括请求的组成部分，如何通过这些组成部分构造一个请求。您可以参看本章节，来构造HTTP请求调用对应的OpenAPI 。 **请求结构**

```
 // 请求
{HTTPMethod} {协议}://{serviceId}.baidubce.com/?请求参数
RequestHeader
RequestBody

================================================================================================
=

// 返回
{HTTP-Version} {Status-Code} {Reason-Phrase}
ResponseHeader
ResponseBody
```

名称	是否必选	描述	示例值
HTTPMethod	是	百度云请求 Method	POST
协议	是	百度云请求 支持通过HTTP或HTTPS协议进行请求通信。为了获得更高的安全性，推荐您使用HTTPS协议发送请求。	https
serviceId	是	服务名	rtc-aiagent
请求参数	否	请求参数	xx
RequestHeader	是	百度云OpenAPI 的公共请求头	authorization
RequestBody	否	接口自定义请求参数，每个 OpenAPI 自定义的请求参数。	app_id
ResponseHeader	是	接口返回结果的 header	x-bce-request-id
ResponseBody	否	接口返回的业务数据	ai_agent_instance_id

请求示例

```
 POST /api/v1/aiagent/stopAIAgentInstance  HTTP/1.1

/** RequestHeader **/
authorization:bce-auth-v1/ffab11fa2b5a4f698d159af2c23f556e/2024-06-06T03:00:20Z/1800/host/3fbe830c1e67389523e2ff09f47ec59c4ed723dcea22018569a3aec968cdfbe6
content-Type:application/json; charset=utf-8
x-bce-date:2015-04-27T08:23:49Z
host:rtc-aiagent.baidubce.com

/** RequestBody **/
{
   "业务请求参数": "json格式"
}

================================================================================================
=

/** ResponseHeader **/
HTTP/1.1 200 OK
x-bce-request-id: 1214cca7-4ad5-451d-9215-71cb844c0a50
date: Wed, 03 Dec 2014 06:42:19 GMT
content-Type: application/json;charset=UTF-8

/** ResponseBody **/
{
   "requestId":"1214cca7-4ad5-451d-9215-71cb844c0a50"
}
```

错误信息 所有错误除了使用HTTP状态码以外，应同时在内容中至少包含下表的参数。各服务应该适当地增加错误时返回的参数以方便定位。

参数名	类型	说明
requestId	String	导致该错误的requestId
code	String	字符串，用于表示具体错误类型
message	String	有关该错误的详细说明

示例：

```
{
  "requestId":"ae2225f7-1c2e-427a-a1ad-5413b762957d",
  "code":"AppParamNotValid",
  "message":"app not exist!"
}
```

创建大模型互动实例 接口描述 创建并启动大模型互动实例接口。

请求语法

```
POST /api/v{version}/aiagent/generateAIAgentCall HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: <bce-authorization-string>
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 名称 | 类型 | 是否必选 | 参数位置 | 描述 | | --- | --- | --- | --- | --- | | app_id | String | 是 | RequestBody | 大模型互动应用ID | | config | String | 否 | RequestBody | 大模型互动实例级别配置，格式是json对象序列化后的string类型 |

config中支持参数说明 | config中字段 | 类型 | 描述 | 示例值 | | --- | --- | --- | --- | | location | String | 用户位置信息 | 北京市海淀区 | | role | String | 简单角色 | “你是旅游达人” | | sceneRole | String | 复杂场景人设 | 实例复杂场景人设名称（需要对应用提前设置场景人设名称和对应的prompt） | | tts_sayhi | String | 招呼语 | 主动打招呼的话，“你好！” | | lang | String | 设置语言 | 枚举类型如下：enum {zh,en} | | audiocodec | String | 设置sdk音频格式 | 可选类型如下：raw（表示使用pcmu）、raw16k（表示使用g722） | | disable_voice_auto_int | bool | 关闭自动打断 | 设置为true则关闭自动打断 | | tts | String | 实例级别tts配置名称。支持内置类型和三方tts，三方类型tts按需需要传入对应的鉴权参数 | 目前支持DEFAULT（内置），VOLC, XUNFEI, BAIDU | | tts_url | String | 实例级别tts配置详情，格式是以tts名称拼接tts详细配置json 对象序列化的string | {"tts_url": "DEFAULT{\"vcn\": \"111\",\"vol\": 2.0,\"spd\": 1.0}"} | | llm | String | 实例级别大模型配置，支持内置模型（无需传参数）和三方模型（OPENAI协议、百度AB、阿里百炼、腾讯元宝、扣子、Dify等。参见：第三方服务集成步骤指引） | | | llm_url | String | 实例级别大模型配置，支持内置模型（无需传参数）和三方模型（OPENAI协议、百度AB、阿里百炼、腾讯元宝、扣子、Dify等。参见：第三方服务集成步骤指引） | | | llm_cfg | String | 实例级别大模型配置，支持内置模型（无需传参数）和三方模型（OPENAI协议、百度AB、阿里百炼、腾讯元宝、扣子、Dify等。参见：第三方服务集成步骤指引） | | | llm_token | String | 实例级别大模型配置，支持内置模型（无需传参数）和三方模型（OPENAI协议、百度AB、阿里百炼、腾讯元宝、扣子、Dify等。参见：第三方服务集成步骤指引） | |

tts_url中支持的参数配置 | tts_url中字段 | 类型 | 描述 | 示例值 | | --- | --- | --- | --- | | vcn | String | 实例级别tts发音人 | test_voice | | vol | Double | 实例级别tts音量配置，内置tts类型范围[0.5~2.0]表示基础声音的倍数，三方tts使用对应配置参数 | 0.5 | | spd | Double | 实例级别tts音速配置，内置tts类型范围[0.5~2.0]表示基础音速的倍数，三方tts使用对应配置参数 | 0.5 | | apid | String | 实例级别tts外部账号id配置。type为DEFAULT时非必要，其余按需 | your_apid | | apikey | String | 实例级别tts外部账号鉴权key、token等。type为DEFAULT时非必要，其余按需 | your_apikey | | apisk | String | 实例级别tts外部账号sk配置。type为DEFAULT时非必要，其余按需 | your_apisk |

请求示例

```
POST /api/v1/aiagent/generateAIAgentCall HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "app_id": "yourAppId",
    "config": "{\"llm\" : \"\", \"llm_token\" : \"no\", \"lang\" : \"zh\", \"tts\": \"DEFAULT\",
\"tts_url\":\"DEFAULT{\\\"vcn\\\": \\\"4103\\\",\\\"vol\\\": 0.1,\\\"spd\\\": 1.1, \\\"apid\\\": \\\"\\\",
\\\"apikey\\\": \\\"\\\"}\"}"
    }
}
```

响应头域 除公共头域外，无其它特殊头域。

响应参数

参数名	类型	描述
ai_agent_instance_id	long	大模型互动实例ID
instance_type	String	大模型互动实例类型
context	Object	大模型互动实例上下文
+cid	Long	大模型互动实例sdk使用数据通信ID
+token	String	大模型互动实例sdk使用数据通信token

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "ai_agent_instance_id": 222,
    "instance_type": "VoiceChat",
    "context": {
        "cid": 1,
        "token": "xxx"
    }
}
```

错误响应码 | Code | Message | HTTP Status Code | 说明 | | --- | --- | --- | --- | | AppParamNotValid | app not exist! | 400 | 大模型应用不存在 | | AIAgentInstanceIdGenerateException | generate ai agent instance id error | 400 | 创建大模型互动实例ID异常 | | AIProxyCreateInstanceFailedException | create ai proxy instance failed | 400 | 启动大模型互动实例异常 |

停止大模型互动实例 接口描述 停止大模型互动实例接口。

请求语法

```
POST /api/v{version}/aiagent/stopAIAgentInstance HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: <bce-authorization-string>
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 名称 | 类型 | 是否必选 | 参数位置 | 描述 | | --- | --- | --- | --- | --- | | app_id | String | 是 | RequestBody | 大模型互动应用ID | | ai_agent_instance_id | Long | 是 | RequestBody | 大模型互动实例ID |

请求示例

```
 POST /api/v1/aiagent/stopAIAgentInstance  HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "app_id": "yourAppId",
    "ai_agent_instance_id": 123456
}
```

响应头域 除公共头域外，无其它特殊头域。

响应参数 N/A

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

服务端打断大模型互动实例播报 接口描述 打断大模型互动实例播报

请求语法

```
 POST /api/v{version}/aiagent/interrupt HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: <bce-authorization-string>
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 名称 | 类型 | 是否必选 | 参数位置 | 描述 | | --- | --- | --- | --- | --- | | app_id | String | 是 | RequestBody | 大模型互动应用ID | | ai_agent_instance_id | Long | 是 | RequestBody | 大模型互动实例ID | | extra_msg | String | 否 | RequestBody | 打断时携带的播报消息 |

请求示例

```
 POST /api/v1/aiagent/interrupt  HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "app_id": "yourAppId",
    "ai_agent_instance_id": 123456,
    "extra_msg": "打断后播报您好"
}
```

响应头域 除公共头域外，无其它特殊头域。

响应参数 N/A

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

错误响应码 | Code | Message | HTTP Status Code | 说明 | | --- | --- | --- | --- | | AIProxyInstanceNotExistException | ai proxy instance not exist | 400 | 大模型互动实例不存在 | **大模型互动实例发消息给sdk 接口描述** 服务端提供接口用来发送指定消息给sdk

请求语法

```
 POST /api/v{version}/aiagent/sendMsg HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: <bce-authorization-string>
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 名称 | 类型 | 是否必选 | 参数位置 | 描述 | | --- | --- | --- | --- | --- | | app_id | String | 是 | RequestBody | 大模型互动应用ID | | ai_agent_instance_id | Long | 是 | RequestBody | 大模型互动实例ID | | message | String | 是 | RequestBody | 自定义消息 |

请求示例

```
 POST /api/v1/aiagent/sendMsg  HTTP/1.1
host: rtc-aiagent.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "app_id": "yourAppId",
    "ai_agent_instance_id": 123456,
    "message": "发送给sdk的消息内容"
}
```

响应头域 除公共头域外，无其它特殊头域。

响应参数 N/A

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

错误响应码 | Code | Message | HTTP Status Code | 说明 | | --- | --- | --- | --- | | AIProxyInstanceNotExistException | ai proxy instance not exist | 400 | 大模型互动实例不存在 |

第三方服务开放协议

开放服务协议配置

开放协议OPENAPI配置说明

创建智能体的配置信息如下： {"llm":"LLMOpenAPI", "llm_url":"http://", "llm_cfg":"user data", "llm_token": "$API-KEY"}

配置字段详解

字段名称	解释	参考值
llm	模型类别	使用固定值: LLMOpenAPI
llm_url	模型服务器地址	http://
llm_cfg	用户自定义值	比如userid 之类的信息
llm_token	用户鉴权的token	传给http的请求头Token

OPENAPI 协议携带sessionId，作为一次会话的标识, 系统自动生成传给模型服务器；

需要实现如下服务接口的协议:

```
 curl --location 'http://10.12.186.247:8810/openapi/v1/chat?appid=AppIdXXX' \
--header 'Content-Type: application/json' \
--header 'Token: llm_token' \
--data '{
    "sessionId": "123332sdfeasfesfe13221",
    "messages": [{"role": "user", "content": "宫廷玉液酒怎么卖？"}],
    "type":"baidu",
    "cfg": "user data"
}'
SSE流式返回:
data:
data:宫廷
data:玉
data:液
data:酒
data:是一种
data:高档
data:的
data:白酒
data:，
data:如果
data:对
data:产品
data:有
data:疑问
data:，
data:可以
data:咨询
data:专业人士
data:或
data:联系
data:品牌
data:官方
data:进行
data:确认
data:。
data:
```

下载专区

大模型实时互动 SDK&Demo下载

下载SDK

本文档提供最新版本的 SDK。如果您需要老版本的 SDK，请查看 历史版本记录。

平台	更新时间	更新内容	SDK文件名	下载
Android	2025.4.25	正式版本发布	Android.ChatAgent.v1.0.26.zip	点击下载
iOS	2025.6.10	新增功能 & BUG修复： • 添加 presetLoudSpeaker 接口，提供预设音频输出设备能力 • 修复连接蓝牙设备后加入会话，仍然从扬声器播放问题 • 修复蓝牙设备与扬声器切换时可能发生的变声问题	BaiduChatAgent_V1.3.0.zip	点击下载
H5	2025.4.25	正式版本发布	BRTC.Agent.H5.SDK.V1.0.4.zip	点击下载
微信小程序	2025.4.25	正式版本发布	BRTC.Agent.MiniApp.SDK.V1.0.5.zip	点击下载
HarmonyOS NEXT	2025.7.01	正式版本发布	BaiduChatAgent_V1.0.0.zip	点击下载
RTOS-乐鑫 esp32-s3	2025.6.9	新增功能 & BUG修复： • 修复了播放长文章或者长故事崩溃问题 • 解决了应用层收不到onFunctionCall回调的问题 • 新增了baidu_chat_agent_engine_send_text_to_TTS接口 • 修改了baidu_chat_agent_engine_send_text传参的逻辑 • 新增了应用层onCallStateChangeCallback处理agent登录失败或者断网重连的参考示例	BRTC.RTOS.SDK.ESP32S3.V3.0.8B06	点击下载
RTOS-杰理 AC7911BA	2025.6.19	杰理AC7911BA正式版本发布	BRTC.RTOS.SDK_JL1.2.V3.0.8B07.BA-20250619.tar.gz	点击下载
RTOS-杰理 AC7911BB	2025.6.19	功能优化： • toyapp修复crash和内存使用	BRTC.RTOS.SDK_JL1.2.V3.0.8B07-20250619.tar.gz	点击下载
RTOS-BK7258	2025.6.23	功能优化： • 优化栈内存使用修复crash问题 • 修复音频播放的卡的问题	BRTC.RTOS.SDK.BK7258.V3.0.8B017.tar.gz	点击下载
Websocket API	2025.5.23	增加C语言SDK	Websocket Demo	js点击下载 Go点击下载 C点击下载

下载进阶功能Demo

进阶功能 Demo 是 大模型实时互动 提供的高级功能的开源示例工程文件。

平台	更新时间	更新内容	SDK文件名	下载
RTOS-杰理AC7911BA	2025.6.19	功能优化： • 修复音乐播放相关的问题 • 修复内存问题导致的crash	TOY_APP_JL1.2_V3.0.8B07_BA-20250619.tar.gz	点击下载
RTOS-杰理AC7911BB	2025.6.19	功能优化： • 修复音乐播放相关的问题 • 修复内存问题导致的crash	TOY_APP_JL1.2_V3.0.8B07-20250619.tar.gz	点击下载

Demo体验

平台	下载地址
Web Demo-语音互动	点击体验
Web Demo-数字人互动	点击体验
安卓端Demo	
IOS端Demo	

历史版本记录

大模型实时互动保留近一年的历史版本，建议接入最新版本SDK。

iOS

平台	发布时间	更新内容	SDK文件名	下载
iOS	2025.6.3	新增功能 & 功能优化： • 支持地图推荐互动能力 • 修复音频模式下权限过度申请问题	BaiduChatAgent_V1.2.9.zip	点击下载
iOS	2025.4.25	正式版本发布	BaiduChatAgent_V1.2.8.zip	点击下载

RTOS

RTOS-乐鑫esp32-s3

平台	发布时间	更新内容	SDK文件名	下载
RTOS-乐鑫esp32-s3	2025.5.22	功能优化： • 增加了rtos sdk nack抗弱网功能 • 优化了底层播放的逻辑，提升了抗弱网能力，抗丢包率20%左右 • 优化了rtos sdk多个task系统调度的逻辑 • 优化了系统配置sdkconfig.defaults.esp32s3	BRTC.RTOS.SDK.ESP32S3.V3.0.8B05	点击下载
RTOS-乐鑫esp32-s3	2025.5.16	新功能： • 增加了G722 16k编解码功能 • 增加了license鉴权的功能 • 优化了正常网络下播放声音不清晰的问题。	BRTC.RTOS.SDK.ESP32S3.V3.0.8B04.tmp20250516	点击下载
RTOS-乐鑫esp32-s3	2025.4.25	全新版本发布	BRTC.RTOS.SDK.ESP32S3.V3.0.8B02	点击下载

RTOS-杰理AC7911BB

平台	发布时间	更新内容	SDK文件名	下载
RTOS-杰理AC7911BB	2025.5.16	新增功能： • 增加了rtos sdk nack抗弱网功能	BRTC.RTOS.SDK_JL1.2_V1.1.9-0513.tar.gz	点击下载

RTOS-BK7258

平台	发布时间	更新内容	SDK文件名	下载
RTOS-BK7258	2025.5.23	正式版本发布： • 优化稳定性和卡音问题 • 支持播放端弱网nack • 新增鉴权功能	BRTC.RTOS.SDK.BK7258.V3.0.8B05.tar.gz	点击下载

最佳实践

构建地图能力智能互动应用

概览

用户在与大模型实时互动，进行语音对话时，可以实现 查询导航信息、发起导航、搜索周边点位 等和地图能力相关的问询，并通过客户端地图交互，完成对应需求

需求场景

地图导航

查询前往目的地的规划路线，发起实时导航，并通过智能互动语音播报导航信息。

互动示例：

输入	特征
导航去香山	明确要去哪里
我要开车去香山	当前位置直接驾车导航到目的地
从香山去北京站怎么走	自动识别导航类型、自定义出发地
我要去故宫，中间先去西直门	带途经点导航

响应交互：给出优先规划结果，可跳转至路线规划页面，并选择路线、发起导航

> 你可以根据互动回调数据，自行设计响应交互流程及视觉，地图推荐场景亦是如此

地图推荐

查询当前位置或指定地点周边POI点位。

互动示例：

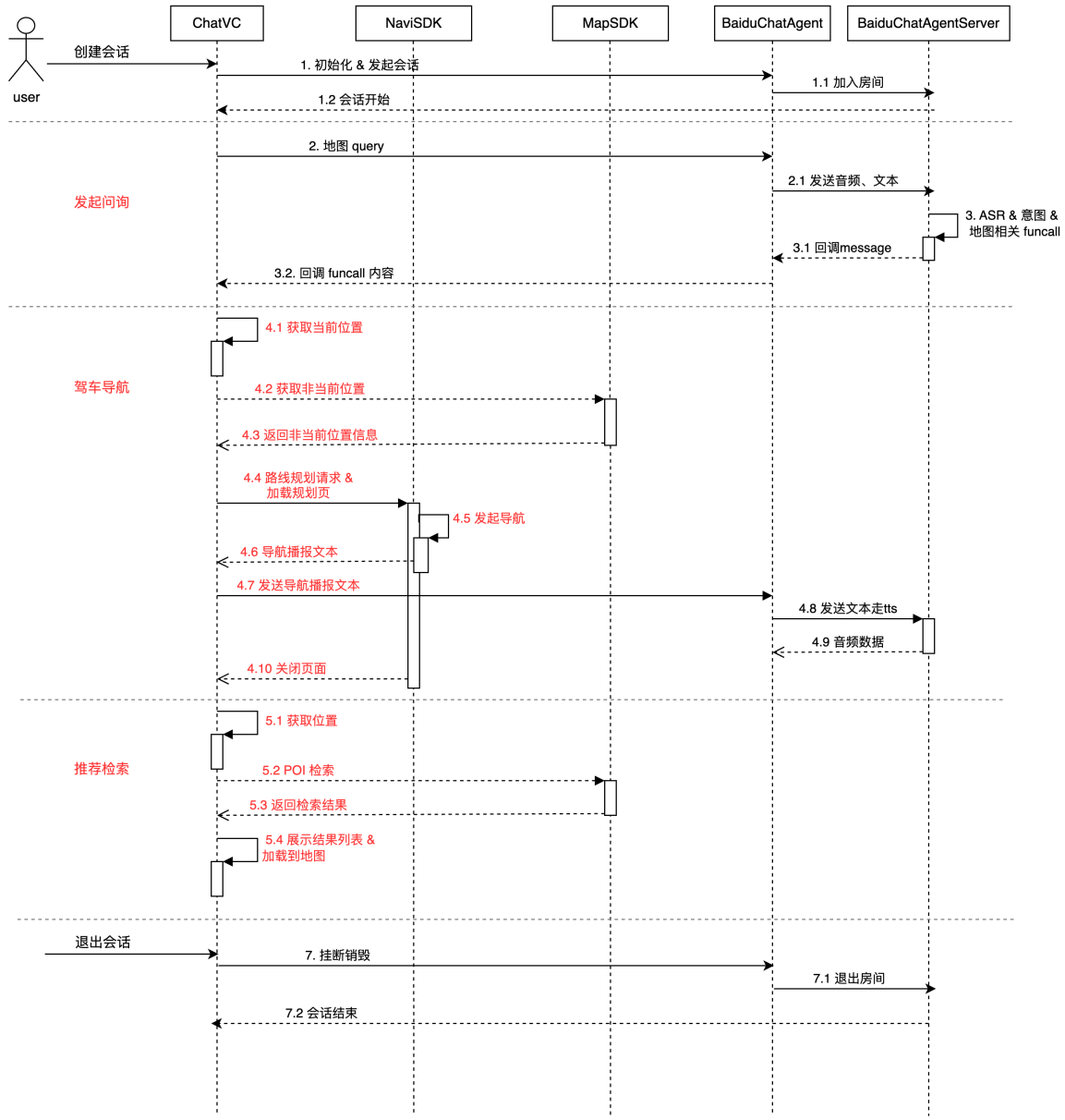
输入	特征
推荐下附近的小吃	当前位置周边
最近的健身房在哪	强调首位，直接展示详情
帮我找找前门附近的酒店	目的地周边
1公里内有哪些公园	限制检索范围
搜索北京的图书馆	行政区范围内检索

方案概述

依赖资源

- BaiduChatAgent SDK：支持 function call 回调能力的百度智能云大模型实时互动SDK
- 地图SDK & 地图导航SDK：百度地图开放平台提供的客户端SDK

功能时序



- 发起问询：
  - 与大模型互动过程中，BaiduChatAgent 根据输入内容，识别为地图类型问询，解析参数后通过接口回调给客户端
- 导航
  - 4.1 & 4.2 回调参数中的各个地点，通过 地图SDK POI 检索，获取定位信息，当前位置用系统方法获取
  - 4.4 利用 导航SDK 请求导航数据，打开新页面，加载路线规划结果视图
  - 4.5 在规划页内发起导航、及导航进行交互
  - 4.6~4.9 导航SDK 将导航播报文本回调给业务，用户调用 Agent 接口直接发送给 TTS 进行播报
  - 4.10 导航交互完成后，用户关闭页面回到 Agent 交互流程
- 推荐
  - 5.1 和导航一样，非当前位置的地点，通过 地图SDK POI 检索，获取定位信息；当前位置用系统方法获取
  - 5.2~5.3 调用 地图SDK 请求检索结果
  - 5.4 自定义列表视图，可在打开独立页面，加载地图底图后叠加展示

回调定义

> 大模型识别到用户输入为特定需求时，会通过 onFunctionCall 接口回调函数参数， 本节内容为大模型识别为地图类型问询后，回调给客户端的地图函数调用内容结构 接口回调 params 参数 json 字符串格式示例：

```
"{\"session_id\":\"1744718758902\",\"content\":\"
{\\\"function_name\\\":\\\"take_photo\\\",\\\"parameter_list\\\":[]}\"}"
```

以下定义说明为 content 中的内容

地图导航

- 函数名

map_navigate

- 参数列表

参数名	类型	是否必填	描述	值
start_position	string	否	导航起点	
navi_position	string	是	导航目的地	
navi_type	string	否	导航类型	[drive(驾车)、bike(骑行)、walk(步行)]

- content 示例

```
{
  "function_name": "map_navigate",
  "parameter_list": [
     {"start_position": "奥体中心"},
     {"navi_position": "天安门"},
     {"navi_type": "drive"}
  ]
}
```

地图推荐

- 函数名

map_poi_search

- 参数列表

参数名	类型	是否必填	描述	值
search_type	string	否	检索类型	[area(城市或行政区)、nearby(周边)]
center_position	string	否	检索中心位置	
radius	number	否	检索半径，单位为米	默认 1000
keyword	string	是	检索关键词	[drive(驾车)、bike(骑行)、walk(步行)]

- content 示例

```
{
  "function_name": "map_poi_search",
  "parameter_list": [
    { "search_type": "nearby" },
    { "center_position": "鸟巢" },
    { "keyword": "餐厅" }
  ]
}
```

能力接入

地图能力配置

地图智能互动依赖百度地图开放平台的SDK能力，需要先在其平台上注册账号并创建应用，具体步骤如下：

- 前往百度地图开放平台，参考 流程指引 完成账号注册，并获取密钥（AK）
- 进入百度地图开放平台 控制台 ，参考 操作流程 创建应用，勾选所需服务，iOS 填写 bundleid；Android 填写包名、开发&测试版本 keysrore SHA1
- 进入百度地图开放平台 开发文档，按照下面的平台接入流程，下载对应资源进行集成

iOS 接入

Demo 体验

你可以直接通过 BaiduChatAgent Demo 体验地图智能互动能力

- 将地图相关 SDK 放在 BaiduChatAgent 的指定目录内，具体结构如下：

```
demo
  └──── ChatAgentApp/
sdk
  └──── Third
      ├──── MapSDK
      │    ├──── BaiduMapAPI_Base.framework
      │    ├──── BaiduMapAPI_Map.framework
      │    ├──── BaiduMapAPI_Search.framework
      │    ├──── BaiduMapAPI_Utils.framework
      │    └──── thirdlibs
      │         ├──── libcrypto.a
      │         └──── libssl.a
      └──── NaviSDK
          ├──── libbaiduNaviOpenSDK.a
          ├──── NaviResource
          │    ├──── baiduNaviSDK.bundle
          │    └──── mode.bundle
          └──── inc
```

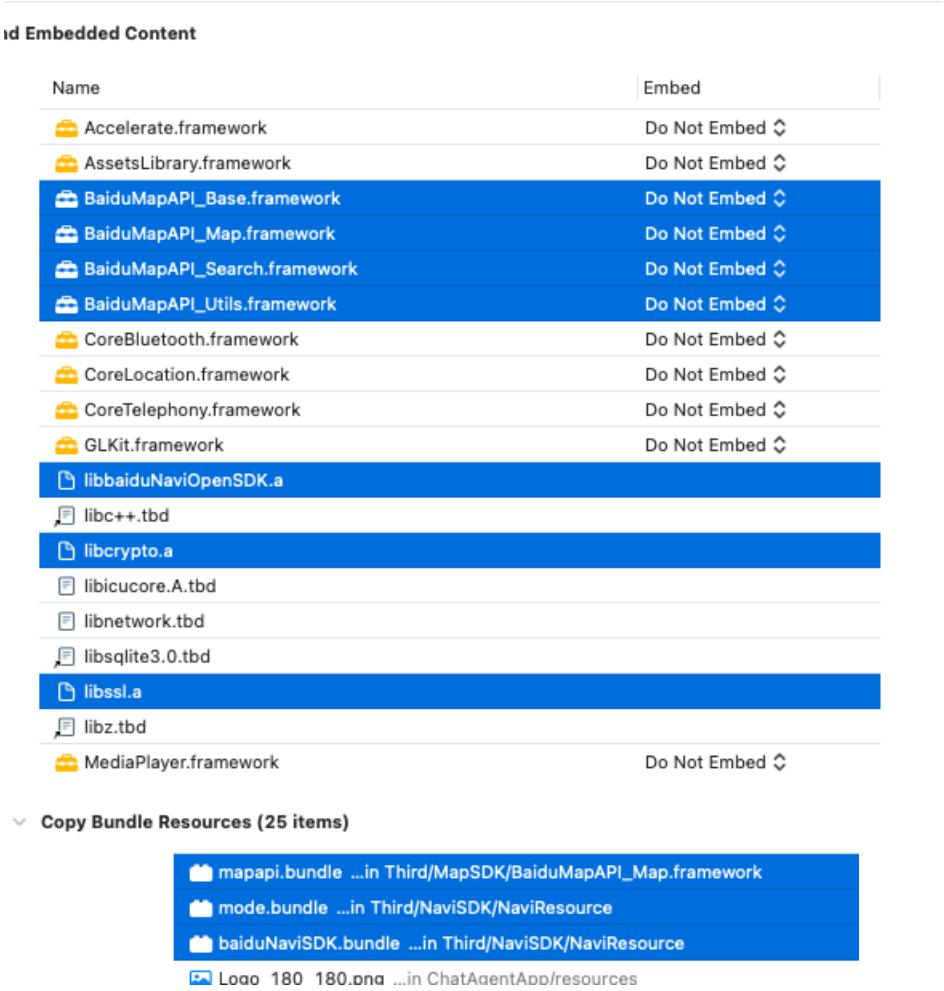
- 打开 ChatAgentApp/ChatAgentApp/ios/funcall/MapNavigate/BNaviUtil.h 文件，修改 BNAVI_APP_KEY 为你应用的 AK
- 编译 ChatAgentApp 运行 Demo，在会话中与大模型进行对话

SDK 集成

为了构建你自己的带有地图只能互动能力得应用，需要先按照以下步骤完成必要的能力集成：

- 添加 BaiduChatAgent SDK 到你的工程
- 在 iOS导航SDK 页面下载 iOS导航SDK资源包 和 隐私清单文件

- 添加地图相关SDK到你的工程



- 参考 ChatAgentApp Demo 的

构建地图交互

为了实现地图交互，主要需要完成以下几个部分：

- 地图能力的初始化及鉴权：
  - 参考 ChatAgentApp Demo 的 BNaviUtil 类的实现，并填充你自己的地图账号AK；
- 地理位置更新：
  - 参考 BCALocationManager，或使用地图SDK的定位能力，在需要时更新当前位置
- function call 回调对接：
  - 参考 ChatAgentApp Demo 的 BCAFuncallHandler 类的实现，完成对函数参数解析和分发
  - 参考 BCAMapNavigateHandler 和 BCAMapPoiSearchHandler， 完成对地图函数的处理逻辑
- 地图SDK能力交互、地图请求回调数据处理等
  - 参考 BCAMapNavigateDriveModel，完成请求地图导航信息
  - 参考 BCAMapNaviDriveViewController 和关联代码，完成路线规划数据展示与交互
  - 参考 BCAMapPoiSearchViewController，完成基础地图视图呈现、添加POI点位信息

实时音视频 RTC

动态与公告

功能发布记录

发布时间	功能分类	功能描述
2024-02	新功能 功能优化	服务端 • 新增云播放器功能，支持向RTC房间输入在线媒体流，详见 创建云播放器 。 Android & iOS • 外部视频编码支持H.265编码格式，详见 外部编码 。 • 新增发送和接收SEI消息功能，详见 发送SEI消息 。 控制台 • 新增云端录制回调认证功能，提升服务的安全性和可维护性，详见 通知（回调）验证 。 • 增加录制文件存储所属地域，与BOS保持一致。 • 优化应用管理模块部分功能的提示信息。
2024-01	新功能 功能优化	Android & iOS • 视频编码支持JPEG编码格式。 iOS • 新增SDK内部断网重连机制，提升SDK稳定性。 • 优化首屏时长。 • 优化部分接口。
2023-12	新功能 功能优化	服务端 • 优化IOT平台抗弱网能力，信令短连接模式达到 70%抗弱网效果。 • 支持IOT平台信令和媒体异或加密，增强传输链路安全性。 Flutter • 发布Flutter SDK，详见 Flutter SDK 。
2023-11	新功能 功能优化 BUG修复	服务端 • SDK转推支持控制重启次数，自动录制支持重启，提升混流稳定性。 • 优化混流调度逻辑，使大核小核机器负载更加均衡，同时避免峰值请求导致调度异常。 • 支持推拉流通过rtmps加密。 Web • 修复新版本浏览器上调用stat 报错的问题。
2023-08	新功能	服务端 • 云端录制和云端转推支持流级别的水印功能，详见 WatermarkConfig结构。 • 新增房间挂起和恢复功能。

2023-07	新功能	控制台 • 新增云端录制回调地址配置，详见 云端录制回调 。
2023-04	新功能	Android & iOS • 新增信令支持QUIC协议传输。 • 默认信令精简模式。
2023-03	新功能	新增本地合流录制等功能。
2023-01	新功能	新增视频截图、音量提示回调、上下行音视频策略配置等功能。
2022-12	新功能	支持设置预览分辨率、音频采集及播放增益、支持播放MP3文件等功能。
2022-10	新功能	发布服务端 API，提供房间管理和用户管理相关接口；新增实时水印、背景分割、本地录制等功能。
2022-09	新功能	屏幕分享支持调节参数及系统音频，补充灵活订阅流信息、支持通话前预览等。
2022-07	新功能	实时通话支持变声能力，包括萝莉、正太等。
2022-04	新功能	• 发布Android 纯音频SDK，提供更低包体积，更低功耗版本，适用于物联网产品。 • 发布多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限。
2021-09	新功能	发布FreeRtos sdk V0.0.1。RTC FreeRtos SDK能够帮助您快速集成RTC能力。 通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。

产品简介

概述

实时音视频RTC（Real-Time Communication）是百度智能云提供的实时音视频通信PaaS平台，依托百度智能云强大的流媒体处理与传输能力、覆盖全球的低延时网络，提供稳定、超低延时、高质量的实时音视频通信服务。帮助您快速搭建多端的实时通信应用，应用于互动娱乐、在线教育、视频客服、视频会议、物联网视频通信等场景。

核心概念

实时音视频RTC

- 应用：是RTC的基础业务单元。每个应用有唯一的AppID，不同应用之间无法进行通信。每个用户可以创建多个应用，每个应用内可创建多个房间，开启鉴权的房间可在控制台获取AppKey。
- 房间：实时音视频的基础通信单元，加入到一个房间内的用户能够互相进行音视频通信。如一个会议、通话、课堂。
- Room Name：房间名称，用户加入房间时输入，用户指定并维护，保证唯一性。
- UiD：用户ID，整数类型，用户的唯一身份标识，由客户业务系统生成并保证唯一性。
- 发布：一个用户将自己的音频、视频、屏幕分享发送出去的动作。
- 订阅：一个用户观看、收听他人音视频的动作。

功能特性

实时音视频RTC功能

功能	说明	场景
语音通话	1对1或多人语音通话，支持48kHz音频采样率，实现高音质通话	语聊房、语音会议、狼人杀
视频通话	1对1或多人视频通话，最高可支持4K高清画质	1对1视频通话、视频会议、视频客服、视频查勘
网络测速	在开始通话前可进行网络质量探测，检测上行和下行网络带宽、丢包、网络抖动和往返时延等网络质量数据	视频通话、视频会议
连麦互动	支持主播与观众连麦互动，观众可平滑上下麦	语聊房、互动直播
跨房间PK	支持主播跨房间PK连麦，主播延时小于300ms	互动直播
云端录制	将RTC房间内用户的音视频进行单路或混流录制，支持设置自定义布局和录制水印，并将录制的文件存储到BOS平台	音视频双录、在线课堂、视频会议
旁路转推	将RTC房间内用户的音视频进行混流转码，并以RTMP等协议转推到CDN直播服务	数字人直播、电商直播
云播放器	支持向RTC房间输入在线媒体流，房间内其他用户可以观看该媒体流，输入源支持直播流和文件类型	语聊房、在线课堂、互动直播
自定义音频采集	支持自己采集外部音频，开发者可以对原始数据进行处理，进行自定义操作	非标设备接入、语音处理
自定义视频采集	支持自定义的视频源和渲染器，例如视频文件、外接设备等	外接摄像头、增加美颜特效
屏幕共享	支持将电脑桌面、窗口共享给其他人	视频会议、在线课堂
视频水印	支持在通话前或通话中开启视频水印，可以设置水印位置、图片、文字大小和颜色	版权声明
截屏	支持对本地或远端视频画面截屏保存	画面采集
AI降噪	AI 降噪可以消除平稳噪音和非平稳噪音，如键盘声、咳嗽声、汽车鸣笛声等	语聊房、视频会议
媒体补充增强信息（SEI）	SEI 信息跟随音视频帧发送，实现 SEI 内容与音视频内容精准同步	互动直播
实时消息与信令	房间内用户向房间中发送信令消息，房间内其他用户收到消息后，可实现自身业务逻辑处理	远端控制、平行驾驶
基础美颜	支持基础的美颜功能，包括设置美白、磨皮、大眼、瘦脸	互动直播、视频会议
变声特效	变声特效可以对人声进行处理，实现不同的声音效果，如萝莉、大叔等	匿名通话
背景分割	支持通过 AI 算法将人物与背景分割，支持模糊背景，自定义图片等	视频会议
超分	支持将低分辨率视频超分成高分辨率视频，提高视频清晰度和画质	互动直播 视频通话

平台支持

平台	环境要求
Andriod	Android 5.0以上
iOS	ios 9.0 及以上
Web	Chrome 108+ Firefox 56+ Safari 11+ Edge 80+ 360安全浏览器 21+

应用场景

实时音视频RTC

视频会议

提供稳定流畅的实时音视频通信服务，满足多人视频会议需求，适用于内部会议、员工培训、远程协作等场景，低成本构建视频会议系统。

在线教育

满足1对1、小班课、直播课堂互动答疑等在线教育场景，提供稳定、高清的实时音视频通信服务，支持服务端同步课程录制，轻松实现师生之间的互动，提升教学体验。

互动社交

支持多种社交场景的互动需求，1v1在线语音聊天、视频聊天，多人语音、视频社交场景。与百度智能云音视频直播LSS相结合，满足直播连麦、主播PK等场景。

视频客服

应用于金融、保险等行业，满足远程业务办理、现场服务等场景需求，提供流畅、高可用的实时音视频通信服务，提升业务办理效率，减少时间人力成本，支持服务端同步录制。

智能硬件

应用于智能硬件，支撑设备与人、控制端的实时通信服务，提供超低延时的实时音视频通信服务，满足设备远程查看、控制、通信等需求。

产品优势

实时音视频RTC

高音质

业内领先的 3A 算法，支持 48kHz 采样的高音质，AI 降噪算法能识别多种场景噪声，可在嘈杂的环境下有效消除噪声，无回声、无啸叫，保持清晰流畅的纯净人声，实现沉浸式互动通话体验

高画质

支持H.264、H.265编码方式，提供视频超分、画质增强、背景分割、基础美颜、视频水印、截图等多种视频处理能力，最高可支持4K超高画质，为用户提供极佳的画质体验

超低延时

依托百度智能云强大的实时音视频处理与传输能力、覆盖全球的低延时网络，在全球范围内提供稳定高质量的实时音视频服务，基于用户距离、节点质量、网络质量探测，自动配置最佳路由，保障用户就近接入，端到端延时低至300毫秒

弱网优化

自研抗弱网算法策略，在弱网丢包恢复中，NACK会和FEC结合使用，针对不同网络环境匹配不同策略，动态调整NACK和FEC比例，平衡带宽和延时，实时精准估计网络状态、动态调整码率。支持多卡聚合传输，在网络信号波动或断网时，有效解决不同位置单一运营商网络覆盖和质量不可靠的问题，提高带宽传输上限，通过实时监测网络变化和连通性，自动选择最优运营商网络通道进行实时音视频传输，音视频抗丢包达70%，抗网络抖动达1000ms

IoT设备适配

针对手表、门锁、无人车等多种物联网设备，可提供轻量级、低功耗的SDK，最小包体<300K，CPU最低运算速度<300MHz，主体程序运行内存<2M，已适配ASR、乐鑫、展锐等多款主流芯片，优化超分算法提升低分辨率视频的清晰度，满足用户清晰流畅的通话体验

全平台互通

提供全平台覆盖的实时音视频SDK及服务端RESTful API，实现跨平台音视频通话，支持Android、IOS、Web、Windows、MacOS、微信小程序、Flutter、Linux、Rtos等多平台全球互通

产品计费

音视频通话计费

本文介绍 RTC 音视频通话计费规则。

计费规则

使用 RTC 服务时，您需要根据音视频规格和用量为音视频通话付费，详情请参见实时音视频RTC价格说明。

- 计费项：音视频通话时长
- 付费方式：后付费
- 付费周期：按小时扣费，即北京时间整点扣费并生成账单。出账单时间是当前计费周期结束后1小时内。例如，10:00-11:00的账单会在12:00之前生成，具体以系统出账时间为准
- 开通服务前需保证账户无欠款

计费公式

音视频通话费用 = 音频通话时长 × 音频单价 + 视频各分辨率档位通话时长 × 相应视频分辨率档位单价

计费说明

- 通话时长=每个用户订阅音频、视频时长的总和
- 按分钟计费，不足1分钟按1分钟计
- 媒体规格根据订阅媒体流的真实数据规格统计（订阅指观看/收听动作）
- 订阅视频数据时只收取视频费用，其中音频不再重复计费

云端录制计费

本文介绍 RTC 云端录制计费规则。

计费规则

自2023年5月1日起，使用 RTC 服务时，如果您还使用了云端录制功能，需要为云端录制功能额外付费，详情请参见实时音视频RTC价格说明。

- 计费项：云端录制时长
- 付费方式：后付费
- 付费周期：按小时扣费，即北京时间整点扣费并生成账单。出账单时间是当前计费周期结束后1小时内。例如，10:00-11:00的账单会在12:00之前生成，具体以系统出账时间为准
- 开通服务前需保证账户无欠款

计费公式

云端录制费用 = 录制音频费用 + 录制视频费用 = 录制音频输入时长 × 单路或多路对应的音频单价 + 录制视频各分辨率档位输入时长 × 单路或多路对应的相应视频分辨率档位单价

计费说明

- 按分钟计费，不足1分钟按1分钟计

- **云端录制时长**，按同一账号下所有应用使用云端录制时录制服务每路流的输入时长来统计云端录制服务的用量。输入时长根据云端录制服务接收音视频流类型，分为**视频录制时长**和**音频录制时长**，**视频录制时长有不同分辨率区分**。
- 录制分为**单流录制**与**合流录制**。单流录制是房间中个每一个用户的音视频都录制成一个独立的文件，按单流对应单价计费。合流录制是将房间中多个用户的音视频合并录制成一个文件，每路用户输入流按合流对应单价计费。
- 合流录制下，**如果所有流均为纯音频，则按照集成音频的方式计费**，即只按照一路集成音频计算音频时长，同时按照合流录制的音频单价计费。
- 录制期间，单路输入流既有视频又有音频时，只按视频时长统计，不会重复计算音频时长。
- 录制期间，多路输入流中有纯音频流时，也需要计算该路流的音频时长。
- 录制过程中，突然中断用户上行，在配置的“等待续录时间”内按音频时长录制计费。

云端转码计费

本文介绍 RTC 云端转码计费规则。

计费规则

自2023年5月1日起，使用 RTC 服务时，如果您还使用了云端转码功能（将各路上行音视频流进行画面混合、音视频转码、转封装、转推至CDN等），需要为云端转码功能额外付费，详情请参见实时音视频RTC价格说明。

- 计费项：云端转码时长
- 付费方式：后付费
- 付费周期：按小时扣费，即北京时间整点扣费并生成账单。出账单时间是当前计费周期结束后1小时内。例如，10:00-11:00的账单会在12:00之前生成，具体以系统出账时间为准
- 开通服务前需保证账户无欠款

计费公式

云端转码费用 = 音频转码费用 + 视频转码费用 = 音频输入转码时长 × 音频转码单价 + 视频输入转码时长 × 相应输入流视频分辨率与输出编码方式对应档位单价。

计费说明

- 按分钟计费，不足1分钟按1分钟计
- **云端转码时长**，按同一账号下所有应用使用云端转码时转码服务每路流的**输入时长**来统计云端转码服务的用量。输入时长根据云端转码服务接收音视频流类型，分为**视频转码制时长**和**音频转码时长**，**视频转码时长有不同分辨率区分**。
- 转码前，输入的同一条流在同一时间内，既有视频又有音频时，只按视频时长统计，不重复计算语音时长。
- 转码前，多路输入流中有纯音频流时，也需要计算该路流的音频时长。
- 转码前，输入的同一条流的分辨率可能会发生变化，则分段统计服务用量，通常情况下60秒更新一次，当分辨率发生变化时则立即上报更新。
- 在纯语音输入的场景下，多个音频输入按集合音频计费，即多音频输入按一份音频输入进行混流转码计费。

余额不足提醒和欠费处理

本文介绍RTC服务余额不足提醒和欠费后服务对应的处理说明。

余额不足提醒

- 根据您最近3天的账单金额来判断您的账户余额（含可用代金券）是否足够支付未来3天的费用，若不足以支付，系统发送续

费提醒。

- 根据您最近1天的账单金额来判断您的账户余额（含可用代金券）是否足以支付未来1天的费用，若不足以支付，系统发送续费提醒。

欠费处理

- 北京时间整点检查您的账户余额是否足以支付本次RTC账单的费用（如北京时间11点整检查账户余额是否足以支付10点至11点的账单费用），若不足以支付，即为欠费，欠费时系统会发送欠费通知。
- 欠费后立即停服，系统会发送欠费停服通知，数据为您保留7天，期间不收取费用，7天内未充值则释放，释放前1天和释放时系统都会发送释放通知。

快速入门

快速开始

本文档将介绍从创建百度智能云账号开始，到完成一次音视频通信为止，之间所需要的必要的操作步骤。

准备工作

1.1 注册百度智能云账号，并完成实名认证。

创建应用

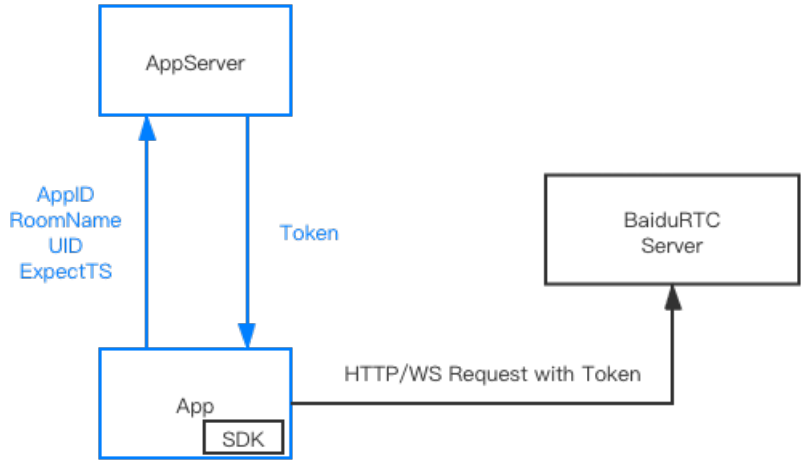
2.1 进入RTC产品控制台，选择应用管理-创建应用。

在AppServer上部署token生成服务

为保证实时通信的安全，在通信时需要进行特殊鉴权。客户在自己的AppServer（客户的后端服务器）上部署token生成服务，生成token并在通信中使用，来实现通信的鉴权。

RTC鉴权机制

鉴权流程如下：



1. 当App需要使用RTC服务时，向AppServer（客户的后端服务器）请求Token；
2. AppServer根据token生成算法以及相应的AppKey生成相应的Token，并下发给App；
3. App在调用RTC SDK时，提供Token；
4. SDK在与RTC后端服务器发起http/websocket连接时提供Token参数；
5. RTC后端对token进行验证，验证不通过时会拒绝访问；验证通过，则提供RTC服务。

部署token生成服务

为了实现上述的鉴权机制，需要在客户的AppServer上部署Token生成服务。

Token计算需要以下参数：

- AppID：由baiduRTC提供，全局唯一，用于识别应用；
- AppKey: 由baiduRTC提供，每个AppID拥有一个AppKey，并且可更新；您可以登录百度智能云- 并进入RTC控制台查看AppID和AppKey。
- RoomName: 房间名称，由客户指定并维护；
- Uid: 用户id，由客户指定并维护，客户保证其唯一性；
- ts: token生成时的秒级别unix时间戳；
- ExpectTS: 过期时间，秒级别unix时间戳；
- Version：鉴权版本，当前取值004。

需要注意的是，**Token与AppID+RoomName+Uid对是唯一绑定关系**，不可以将一个正确的token用于其它的AppID+RoomName+Uid对。

Token计算步骤

1. 输入需要的6个参数。
2. 生成当前秒级别时间戳ts。格式需要转换为十进制Unix时间戳。推荐一个在线转换网站
3. 生成randomString。随机一个int型整数，并转成对应16进制数字字符串，并补齐至8位。
4. 生成签名。使用HmacSHA1算法加密来计算签名，结果返回16进制字符串。 key=appKey; data="ACS"（固定的ACS三个字母） + appID + ts + randomString + roomName + uid + expectTs; signature = HamcSHA1(key, data);
5. 拼接token

token = version + signature + ts + randomString + expectTs;

Token的格式：

字段	说明	长度	示例
version	鉴权版本	3	004
ts	token生成时的秒级别unix时间戳	10字符	154476606
signature	签名	40字符	8f50a1f280e69f4581dd8bf8b3b9cc9d277cd3a6
randomString	salt string，int转16进制字符串	8字符，不足补0	dabdd97c
expectTs	过期时间，秒级别unix时间戳	10字符	1578380254

计算示例

准备好的各参数为： appID: app-jcagj2g5ecrqv7bn

version: 004

ts: 1553144847

randomString: dabdd97c

expectTs: 1578380254

roomName：aaa

uid: 54321

AppKey: s-jcagj2g5eewai6u3p3

计算得signature: HamcSHA1(s-jcagj2g5eewai6u3p3, ACSapp-jcagj2g5ecrqv7bn1553144847dabdd97caaa543211578380254) = 08b1a06afc9c138625dc7fc9ab5e7770c5156f23

构造token为：00408b1a06afc9c138625dc7fc9ab5e7770c5156f231553144847dabdd97c1578380254

您可以下载token生成实例代码进行参考（示例代码包含JAVA和Python两种语言）。

您也可以使用token在线生成工具来生成token; 使用token在线校验工具来校验token。

集成客户端SDK

您可以访问 下载专区 下载不同终端最新版SDK。

开通服务

您的应用接入实时音视频服务，您必须先开通实时音视频RTC产品服务。

要开通实时音视频RTC产品服务，您需要遵循以下步骤：

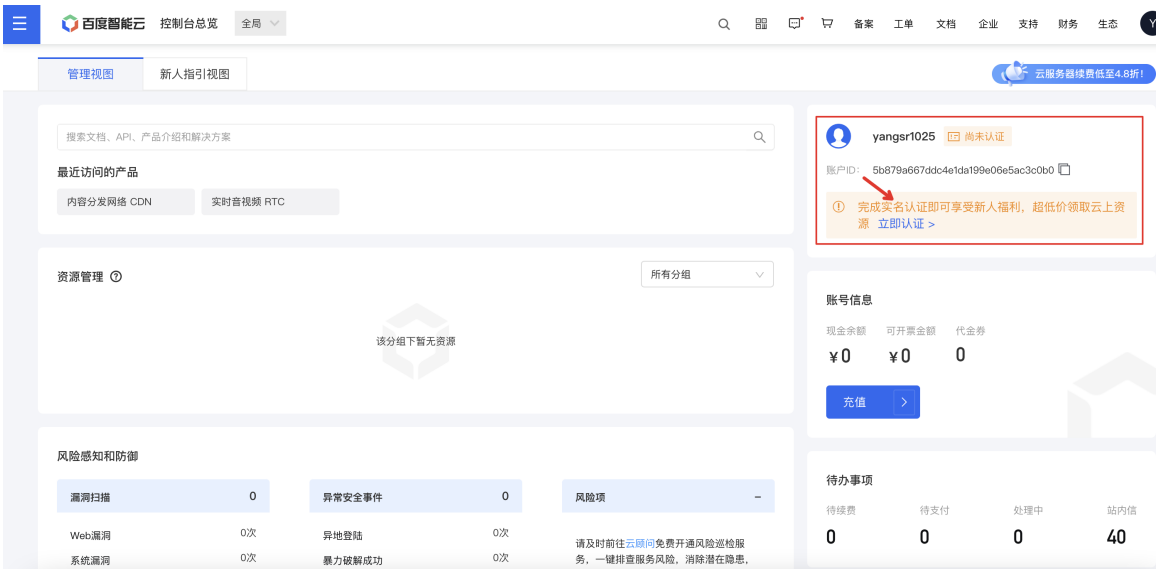
1、登录百度智能云控制台

1. 进入百度智能云控制台，如果您是首次登录，请先注册账号，参看账号注册。

2. 如果您已拥有百度智能云账号，请先登录，参看登录。

2、实名认证

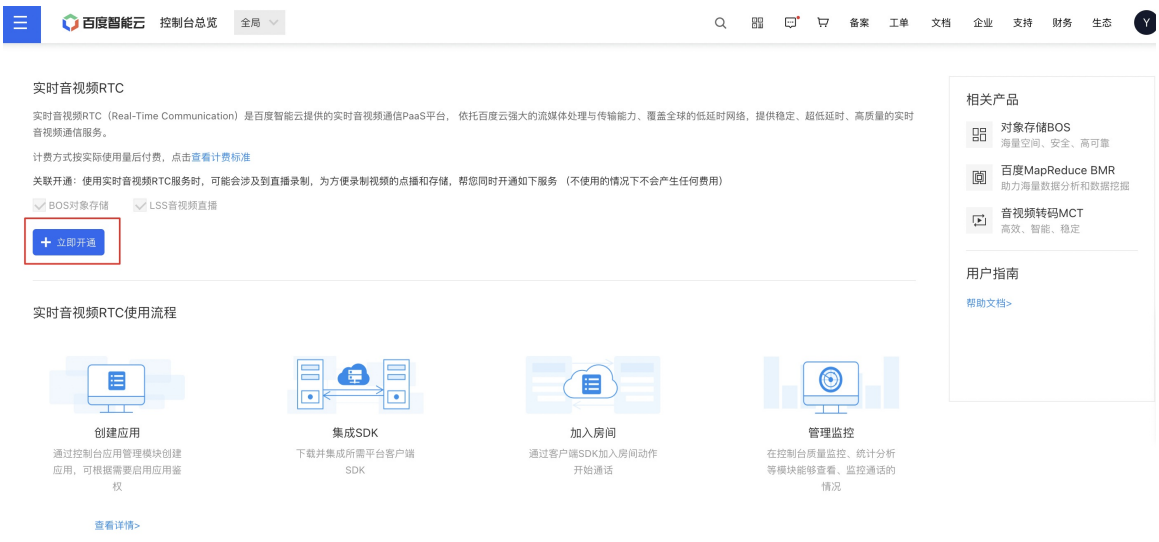
登录成功后，您必须先进行实名认证，参看实名认证：



3、申请开通 RTC 服务

您可以在总览页选择 「**实时音视频 RTC**」 进入并**申请开通**。

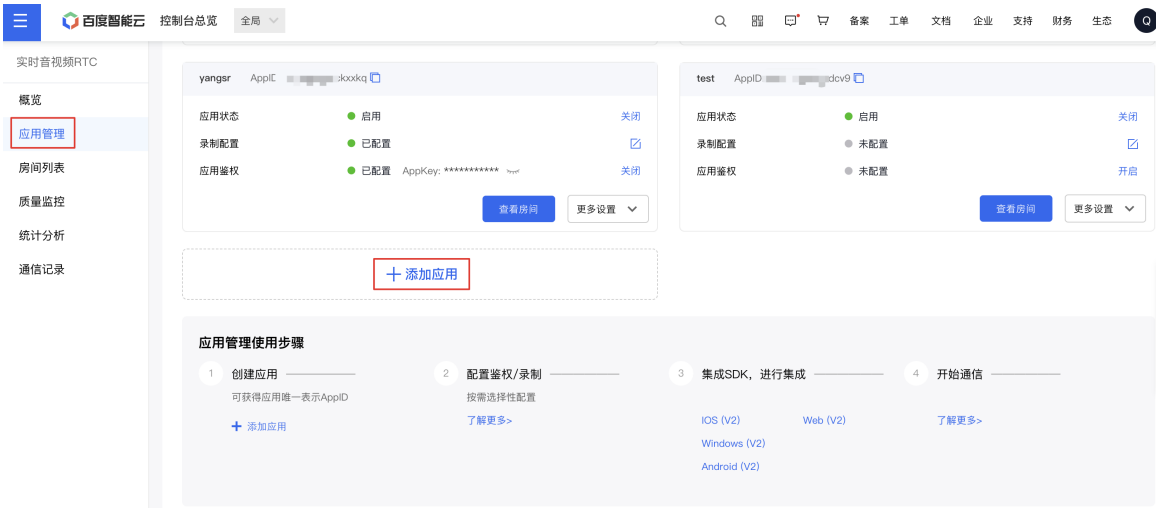
也可以通过 **实时音视频 RTC** 产品详情页 点击 **立即使用** 申请开通。



4、创建 RTC 应用，获取 AppId

进入 RTC 产品控制台后，您可以在 「**应用管理**」 中管理您的应用（包括修改应用名称、应用状态、查看 AppID 、 AppKey 等）。

如果您需要创建新的应用，可以点击**添加应用**。



注释

1. AppId 是每个应用的唯一标识符，在调用 RTC SDK 的 API 接口实现功能时，您必须填入您获取到的 AppId 。

2. AppKey 是每个应用对应的密钥，用于生成 Token 鉴权，请妥善保管。

集成RTC SDK构建应用

集成Android SDK

准备环境 本节将介绍如何创建项目，将BRTC SDK集成进你的项目中。

- Android Studio 3.2 或以上版本，Gradle 4.6或以上版本，编译环境请选择支持java8
- Android KK（4.4）及以上的设备

*注：经过验证的开发环境如下：

```
------------------------------------------------------------
Gradle 4.6
------------------------------------------------------------

Groovy: 2.4.12
Ant: Apache Ant(TM) version 1.9.9 compiled on February 2 2017
JVM: 1.8.0_192 (Oracle Corporation 25.192-b12)
OS: Mac OS X 10.13.6 x86_64
```

下载SDK

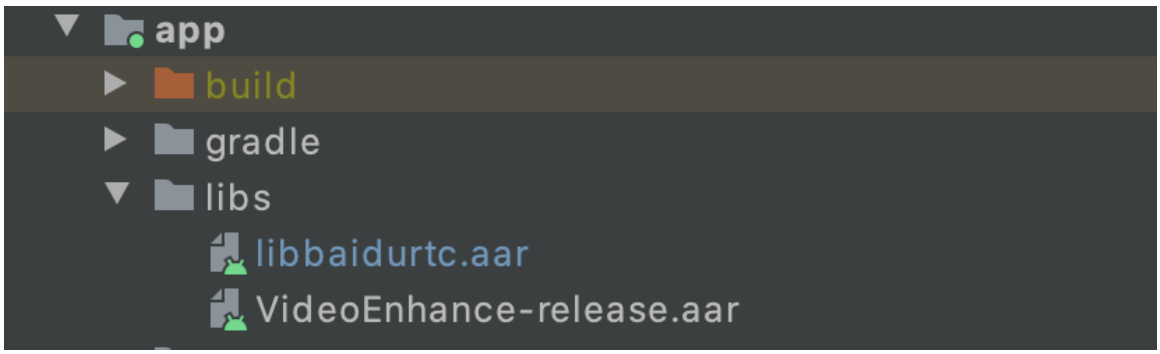
maven方式 推荐使用 Maven 在项目中接入 RTC SDK，步骤如下：

SDK maven 版本列表： https://repo1.maven.org/maven2/com/baidubce/mediasdk/brtc/

```
implementation 'com.baidubce.mediasdk:brtc:2.xx.xx'
```

离线方式 进入RTC文档中心，点击“下载专区>SDK&Demo下载”，即可下载客户端SDK。下载后请校验下载的包md5值与SDK中心里记录的是否一致。

创建Android项目，若已有 Android 项目，可以直接集成 SDK
将SDK包内libbaidurtc.aar 拷贝到项目的libs目录.



集成SDK

maven依赖可以直接进行使用，离线方式需要手动添加依赖；

添加项目权限

```
 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.baidu.rtc.videoroom">

   <uses-feature android:name="android.hardware.camera" />
   <uses-feature android:name="android.hardware.camera.autofocus" />
   <uses-feature android:glEsVersion="0x00020000" android:required="true" />

   <uses-permission android:name="android.permission.CAMERA" />
   <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
   <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
   <uses-permission android:name="android.permission.RECORD_AUDIO" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
...
</manifest>
```

屏幕分享项目权限

```
 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.baidu.rtc.videoroom">

   // 前台服务
   <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>

   // 多进程模式下，添加注册（与多流模式互斥）
   <activity
      android:name=".screenshare.ScreenCaptureAssistantActivity"
      android:theme="@android:style/Theme.Translucent"
      android:process=":brtc_sharescreen" />
   <service
      android:name=".screenshare.ScreenSharingService"
      android:process=":brtc_sharescreen" />
   <service
      android:name="com.baidu.rtc.CaptureScreenService"
      android:enabled="true"
      android:process=":brtc_sharescreen"
      android:foregroundServiceType="mediaProjection"/>

   // 多流模式下，添加注册（与多进程模式互斥）
   <service
      android:name="com.baidu.rtc.CaptureScreenService"
      android:enabled="true"
      android:foregroundServiceType="mediaProjection"/>
...
</manifest>
```

防止代码混淆 在 app/proguard-rules.pro 文件中添加如下行，防止代码混淆：

```
 -keep class com.baidu.cloud.rtcbridge.framecapture.RtcFrameCaptureImpl {*;}
-keep class com.baidu.cloud.rtcbridge.frameprocessor.RtcFrameProcessorImpl {*;}

-keep class com.baidu.rtc.BaiduRtcRoom {*;}
-keep class com.webrtc.** {*;}
-keep class com.baidu.rtc.** {*;}
-keep class com.baidu.cloud.rtcbridge.framecapture** {*;}

-dontwarn com.baidu.rtc.**
-dontwarn com.webrtc.**

-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**

-keep class okio.** { *; }
-keep interface okio.** { *; }
-dontwarn okio.**

-keep class javax.annotation.** {*;}

// crash 上报
-dontwarn com.tencent.bugly.**
-keep public class com.tencent.bugly.**{*;}
```

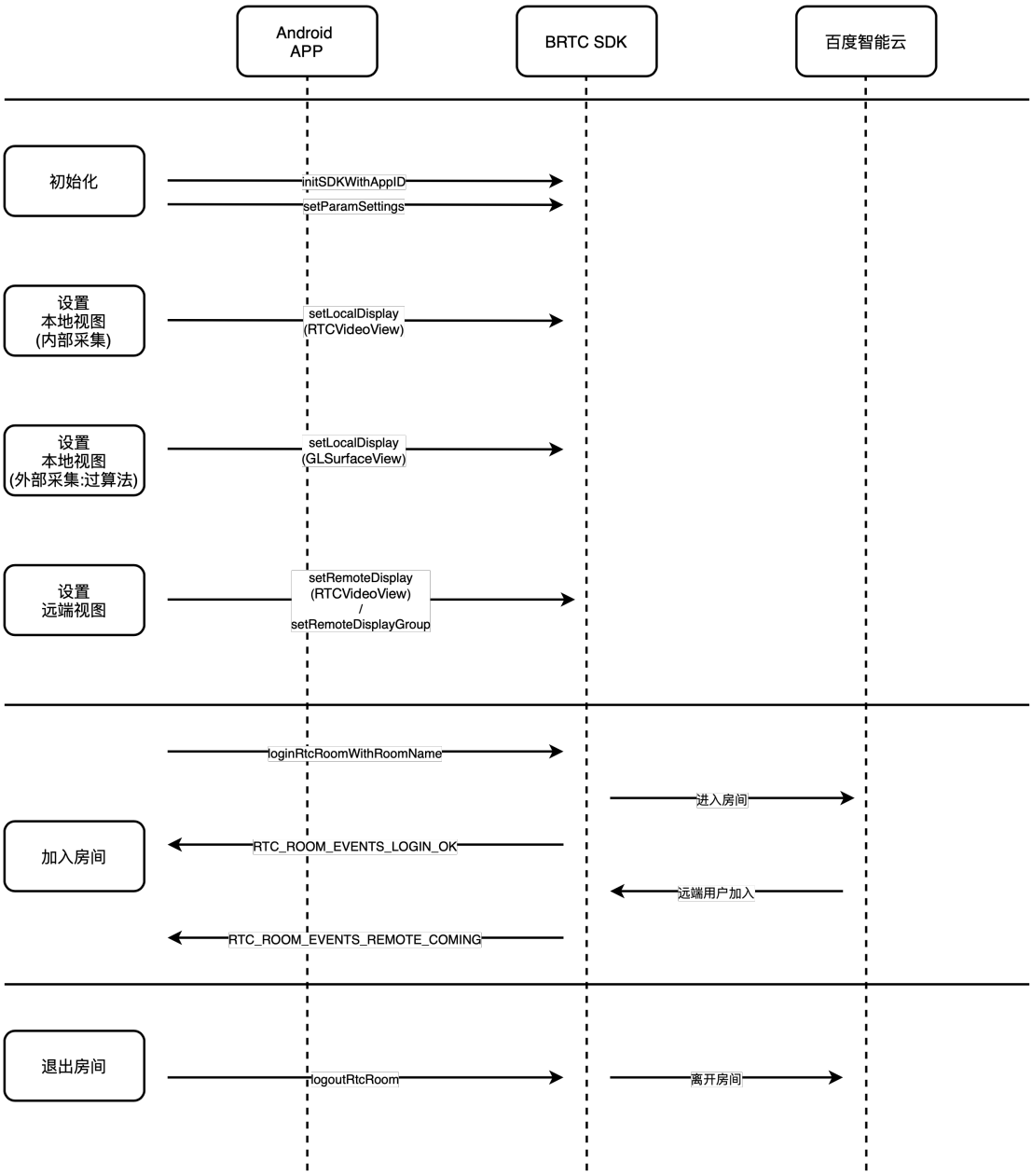
项目配置 在项目的build.gradle中加入如下配置代码， 即可使用。

```
  android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

implementation 'com.squareup.okhttp3:okhttp:3.5.0'
implementation 'com.squareup.okhttp3:logging-interceptor:3.5.0'
implementation(name: 'libbaidurtc', ext: 'aar')
```

实现音视频通话

本节介绍如何实现音视频通话。音视频通话的API调用时序见下图:



外部采集算法：美颜、滤镜、视频增强等算法。

代码示例

1.在您的开发项目中导入包

```
import com.baidu.rtc.BaiduRtcRoom;
```

2.定义 rtc room 变量

```
private BaiduRtcRoom mVideoRoom
```

3.初始化 sdk 并设置代理。初始化的时候要带上上下文环境 context，appid，token 串，并确定是否开启后下载

```
mVideoRoom = BaiduRtcRoom.initWithAppID(this,mAppId,mTokenStr, true);
mVideoRoom.setBaiduRtcRoomDelegate(this);
```

4.音视频参数设置：

```
 RtcParameterSettings cfg = RtcParameterSettings.getDefaultSettings();
cfg.VideoResolution = mVideoResolution;
cfg.VideoFps = 30;
cfg.AutoPublish = true; //default is true.  for mVideoRoom.startPublish() set to false.
cfg.AutoSubScribe = true; //default is true.  for mVideoRoom.subscribeStreaming() set to false.
mVideoRoom.setParamSettings(cfg,RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);
```

5. 视频显示view设置

```
 mVideoRoom.setLocalDisplay((RTCVideoView) findViewById(R.id.local_rtc_video_view));
//单人模式:
mVideoRoom.setRemoteDisplay((RTCVideoView) findViewById(R.id.remote_rtc_video_view));
// 给定用户Id 绑定远端view
mVideoRoom.setRemoteDisplay(rtcVideoView, userId);

// 多人模式，不绑定userId，
RTCVideoView[] vg = new RTCVideoView[5];
vg[0] = (RTCVideoView) findViewById(R.id.remote_rtc_video_view);
vg[1] = (RTCVideoView) findViewById(R.id.remote_rtc_video_view1);
vg[2] = (RTCVideoView) findViewById(R.id.remote_rtc_video_view2);
vg[3] = (RTCVideoView) findViewById(R.id.remote_rtc_video_view3);
vg[4] = (RTCVideoView) findViewById(R.id.remote_rtc_video_view4);
mVideoRoom.setRemoteDisplayGroup(vg);

// 多人模式，绑定userId的方式请用多个 setRemoteDisplay(rtcVideoView, userId)
```

6. 登录房间

```
 mVideoRoom.loginRtcRoomWithRoomName(mRoomName,java.lang.Long.parseLong(mUserId),mUserName);
```

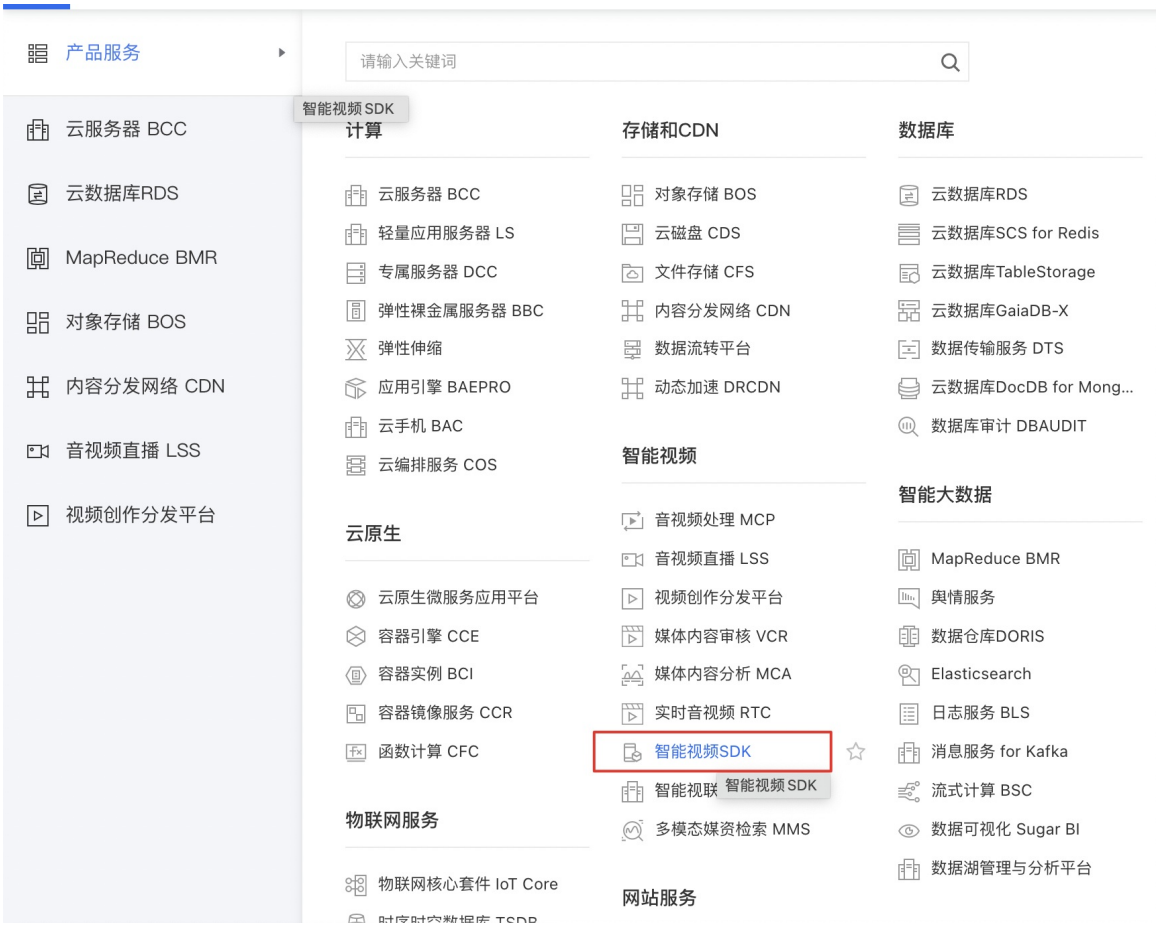
7. 登出房间，结束音视频通话

```
 mVideoRoom.logoutRtcRoom();
mVideoRoom.destroy();
```

8. 外部算法接入 8.1 美颜算法接入

8.1.1）官网申请license

官网地址： https://console.bce.baidu.com/

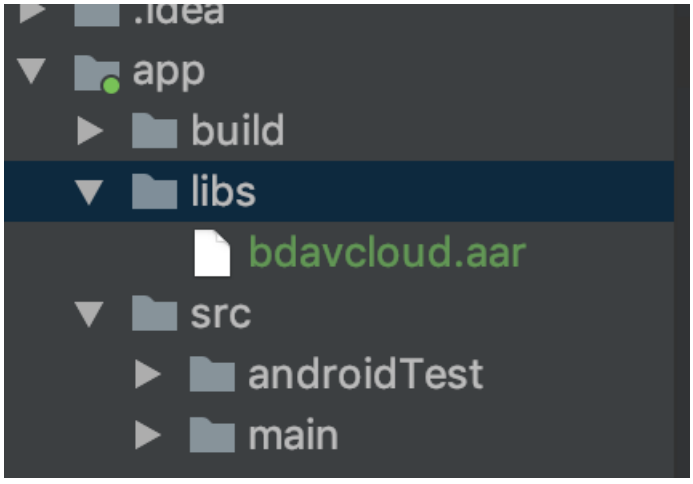


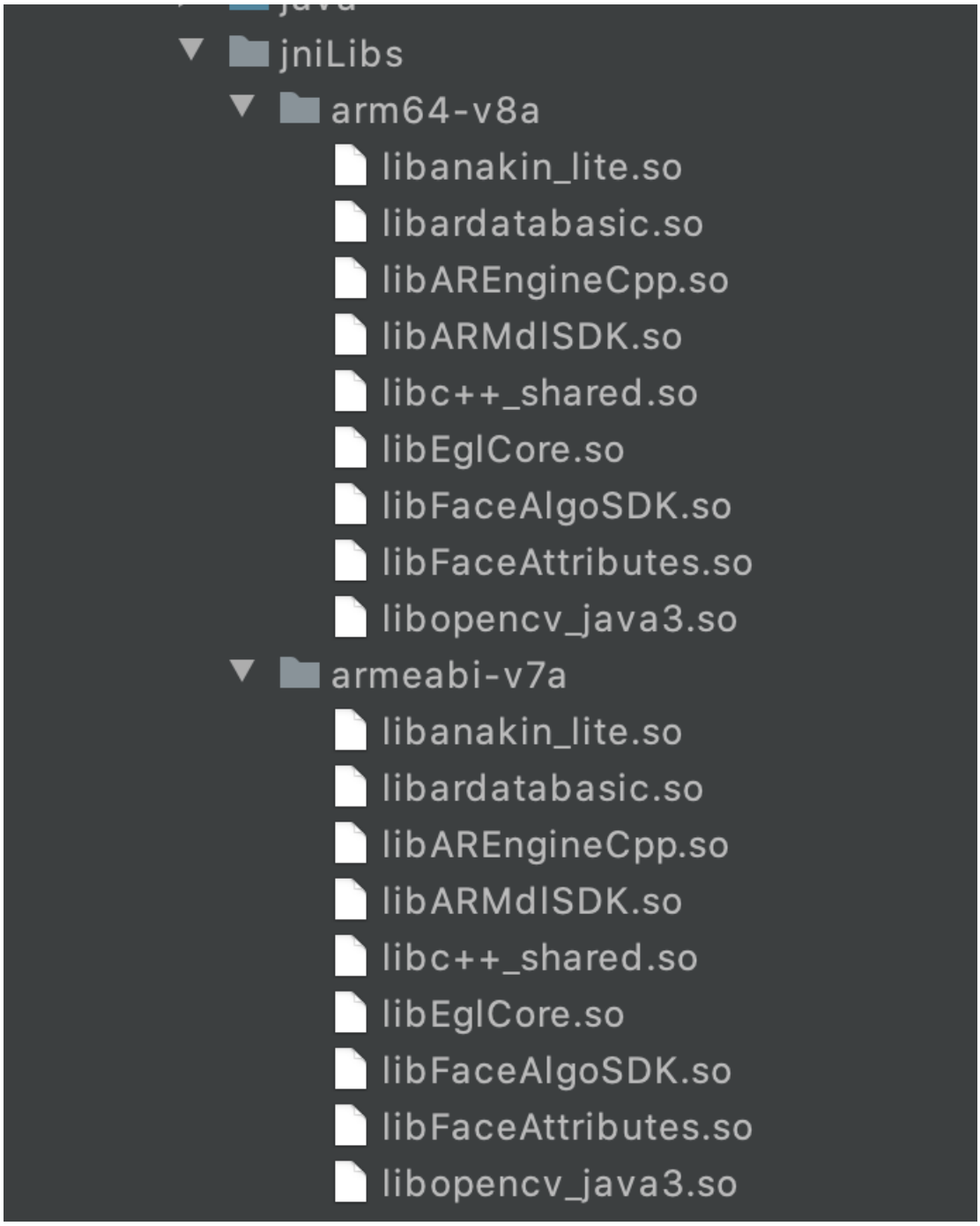
产品服务——>短视频创作SDK VC_SDK——>选择license申请——>填写相关信息——>得到license 备注：如操作遇到问题，请联系人工服务。

8.1.2)项目路径

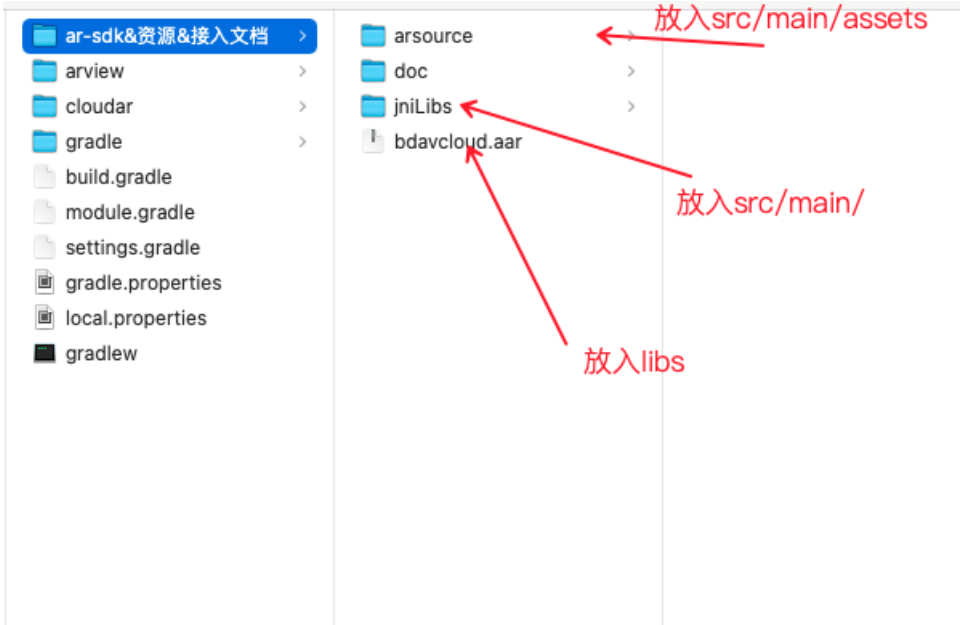
v2.4.0之前：

将下载解压缩之后的 SDK 目录下的aar文件拷贝到工程的 app/libs 目录下（如下图）





SDK包含的内容



8.1.3)在项目build.gradle添加库依赖

RTC美颜 SDK maven: https://repo1.maven.org/maven2/com/baidubce/mediasdk/brtc_frameprocessor/

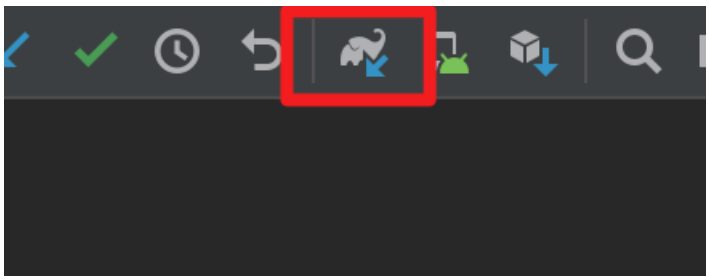
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    api fileTree(dir: 'libs', include: ['*.aar'])
    implementation 'com.baidubce.mediasdk:brtc_frameprocessor:2.13.0.2'
    implementation 'com.android.support:appcompat-v7:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

    implementation 'com.android.support:recyclerview-v7:28.0.0'
    implementation "com.github.bumptech.glide:glide:4.6.1"

    api "com.facebook.fresco:fresco:1.11.0"
    api 'com.airbnb.android:lottie:2.7.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    api 'com.google.code.gson:gson:2.8.5'
}
```

8.1.4)同步SDK，单击 Sync Now 按钮，完成短视频sdk的集成工作。



注意：在使用拍摄器SDK，需要申请产品对应的授权文件，如无授权，产品无法正常使用。

v2.4.0（含）之后： 拿到鉴权后，参考Demo，直接使用arview中libbaidurtc-frameprocessor.aar即可

8.1.5)配置license授权

第一步：修改自定义Application类，替换自己申请的licenseID（可参考demo RtcApplication） 第二步：替换app/src/main/assets中后缀名为.license的文件为上面申请的license文件（licenseID对应license文件） 注意：接入sdk必须在RtcApplication初始化sdk

```
 public static final String LICENSEID = "860226087481886720061";
// 初始化sdk工作，必须
BDCloudSdkInitManager.getInstance().onCreate(mContext, LICENSEID);
String packageName = getPackageName();
```

8.1.6)配置app权限 在 AndroidManifest.xml 中配置 App 的权限，

9. 屏幕分享

demo示例入口：CallActivity.java

屏幕分享管理类：ScreenSharingManager.java **v2.5.0修改aidl包名，旧版本升级需要参考2.5.0版本demo修改**

```
 // 初始化屏幕分享管理类
mScreenSharingManager = new ScreenSharingManager(ScreenSharingManager.MODE_MULTI_PROCESS, this);
mScreenSharingManager.setVideoRoom(mVideoRoom);
mScreenSharingManager.setRoomName(mRoomName);
mScreenSharingManager.setBaiduRtcRoomDelegate(this);
```

9.1）**多进程模式屏幕分享**，支持服务端录制

多进程屏幕分享实现由业务侧创建一个独立进程，在独立进程中，创建BRTC实例并loginRoom实现，具体代码：

推流端（启动和停止屏幕分享）：

```
 // ScreenSharingService.java中部分代码，屏幕分享用户加入房间（启动屏幕分享）：
private void loginVideoRoom(Intent screenResultData) {
    // 初始化BRTC，注意此处初始化最后一个参数，因为此为屏幕分享用户，默认不初始化音频管理
    // 注意！！！如开启鉴权，则token需要按规则生成，规则参见https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu#rtc鉴权机制
    mVideoRoom = BaiduRtcRoom.initWithAppID(this, appId, token, "", false,
            false);
    mVideoRoom.setBaiduRtcRoomDelegate(this);
    // 省略部分代码，仅保留屏幕分享相关代码，配置屏幕分享参数
    RtcParameterSettings cfg = RtcParameterSettings.getDefaultSettings();
    cfg.HasVideo = false;
    cfg.HasAudio = false;
    if (screenResultData != null) {
        cfg.HasScreen = true;
        cfg.screenIntentData = screenResultData;
        cfg.videoEncodeParams = new HashMap<>();
        cfg.videoEncodeParams.put(RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_SCREEN, mShareScreenParams);
    }
    cfg.EnableMultistream = true; // 多流
    cfg.AutoPublish = true; // 自动推流
    cfg.AutoSubScribe = false; // 多进程模式，屏幕流只推不拉
    mVideoRoom.setParamSettings(cfg, RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);
    // 登录房间
    mVideoRoom.loginRtcRoomWithRoomName(mRoomName, java.lang.Long.parseLong(mUserId), mUserName);
}
// ScreenSharingService.java中部分代码，屏幕分享用户离开房间（停止屏幕分享）：
private void logoutVideoRoom() {
    ScreenCaptureAssistantActivity.mCallback = null;
    mVideoRoom.logoutRtcRoom();
    mVideoRoom.destroy();
    mVideoRoom = null;
}
```

推流端（启动或停止屏幕分享回调事件）：

```
public void onRoomEventUpdate(int roomEvents, long data, final String extraInfo) {
  super.onRoomEventUpdate(roomEvents, data, extraInfo);
  switch (roomEvents) {
     // 收到屏幕共享停止状态（被其他进程占用等）
     case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_CAPTURE_SCREEN_ON_STOP:
        // 共享中，但被停止，非主动停止，同步UI状态
        break;
     // 收到屏幕共享启动、停止状态
     case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_SHARE_SCREEN_START:
     case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_SHARE_SCREEN_STOP:
        // 操作屏幕分享成功
        if (data == BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_STATE_SUCCESS) {
           // 记录状态等
        } else {
           // 失败
           Log.d(TAG, extraInfo);
        }
        });
        break;
  }
}
```

拉流端，根据特殊用户或者流类型或者自定义用户事件，同步拉流端当前用户是否是屏幕分享用户，如下示例特殊用户或者流类型判断：

```
public void onRoomEventUpdate(int roomEvents, long data, final String extraInfo) {
  switch (roomEvents) {
        // 远端用户加入
        case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_COMING:
            // 远端为特殊用户id（屏幕共享）
            if (data == ScreenSharingManager.SCREEN_SHARE_USER_ID) {
                onScreenComing(data);
                return;
            } else {
                // 远端不是特殊用户id（屏幕共享），但用户id不确定是什么，所以从target里查找匹配（目前target仅
Android、iOS端可互识别）
                ArrayList<CommonDefine.StreamInfo> comingStreams = mVideoRoom.getComingStreams(data);
                if (comingStreams != null && comingStreams.size() == 1) {
                    for (CommonDefine.StreamInfo streamInfo : comingStreams) {
                        if (TextUtils.equals(streamInfo.type,
                                RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_DEFAULT)
                                && TextUtils.equals(RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_SCREEN,
                                streamInfo.description)) {
                            onScreenComing(data);
                            mCurrentScreenUserId = data;
                            return;
                        }
                    }
                }
            }
            // 普通用户加入房间逻辑
            break;
        // 远端用户离开
        case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_LEAVING:
            // 远端为特殊用户id（屏幕共享）
            if (data == ScreenSharingManager.SCREEN_SHARE_USER_ID || data == mCurrentScreenUserId) {
                if (mRemoteViews != null
                        && mRemoteViews.containsKey(RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_SCREEN)) {
                    // 当前为拉流端，移除UI
                    mRemoteViews.remove(RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_SCREEN);
                    notifyViewChanged();
                }
                mCurrentScreenUserId = INVALID_USER_ID;
                return;
            }
            // 普通用户离开房间逻辑
            break;
        default:
            break;
    }
}
```

9.2）**多流模式屏幕分享**，不支持服务端录制

多流模式实现方式是在当前用户BRTC通讯流中，增加和移除一条媒体流完成，由于媒体流中包含两个视频流，需要使用接口获取当前流的信息，以此区分一条流具体是相机流还是屏幕流。

推流端（启动和停止屏幕分享）：

```
 // 参见ScreenSharingManager.java部分代码

// 配置参数
RtcParameterSettings parameterSettings = RtcParameterSettings.getDefaultSettings();
parameterSettings.screenIntentData = resultData;
if (parameterSettings.videoEncodeParams == null) {
    parameterSettings.videoEncodeParams = new HashMap<>();
}
parameterSettings.videoEncodeParams.put(RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_SCREEN,
        getScreenShareParams());
mVideoRoom.setParamSettings(parameterSettings,
        RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_SHARE_SCREEN);
// 启动分享
mVideoRoom.startShareScreen();

// 停止推流
mVideoRoom.stopShareScreen();
```

推流端（启动或停止屏幕分享回调事件）：与多进程模式一致

拉流端：根据事件，区分是否中间增加、移除一路流，或者是用户加入时就以两条视频流加入

```
 public void onRoomEventUpdate(int roomEvents, long data, final String extraInfo) {
   switch (roomEvents) {
          // 远端用户加入
          case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_COMING:
              // getComingStreams 获取当前加入用户的流信息，进而为流关联特定的View
              onComing(data, mVideoRoom.getComingStreams(data));
              break;
          // 已加入的远端用户，有新的流加入
          case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_ADDED_STREAM:
              // getOperatingStreams 获取用户的加入的流信息
              onAddedStreams(data, mVideoRoom.getOperatingStreams(data));
              break;
          // 远端用户离开
          case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_LEAVING:
              // 处理移除用户View等
              break;
          // 已加入的远端用户，有流离开
          case BaiduRtcRoom.BaiduRtcRoomDelegate.RTC_ROOM_EVENT_REMOTE_REMOVED_STREAM:
              // getOperatingStreams 获取用户的移除的流信息
              onRemovedStreams(mVideoRoom.getOperatingStreams(data));
              break;
          default:
              break;
      }
}
```

集成Android 纯音频SDK

集成SDK

环境准备

本节将介绍如何创建项目，将BRTC SDK集成进你的项目中。

Android Studio 3.2 或以上版本，Gradle 4.6或以上版本，编译环境请选择支持java8 Android KK（4.4）及以上的设备

*注：经过验证的开发环境如下：

```
------------------------------------------------------------
Gradle 4.6
------------------------------------------------------------

Groovy: 2.4.12
Ant: Apache Ant(TM) version 1.9.9 compiled on February 2 2017
JVM: 1.8.0_192 (Oracle Corporation 25.192-b12)
OS: Mac OS X 10.13.6 x86_64
```

下载和安装

进入RTC文档中心，点击“下载专区>SDK&Demo下载”，选择Android 纯音频SDK，即可下载客户端SDK。下载后请校验下载的包md5值与SDK中心里记录的是否一致。

快速入门

SDK中包含Demo工程，可用于参考接入

```
// 初始化SDK，获取appID，token 请于官网注册（https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu）
BaiduRtcRoom  mAudioRoom = BaiduRtcRoom.initWithAppID(context, "appid", "token");

// sdk 参数集合
RtcParameterSettings cfg = RtcParameterSettings.getDefaultSettings();
cfg.AutoPublish = true; // 登录后自动推流
cfg.AutoSubScribe = true; // 登录自动订阅
cfg.HasVideo = false;    // 纯音频SDK 无视频
cfg.HasData = false;     // 关闭数据通道

// 设置SDK 参数
mAudioRoom.setParamSettings(cfg,
            RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);

// 设置SDK 回调监听
mAudioRoom.setBaiduRtcRoomDelegate(delegate);

// 登录
mAudioRoom.loginRtcRoomWithRoomName(mRoomName, java.lang.Long.parseLong(mUserId), mUserName, true);

// 登录成功后会自动推流

// 登出
mAudioRoom.logoutRtcRoom();

// 资源销毁
mAudioRoom.destroy();
```

日志上报

UserId 推荐每个用户唯一，方便后续排查问题。

BaiduRtcRoom.setVerbose(true) 用于打开日志，建议在调试阶段打开此开关，方便调试。

setWriteLogConfig 用于打开日志写本地文件能力。

enableAutoUploadLog(true) 开启后，会自动上传日志文件到云服务器，方便排查线上问题。

问题上报

SDK 相关问题可提工单，携带 AppId、UserId、RoomName参数，和现象与问题时间点

异常处理

可参考下面实现，详细处理可参考 Demo 工程中重试逻辑实现

```
// 部分异常回调事件，收到需进行重连处理
  /**
   * 链接丢失，需进行重连处理
   */
  public void onConnectionLost(int code);

  /**
   * 登录失败后，需进行重连处理
   *
   * @param code  0 成功，其他失败, 101 timeout
   * @param roomId 房间id
   */
  public void onLoginRtcRoom(int code, long roomId);
  /*
   * 异常事件，需进行重连处理
  */
  public abstract void onError(/*RtcErrorCodes*/int errorInfo);

/* 重连参考代码，详细代码可参考Demo工程 */
  // 登出
  mAudioRoom.logoutRtcRoom();
  // 资源销毁
  mAudioRoom.destroy();
  async sleep(200ms);
  mAudioRoom = BaiduRtcRoom.initWithAppID(context, "appid", "token");
  mAudioRoom.setParamSettings(cfg,
          RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);
  mAudioRoom.setBaiduRtcRoomDelegate(delegate);
  mAudioRoom.loginRtcRoomWithRoomName(mRoomName, java.lang.Long.parseLong(mUserId), mUserName);
```

混淆配置

```
-keep class org.webrtc.** {*;}
-keep class com.baidu.rtc.BaiduRtcRoom {*;}
-keep class com.baidu.rtc.BaiduRtcRoom$* {*;}
```

权限

```
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

集成iOS SDK

说明

- 提供的项目以 Xcode 13.0 为例 **准备环境**
- Xcode 9.0 或以上版本
- 支持 iOS 9.0 或以上版本的 iOS 设备

- 支持音视频功能的模拟器或真机

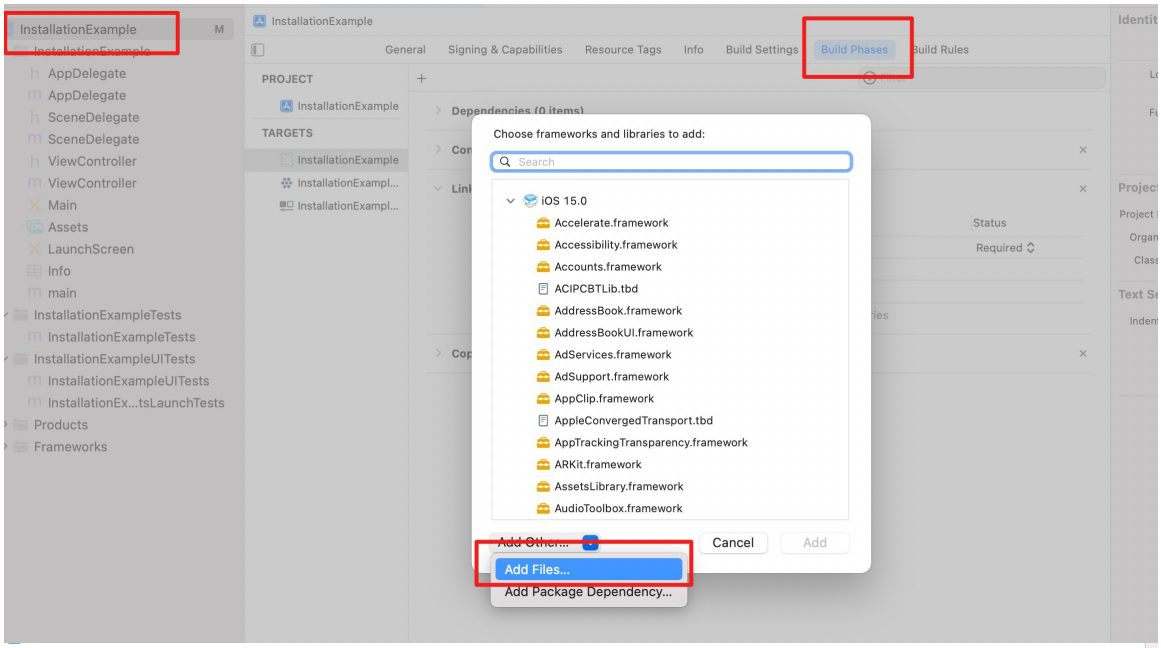
下载SDK

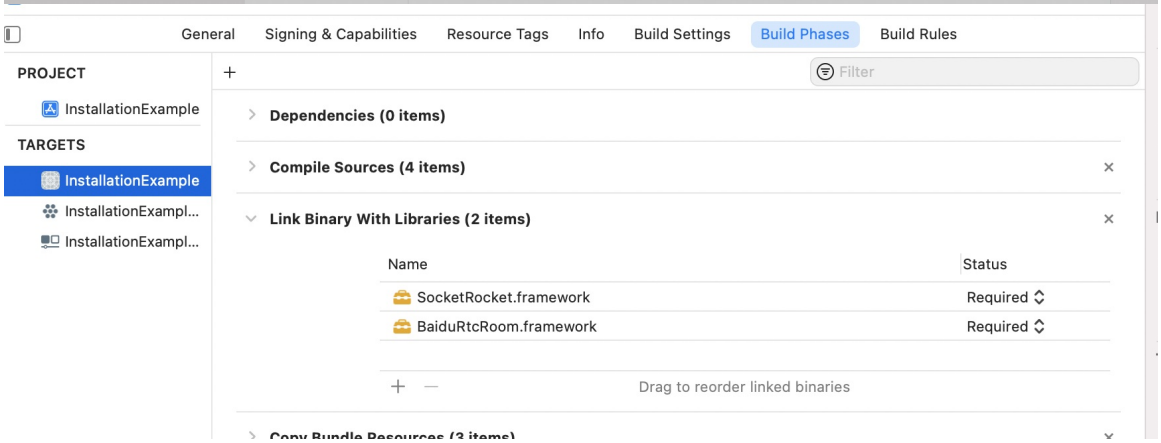
进入RTC文档中心，点击“下载专区>SDK&Demo下载”，即可下载iOS客户端。

集成SDK 创建一个iOS项目，若已有 iOS 项目，可以直接集成 SDK。此版本SDK提供静态库及动态库形式。

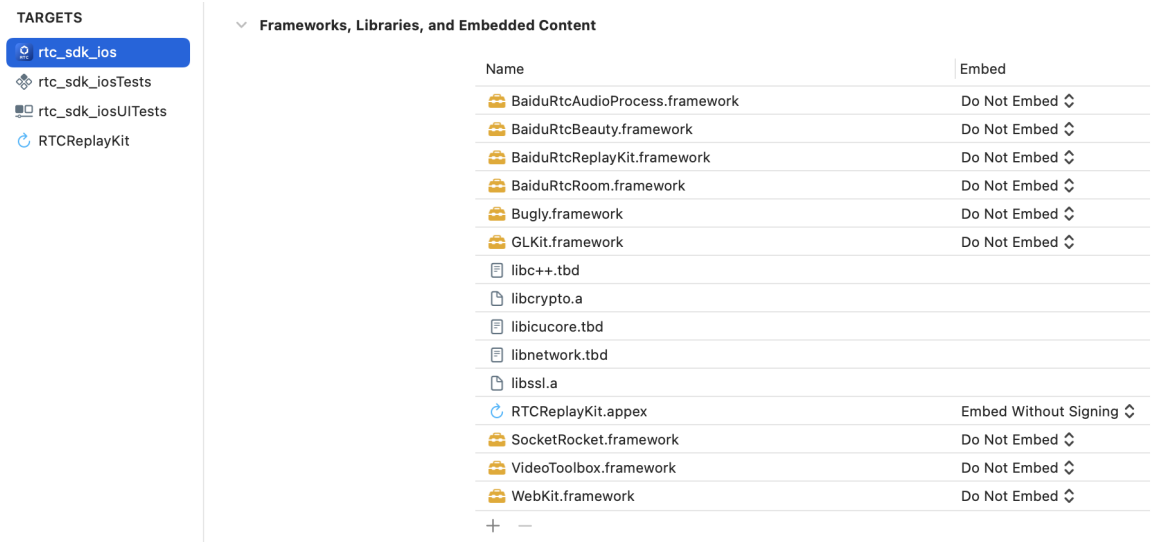
添加 SDK 静态库文件

1. 将下载的SDK中lib文件夹内的 BaiduRtcRoom.framework SocketRocket.framework文件复制到项目文件夹下
2. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，添加 SDK 静态库文件到项目。





3. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，将libyuv.a，libcrypto.a，libssl.a添加到项目中
4. 添加依赖库



其中 ：

BaiduRtcRoom 为 BRTC SDK 主依赖库，必须添加；

SocketRocket 为三方依赖库，必须添加；

libyuv.a，libcrypto.a，libssl.a为百度内部依赖库，必须添加；

其他 BaiduRtc 前缀 framework 为 BRTC SDK 扩展模块依赖库，按需添加；

其余为系统依赖库，必须添加

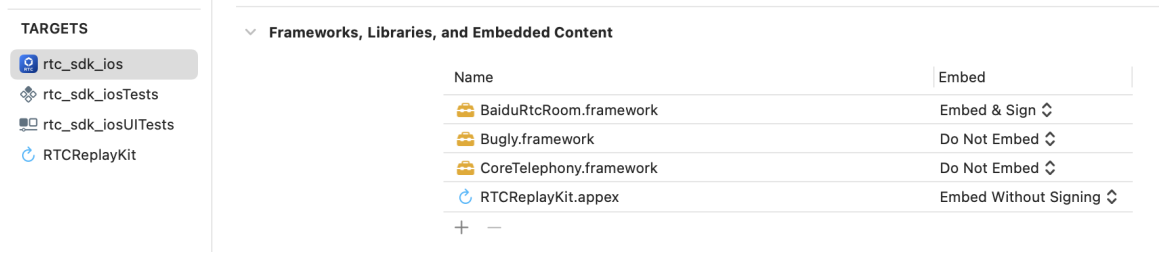
添加 SDK 动态库文件

若使用动态库SDK，可按以下步骤进行集成：

1. 将下载的 SDK 中 dynamic 文件夹内的 BaiduRtcRoom.framework 文件复制到项目文件夹下

2. 选择：项目 TARGET -> Build Phases -> Link Binary With Libraries，添加 SDK 动态库文件到项目。

注意动态库须以 Embed & Sign 形式依赖

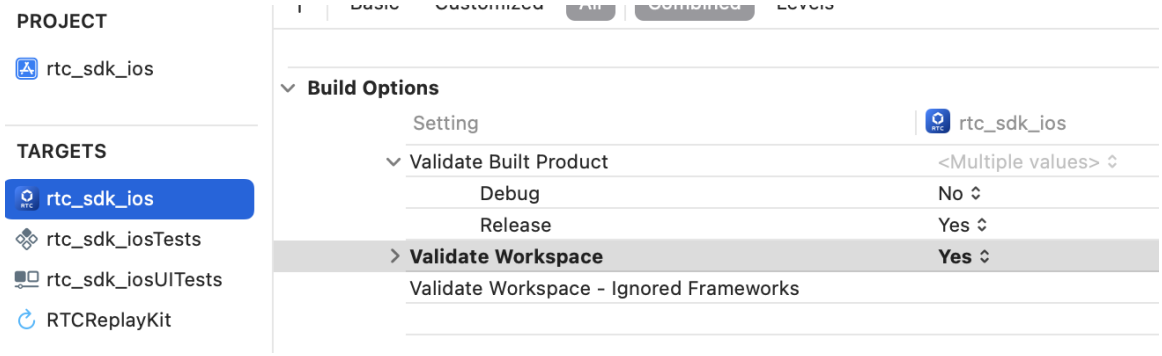


3. xcode 兼容

若出现 Building for iOS, but the linked and embedded framework was built for iOS + iOS Simulator 错误，说明所使用的 xcode 版本默认配置并不直接兼容 framework 形式的动态库，可按以下方式解决：

一、Validate Workspace

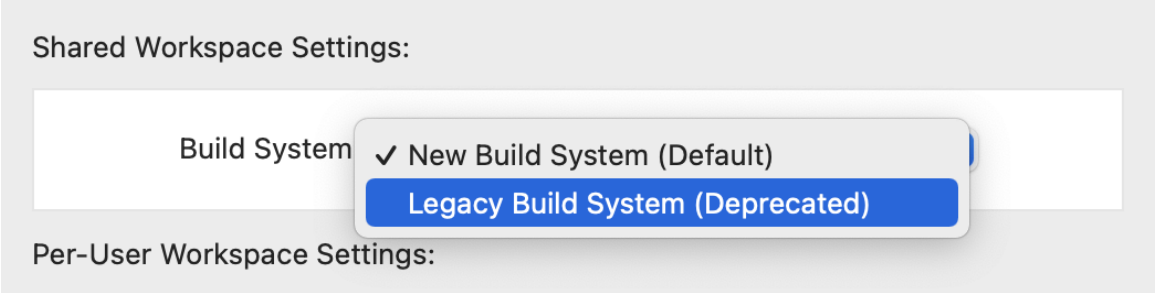
在 宿主 xcode build settings 中启用 Validate Workspace，该配置将对 xcode 构建配置做兼容性检测， 但通过此方式解决后，对应的错误会变成⚠存在。



二、Legacy Build System

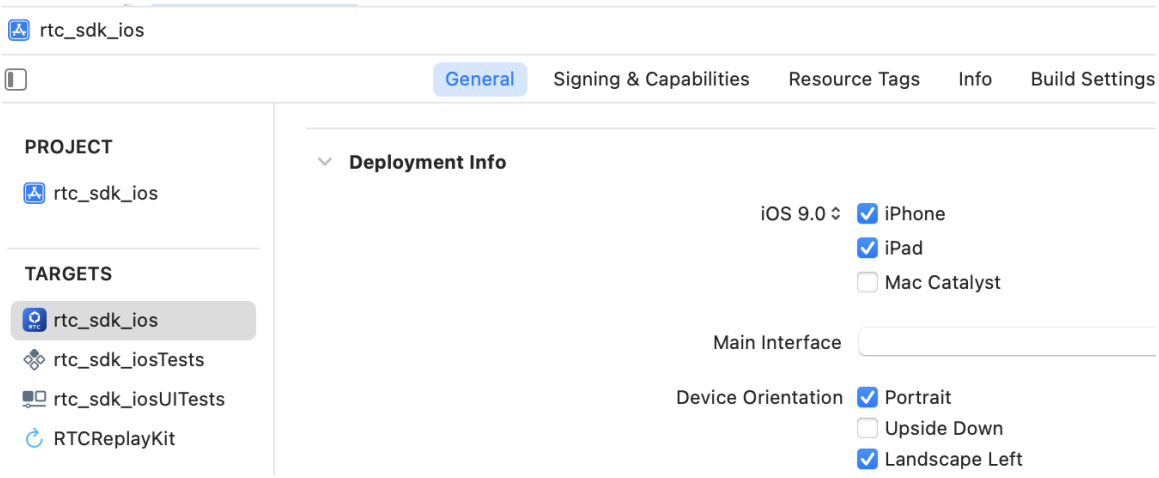
若兼容性检测设置未能解决报错问题，可将 xcode 的 Build System 改为 Legacy Build System

Xcode -> File -> Workspace Settings -> Shared Workspace Settings



项目设置

1. 打开 Xcode，选择：项目 TARGET -> General -> Deployment Target，设置 9.0 或以上版本。



使用扩展能力模块

BRTC 自 v2.3 开始，拆分或添加部分功能到独立于主产物的扩展模块产物中，

接入方可按需和主产物一起集成，以使用相关能力。

现有扩展模块列表如下，版本列 表示使用该扩展能力时，同时集成的主产物所需满足的最小版本：

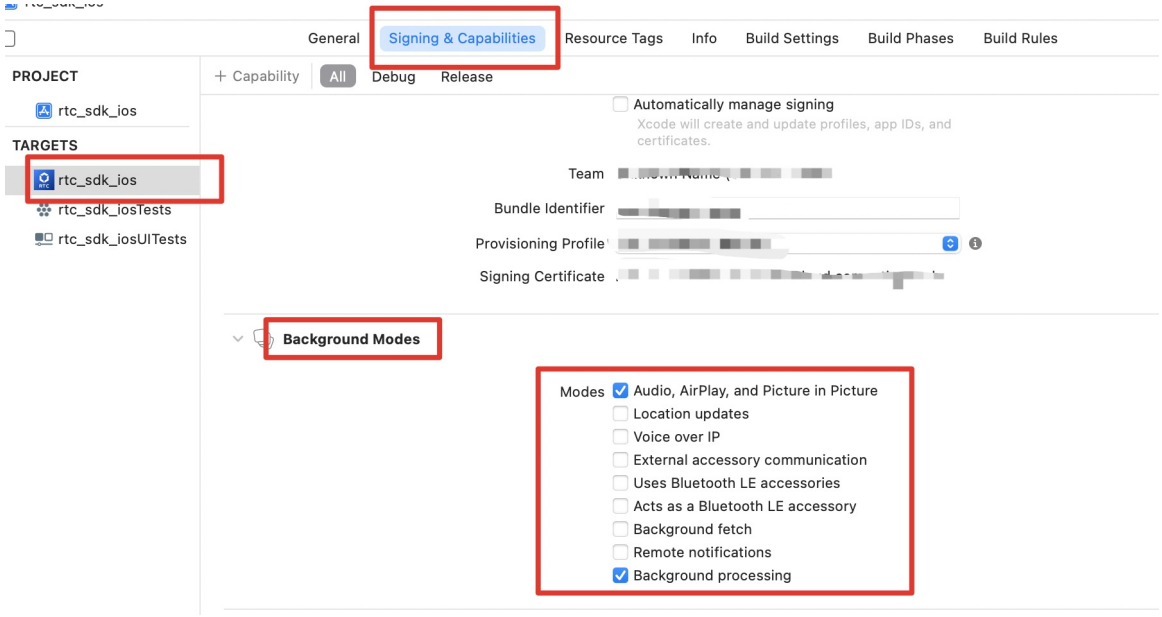
名称	描述	主产品最低要求版本	能力
BaiduRtcReplayKit	屏幕分享	v2.3	主产物使用 应用内屏幕分享、系统屏幕分享 能力必须依赖的模块
BaiduRtcBeauty	美颜	v2.3	采集视频前处理美颜，包括美白、磨皮、塑形 等
BaiduRtcAudioProcess	音频处理	v2.4	采集音频前处理，包括 人声变声 等
BaiduRtcVideoProcess	视频处理	v2.6	采集视频水印 等

添加相机/麦克风权限

在Info.plist文件中添加Privacy相机/麦克风权限

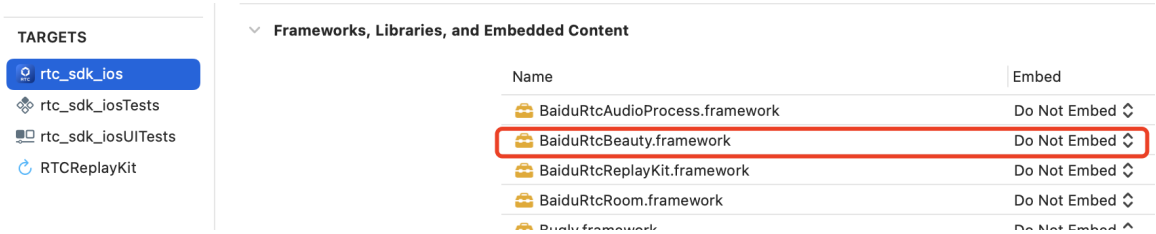
添加Background Modes

在Target -> Signing & Capabilities -> Background Modes中添加如下Modes



使用美颜采集功能

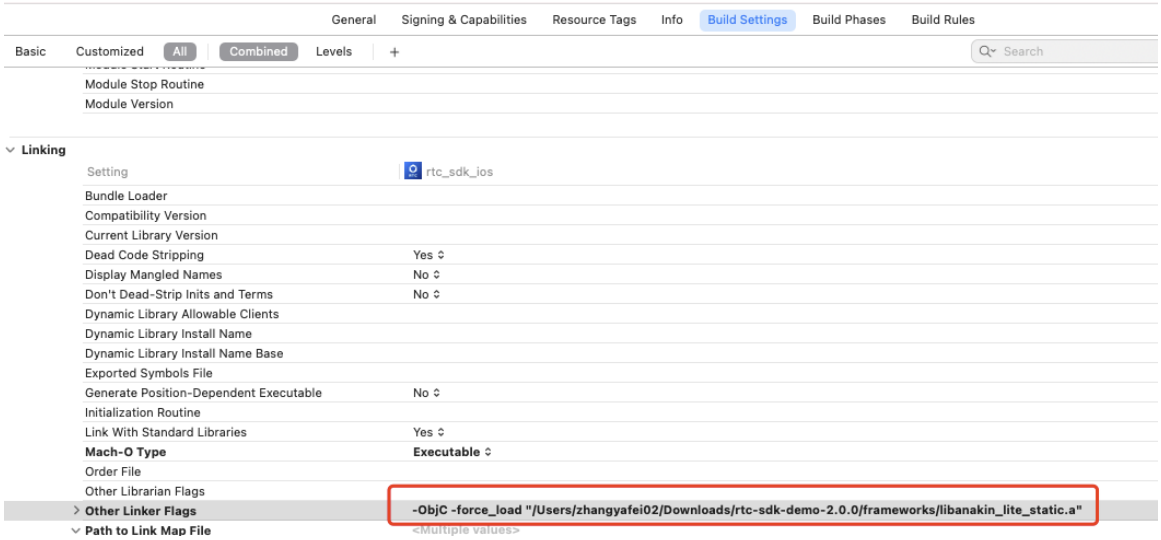
1. 集成美颜扩展模块 BaiduRtcBeauty.framework



2. 添加依赖

（V2.6.0 之前版本）

添加 libanakin_lite_static.a，并且TARGET->Build Settings->Other Linker Flags添加-ObjC与-force_load



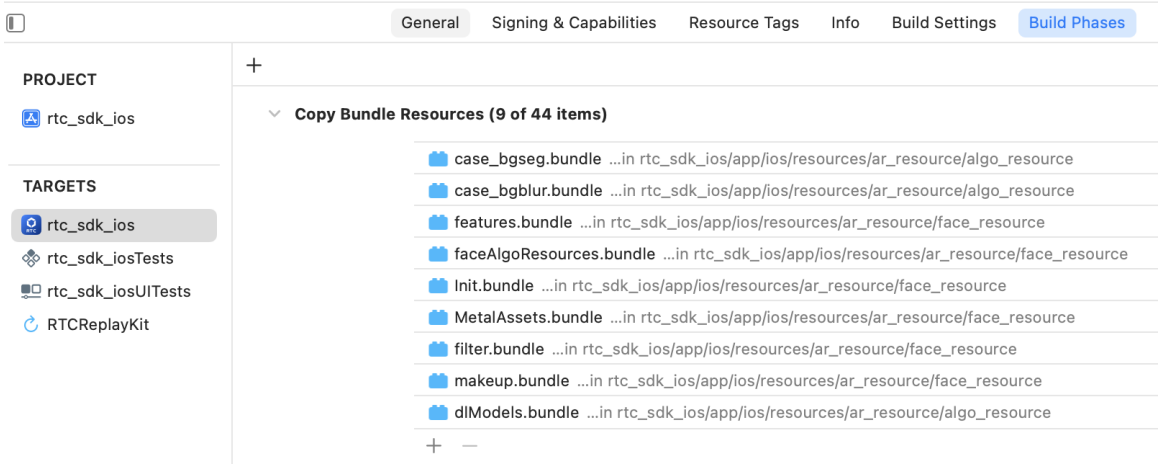
(V2.6.0 及之后版本)

主工程依赖添加 ProtocolBuffers.framework, ZipArchive.framework

3. 添加美颜能力所需资源 bundle

其中 case_bgseg.bundle, case_bgblur.bundle, dlModels.bundle **为 V2.6.0 版本新增**

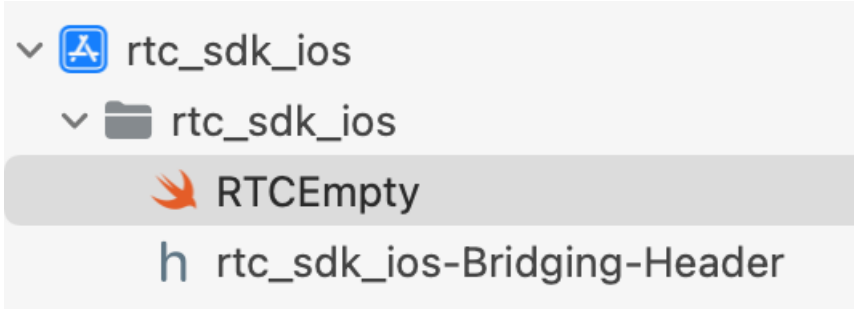
如不需要虚拟背景能力，可不添加 case_bgseg.bundle, case_bgblur.bundle



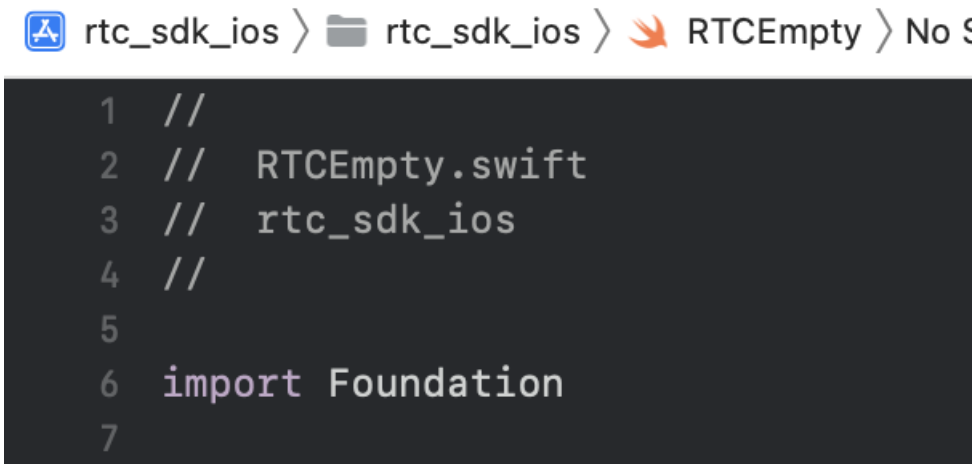
4. 工程配置

（V2.6.0 及之后版本）

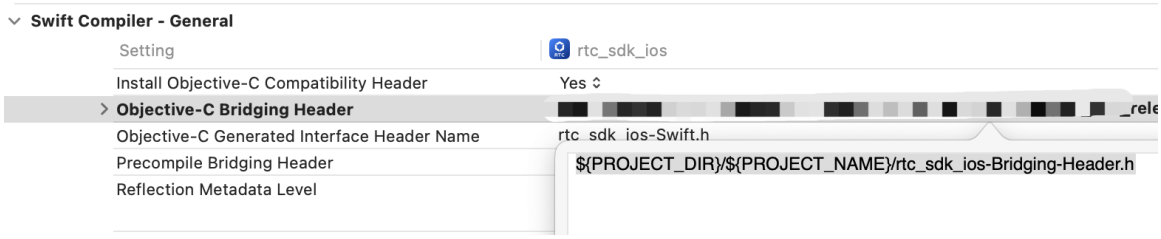
若主工程没有 swift 代码，需要配置 swift 环境以供美颜扩展使用：



在工程中创建空 swift 文件，如 RTCEmpty.swift，以使工程执行 swift 编译：

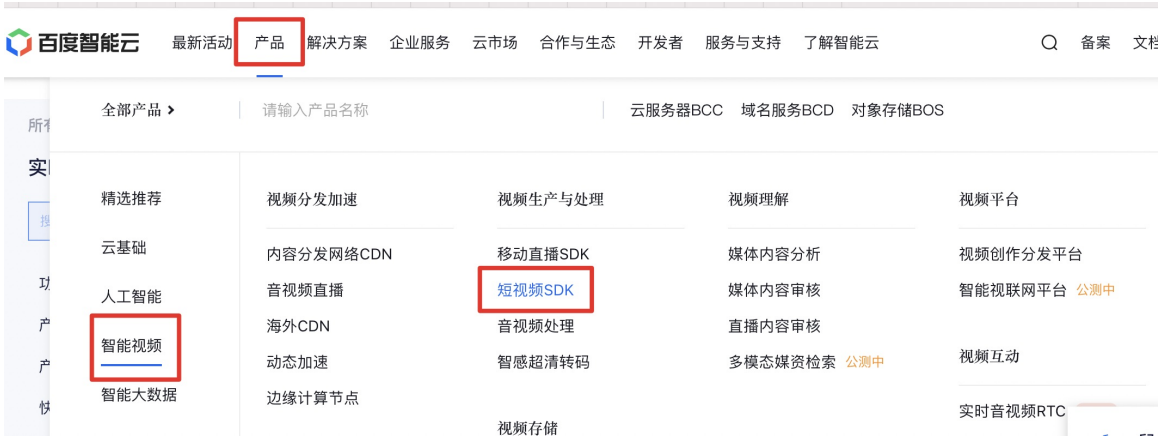


创建空的桥接头文件（创建 swift 文件后 xcode 会自动提示是否进行桥接头配置），如 rtc_ios_sdk-Bridging-Header.h，并在 Build Settings 中做相应配置：

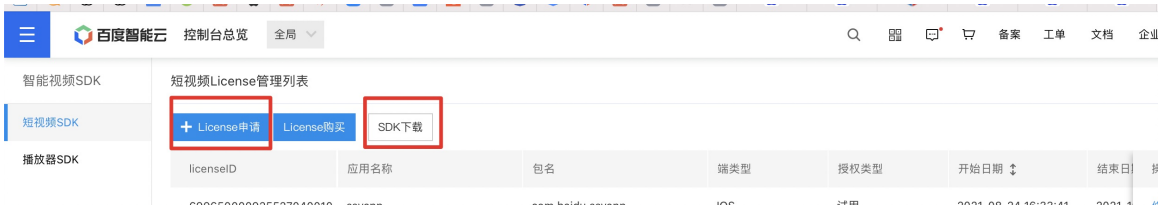


5. 使用美颜功能，需要官网申请licenseId与对应的license授权文件，申请步骤如下：

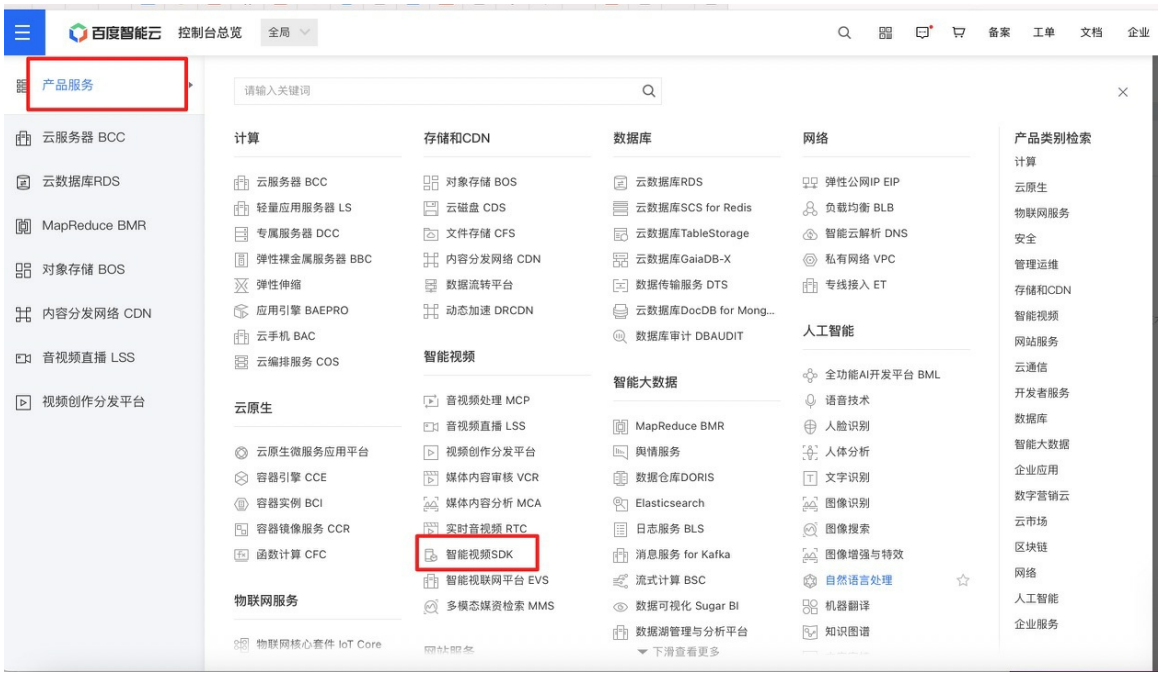
- 进入官网百度智能云，选择短视频SDK



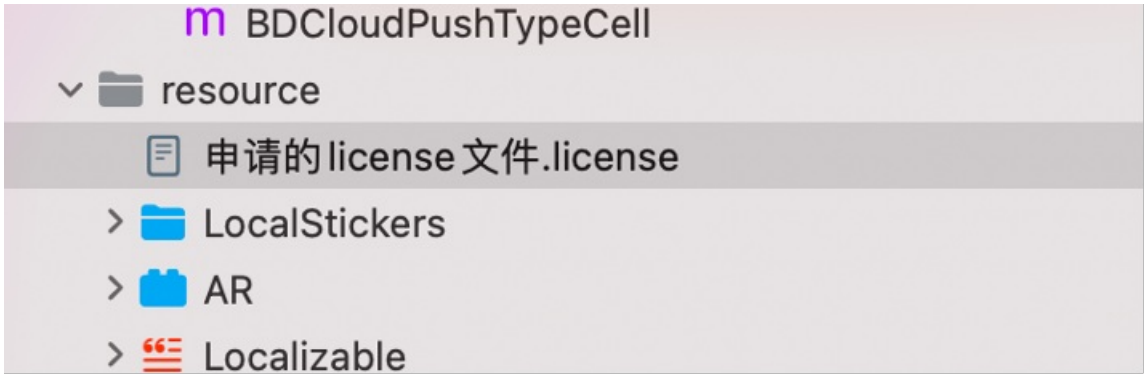
- 选择立即使用，选择license申请，填写项目信息，申请的功能权限，下载SDK



- 或直接进入license申请，选择产品服务->智能视频SDK->选择申请license，填写相关信息，获取license



- 审核成功后，下载对应的license文件，将license文件放入到工程中



- 在AppDelegate.mm中，添加如下代码，将licenseId替换自己申请的

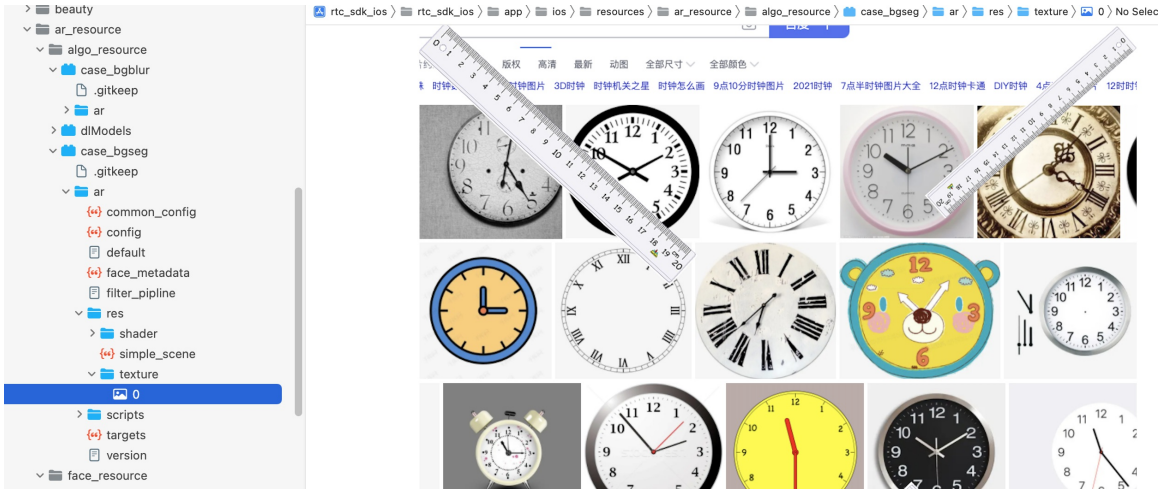
```
###### import <BaiduRtcBeauty/BDCloudAVStreamContext.h>
```

```
 并替换下方licenseID。
 */
NSString *licenseID = @"";
[[BDCloudAVStreamContext sharedInstance] verifySDKLicense:licenseID
                                        completionHandler:^(NSError * _Nonnull error)
{
    if (!error) {
        NSLog(@"授权成功!");
```

可参考短视频SDK集成申请license授权。

使用虚拟背景功能

> 虚拟背景即采集侧前处理虚拟背景，包括自定义背景、背景模糊能力
>
> 1. 资源依赖：
>
> 使用背景模糊能力，需要将 case_bgblur.bundle 添加到主工程
>
> 使用自定义背景能力，需要将 case_bgseg.bundle 添加到主工程
>
> 替换 bundle 中下图所示的图片资源，以使用自定义背景图

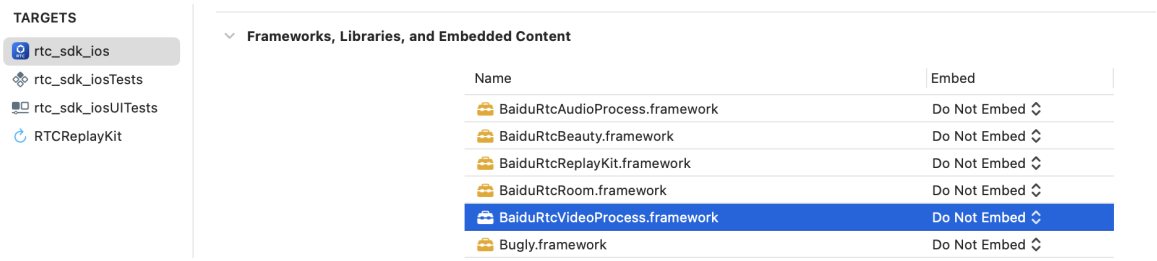


2. 接口调用：

参考 API 文档使用 BaiduRtcBeautyManager 的 setHumanSegEnabled:effectParams: 接口使用能力

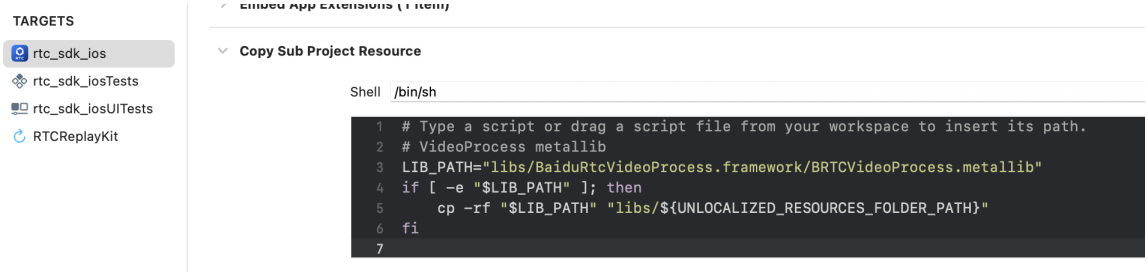
使用视频处理功能

1. 集成视频处理扩展模块 BaiduRtcVideoProcess.framework

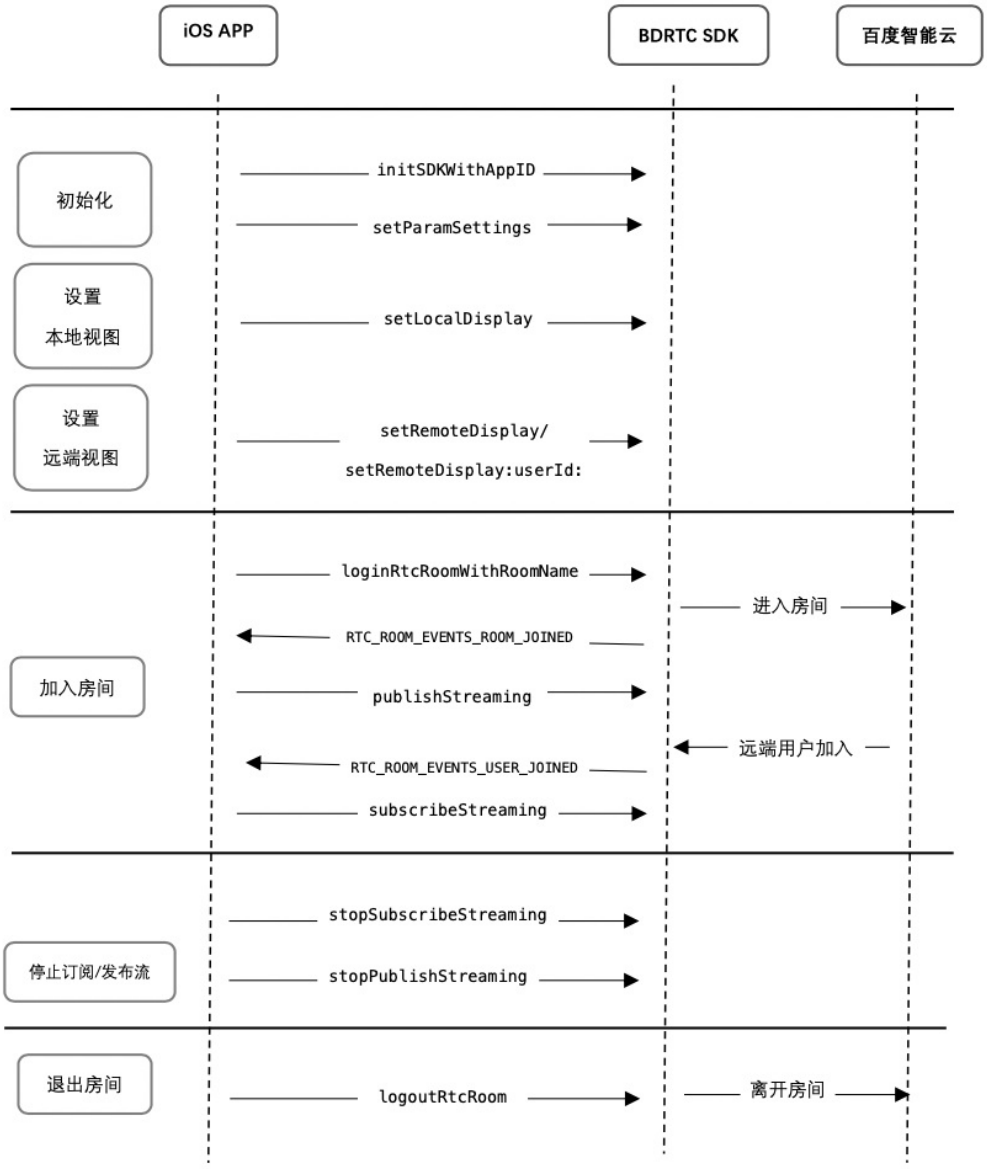


2. 添加依赖资源：

视频处理使用 Metal 能力，需手动或脚本添加扩展 framework 内部的 BRTCVideoProcess.metallib 到应用主工程中



实现音视频通话 本节介绍如何实现音视频通话。视频通话的 API 调用时序见下图：



代码示

例 1.在您的开发项目中导入头文件

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

2.定义 rtc room hanlde 变量

```
@property (nonatomic, strong) BaiduRtcRoomApi *rtcRoomApi;
```

3.初始化 sdk, 返回 rtc room handle, 初始化的时候要带上 appid，token 串，和 delegate

3.1 内部采集

```
self.rtcRoomApi = [[BaiduRtcRoomApi alloc] initSDKWithAppID:self.appId
                                                  tokenStr:self.tokenStr
                                                  delegate:self];

RtcParameterSettings *rps = [[RtcParameterSettings alloc] init];
// hasVideo 和 hasAudio 默认开启，如果需要关闭可以设置为 NO
rps.hasVideo = YES;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_HAS_VIDEO];
rps.hasAudio = YES;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_AUDIO_PARAM_SETTINGS_HAS_AUDIO];
```

3.2 自定义采集

```
 /**
 视频自定义采集
 FooCameraCapturer 是您自己的视频源实现类，需要实现 id<BaiduVideoCapturer> 协议。
 如果您的视频源是系统相机源，并且想要通过 BRTC 接口实现对相机的部分控制，则需要同时实现
id<BaiduCameraController> 协议
 */
FooCameraCapturer *capturer = [FooCameraCapturer new];
[self.rtcRoomApi setVideoCapturer:capturer];

/**
 音频自定义采集
 FooAudioExternalDevice 是您自己的音频采集源实现类，需要实现 id<BaiduRtcRoomApiAudioExternalDeviceDelegate>
协议
 */
FooAudioExternalDevice *audioDevice = [[FooAudioExternalDevice alloc] init];
rps.isEnableExternalAudioDevice = YES;
[self.rtcRoom setParamSettings:rps paramType:RTC_AUDIO_PARAM_SETTINGS_DEVICE_MODE];
[self.rtcRoom setAudioExternalDeviceDelegate:audioDevice];
```

4.音视频参数设置：

```
 // 视频采集参数
RtcVideoBaseParams *captureParams = [[RtcVideoBaseParams alloc] init];
captureParams.videoFps = 15;
captureParams.videoWidth = 720;
captureParams.videoHeight = 1280;
rps.videoCaptureParams = captureParams;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_CAPTURE_PARAMS];

// 视频采集参数 - 前置摄像头
BOOL frontCamera = [[RTCSettingsModel sharedInstance] currentUsingFrontCameraSettingFromStore];
[self.rtcRoomApi setCameraFace:frontCamera];

// 视频编码参数
RtcVideoEncodeParams *videoParams = [[RtcVideoEncodeParams alloc] init];
videoParams.videoFps = 15;
videoParams.videoWidth = 720;
videoParams.videoHeight = 1280;
videoParams.videoBitrate = 1500;

// 屏幕分享视频参数
RtcVideoEncodeParams *screenVideoParams = [[RtcVideoEncodeParams alloc] init];
screenVideoParams.videoFps = 10;
screenVideoParams.videoBitrate = 2000;
screenVideoParams.videoWidth = 1080;
screenVideoParams.videoHeight = 1920;

// 视频编码参数设置
rps.videoEncodeParams = @{
    RTC_MEDIA_TARGET_VIDEO_DEFAULT: videoParams,
    RTC_MEDIA_TARGET_VIDEO_SCREEN: screenVideoParams
};
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_TARGET_ENCODE_DICT];
```

5. 其他可选音视频配置

```
    // 设置切后台视频流推流模式
    rps.videoBgPushMode = BRTC_VIDEO_BG_PUSH_LASTFRAME;
    [self.rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_VIDEO_BG_PUSH_MODE];

    // 音频场景
    BRTCAudioProfileType audioProfile  = BRTC_AUDIO_PROFILE_HIGH_QUALITY;
    BRTCAudioScenario   audioScenario = BRTC_AUDIO_SCENARIO_DEFAULT;
    [self.rtcRoomApi setAudioProfile:audioProfile scenrio:audioScenario];

    // 使用外部渲染，外部渲染远端视频画面。若 enable 外部渲染，需实现 BaiduRtcApiRenderDelegate protocol, 并通过 setRenderDelegate 设置给 SDK.
    rps.isEnableExternalRender = YES;
    [self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_RENDER_MODE];

    // 使用声音录制, 连麦时混合后的声音抛给用户处理。若 enable 本地采集和远端声音混功能，需实现 BaiduRtcApiAudioRecordDelegate protocol, 并通过 setAudioRecordDelegate 设置给 SDK
    rps.isExportAudioRecordPlayoutMix = YES;
    [self.rtcRoomApi setParamSettings:rps paramType:RTC_AUDIO_PARAM_SETTINGS_EXPORT_RECORD_PLAYOUT_MIX];
```

6. 媒体流设置

```
  //是否自动发布流。默认开启，若不需要自动推流可设置为 NO
 rps.isAutoPublish = YES;
 [self.rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_AUTO_PUBLISH];

 //是否自动订阅流。默认开启，若不需要自动拉流可设置为 NO
 rps.isAutoSubscribe = YES;
 [self.rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_AUTO_SUBSCRIBE];
```

7. 视频显示 view 设置

```
    // 设置本地视频显示view
    [self.rtcRoomApi setLocalDisplay:_videoCallView.localVideoView];

 // 是否开启多人模式（YES：多人模式；NO：两人模式）
 rps.isMultiPlayerModel = YES;
 [self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_SESSION_MODE];

    // 设置远端用户视频显示view
    //当多人模式 rps.isMultiPlayerModel = NO 时，通过此方法设置远端画面
    [self.rtcRoomApi setRemoteDisplay:self.remoteView];

    /**
    当多人模式rps.isMultiPlayerModel = YES时，在BaiduRtcRoomDelegate代理回调里通过以下方法设置远端画面。
    [self.rtcRoomApi setRemoteDisplay:videoInfo.videoView userId:rtcStreamInfo.userId];
    */
```

6. 登录房间

```
    uint32_t tempUserId = [self getRandomNumber:1000 to:9000];
    [self.rtcRoomApi loginRtcRoomWithRoomName:self.roomName
                                       userID:tempUserId
                                  displayName:@"James"];
```

7. 发布流

```
 // 不自动发布流时，可使用该方法推流
// 注意需要等待加入房间成功，即代理方法onRoomEventUpdate回调RTC_ROOM_EVENTS_ROOM_JOINED事件之后调用
  [self.rtcRoomApi publishStreaming];
```

8. 登出房间，结束音视频通话

```
    [self.rtcRoomApi logoutRtcRoom];
```

集成Web SDK

集成SDK

根据本文指导快速集成 BRTC Web SDK 并在你自己的 app 里实现实时音视频通话。

注意：由于浏览器的安全策略对除 127.0.0.1 以外的 HTTP 地址作了限制，BRTC Web SDK 仅支持 HTTPS 协议或者 http://localhost（http://127.0.0.1），请勿使用 HTTP 协议部署你的项目。

准备环境 支持BRTC SDK的浏览器，如下：

Chrome 72+，Edge(83.0.478.56)+, Firefox 75+等

注意：如果你的网络环境部署了防火墙，请根据应用企业防火墙限制打开UDP 10010端口。

下载SDK

npm i baidurtc 或通过下文的链接获取

集成SDK

创建一个Web项目，如果你已经有一个 Web 项目了，可直接集成 SDK。

选择如下任意一种方法获取 BRTC Web SDK：

1、使用 CDN 方法获取 SDK

该方法无需下载安装包。在项目文件中，将以下代码添加到`<head>` 中的一行：

```
<script type="text/javascript" src="https://brtc-sdk.cdn.bcebos.com/npm/baidurtc@1.1.18/baidu.rtc.sdk.js" ></script>
```

2、从npm获取 SDK

最新版BRTC Web SDK。

将获得的 baidu.rtc.sdk.js 文件保存到项目文件所在的目录下。

在项目文件中，将如下代码添加到 `<style>` 上一行：

```
<script src="./baidu.rtc.sdk.js"></script>
```

现在，我们已经将 BRTC Web SDK 集成到项目中了。接下来我们要通过调用 BRTC Web SDK 提供的核心 API 实现基础的音视频通话功能。

直接在html中引入使用:

https://brtc-sdk.cdn.bcebos.com/npm/baidurtc@1.1.18/baidu.rtc.sdk.js

参考demo页面：

https://brtc.cdn.bcebos.com/brtc.html

https://redwinner.github.io/brtc/brtc.html

https://unpkg.com/baidurtc/demo/brtc.html

集成Windows SDK

开发环境

请确保开发环境满足以下技术要求：

- Visual Studio 2017 版本
- 安装Windows Softerware Development Kit - Windows 10.0.17763.132 及以上
- 安装Microsoft Visual C++ 2017 Redistributable (x64/x86)
- Windows7、Windows8、Windows10或以上版本
- 支持麦克风、摄像头采集，或拥有支持音视频功能的外部设备

SDK下载

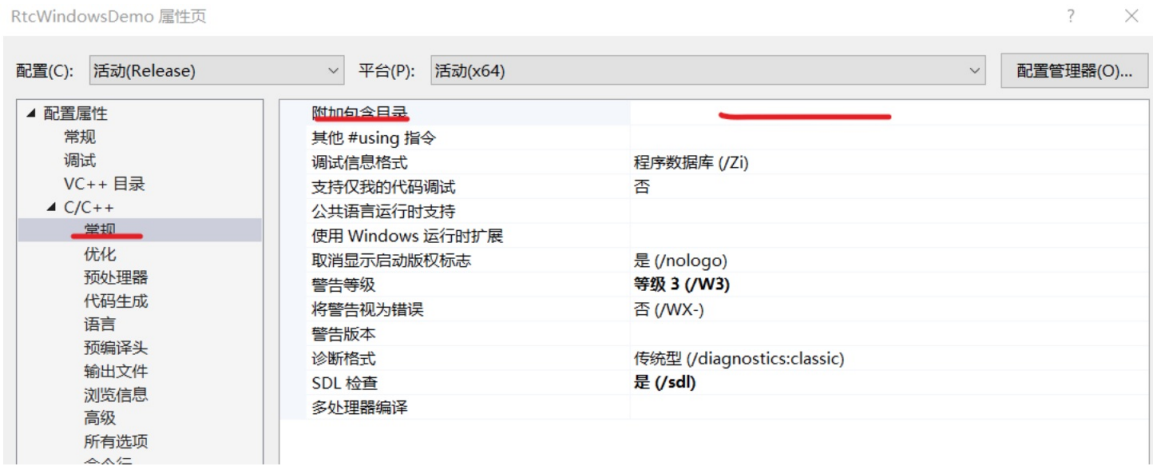
请前往RTC文档中心“下载专区>SDK&Demo下载”进行下载。

SDK目录说明

1. include目录 ------ 放置api头文件
2. libs目录 ------ 放置 RtcWindowsSdkDll.dll 和 RtcWindowsSdkDll.lib，以及证书文件 a
3. third_libs目录 ------ 是libs目录的子目录，放置依赖的第三方库 ffmpeg.dll/ffmpeg.dll.lib
   websockets_rtc.dll/websockets_rtc.lib
   libcrypto-1_1-x64.dll
   libssl-1_1-x64.dll
   libcurl.dll/libcurl.lib

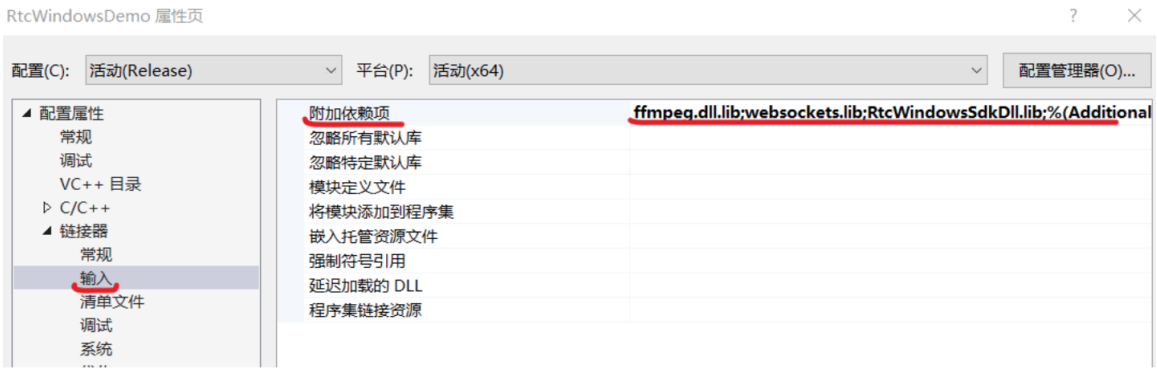
SDK集成

1. 将include目录加入到头文件搜索路径。工程目录选项右键”属性”—->”c/c++”—->”常规”—->附加包含目录 添加include目录，如下图：



2. 将libs,third_libs 目录加入到库搜索路径。工程目录选项右键”属性”—->”链接器”—->”常规”—->附加库目录 添加libs/third_libs目录，如下图：

3. 指定链接库，指定libs/third_libs目录下的xxxx.lib。工程目录选项右键”属性”—->”链接器”—->”输入”—->附加依赖项 添加xxxx.lib，如下图：

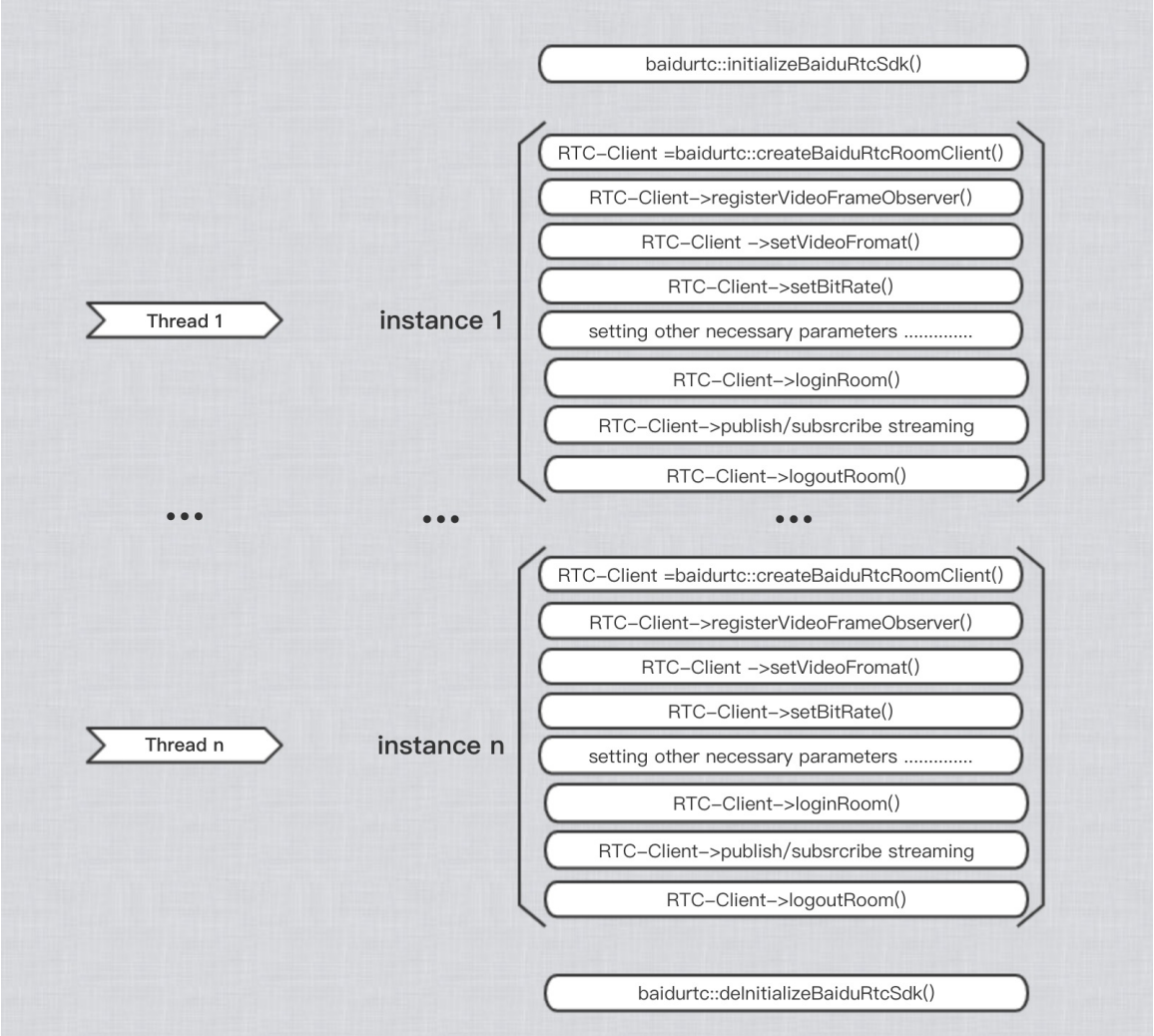


4. 将libs 和 third_libs目录下的xxxx.dll复制到二进制运行文件目录下。

完成以上步骤后，即可开始调用Windows RTC的API。

API调用流程

- c++ interface calling flow



- c# interfac calling flow

API codes使用样例

```
 ......
externalVideoCapturer_ = std::make_shared<ExternalVideoCapturer>();
    rtcRoomClient_ = baidurtc::createBaiduRtcRoomClient();
    if (rtcRoomClient_) {
        rtcRoomClient_->initSdk((char*)appid, (char*)token, (char*)cerpath);
        if (isEnableExternalVideoCapturer) {
            rtcRoomClient_->registerExternalVideoCapturer(externalVideoCapturer_);
        }
        rtcRoomClient_->registerVideoFrameObserver(videoFrameListener_, 2);
        rtcRoomClient_->registerRtcMessageListener(this);
        rtcRoomClient_->setBitRate(300000);//500kbs;
    }
......
case WM_COMMAND:
        if (button_ == reinterpret_cast<HWND>(lp)) {
            if (BN_CLICKED == HIWORD(wp)) {
                //OnDefaultAction();
                char roomId[52];
                GetWindowTextA(
                    edit1_,
                    roomId,
                    52);
                if (roomId[0] == '\0') {
                    MessageBox("提示", "房间号不能为空", true);
                    return false;
                }
                //produce random user id
                int userId = random(10000, 20000);
                std::string userId_str = int2string(userId);
                if (isPublisher) {
                    rtcRoomClient_->publishStreaming(roomId, (char*)userId_str.c_str(), true);
                }
                else {
                    rtcRoomClient_->subscribeStreaming((char*)roomid, (char*)userid, (char*)feedid);
                }
            }
        }
......
```

打包应用程序-依赖的系统库

- libgcc_s_dw2-1.dll
- libstdc++-6.dll
- libwinpthread-1.dll
- ucrtbased.dll
- vcruntime140d.dll
- vc++ 运行环境 vcredist_x64, 在打包程序选项中选中后，会自动打包到安装程序包中

集成Linux SDK

开发环境

请确保开发环境满足以下技术要求：

- GCC 5.4.0 或以上版本
- x86_64处理器

- centos7、Ubuntu 16.04或以上版本
- 开发环境需联网

SDK下载

请前往RTC文档中心“下载专区>SDK&Demo下载”进行下载。

SDK目录说明

1. sdkdemo/include目录 ------ 放置api头文件
2. bin目录 ------ 放置libbaidurtc.so 及其依赖库 和ssl证书文件a.cer
3. sdkdemo目录 ------ 放置demo代码的目录

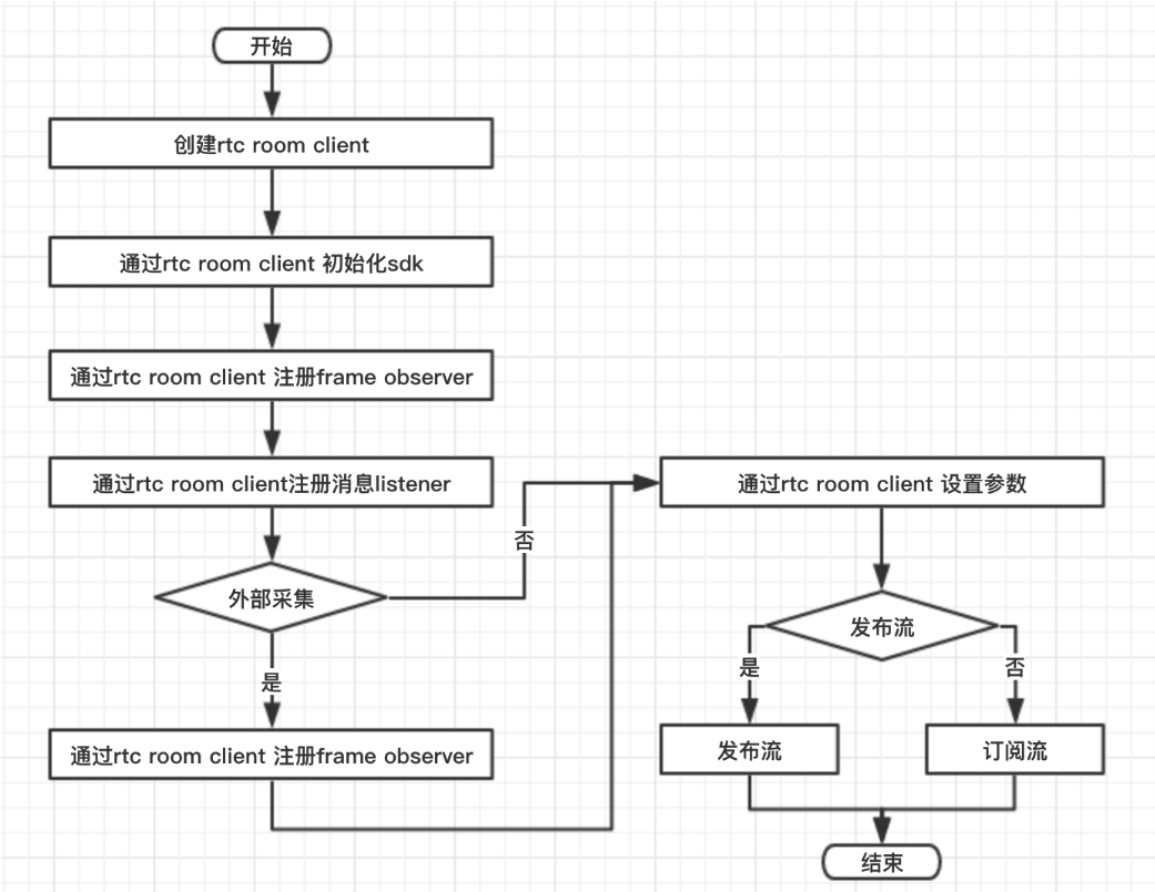
SDK集成

1. 将demo/include目录加入到Makefile头文件搜索路径。
2. 采用dlopen加载libbaidurtc.so 库， 具体参考demo源代码。
3. 将bin中的so库文件都复制到执行目录下。

完成以上步骤后，即可开始调用Linux RTC的API。

API调用流程

- 调用流程图



API codes使用样例

```
 ......
int main_push(int argc, char* argv[])
{
    void* handle = dlopen("libbaidurtc.so", RTLD_LAZY);
    if (handle == NULL) {
        fprintf(stderr, "Could not open sdk: %s\n", dlerror());
        return 1;
    }
    f_createBaiduRtcRoomClient* createClient = (f_createBaiduRtcRoomClient *)dlsym(handle,
"_ZN8baidurtc24createBaiduRtcRoomClientEv");
    if (createClient == NULL) {
        fprintf(stderr, "Could not find sdk_func: %s\n", dlerror());
        return 1;
    }
    f_getVersion *version = (f_getVersion*)dlsym(handle, "getBaiduRtcSdkVersion");

    printf("BRTC.Linux.SDK Version is: %s\n",version());

    printf("Calling API\n");

    RtcParameterSettings s;

    g_BrtcClient = createClient();
    setListener(g_BrtcClient, g_myListener);

    s.HasData = false;
    s.HasVideo = true;
    s.HasAudio = true;
    s.AudioINChannel = 1;
    s.AudioINFrequency = 16000;
    s.ImageINType = RTC_IMAGE_TYPE_JPEG;
    s.AsPublisher = true;
    s.AsListener = false;
    s.AutoPublish = true;

    g_BrtcClient->setParamSettings(&s,s.RTC_PARAM_SETTINGS_ALL);
    g_BrtcClient->setAppID("GET-FROM-BAIDU");
    g_BrtcClient->setMediaServerURL("wss://rtc.exp.bcelive.com:8989/janus");
    g_BrtcClient->setCER("../bin/a.cer");

    std::string uid = std::to_string(1234500000+random()/100000);
    g_BrtcClient->loginRoom("2131",uid.c_str(),"BRTC Linux SDK push demo","token");
......
```

Demo Codes下载

请前往RTC下载中心进行下载。

打包应用程序-依赖的文件列表

- libbaidurtc.so
- libstdc++.so.6
- libturbojpeg.so.0
- libwebsockets.so.14
- libcrypto.so.1.0.0
- libssl.so.1.0.0

- a.cer

集成macOS SDK

准备环境

- Xcode 9.0 或以上版本
- 支持 macOS 10.10 或以上版本的 macOS 设备

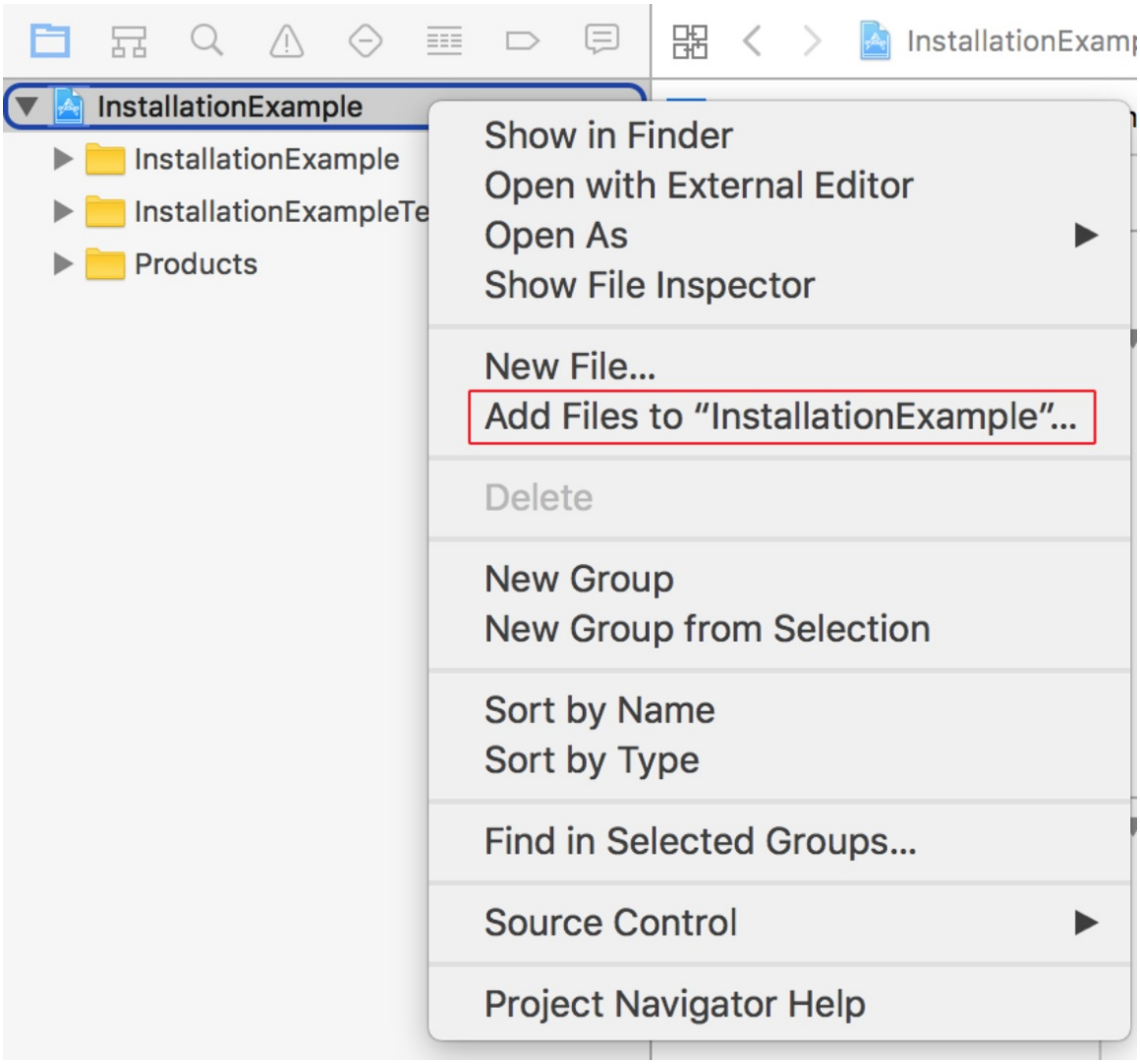
下载SDK

进入RTC文档中心，点击“下载专区>SDK&Demo下载”，即可下载macOS客户端。

集成SDK 创建一个macOS项目，若已有 macOS 项目，可以直接集成 SDK

添加 SDK 静态库文件

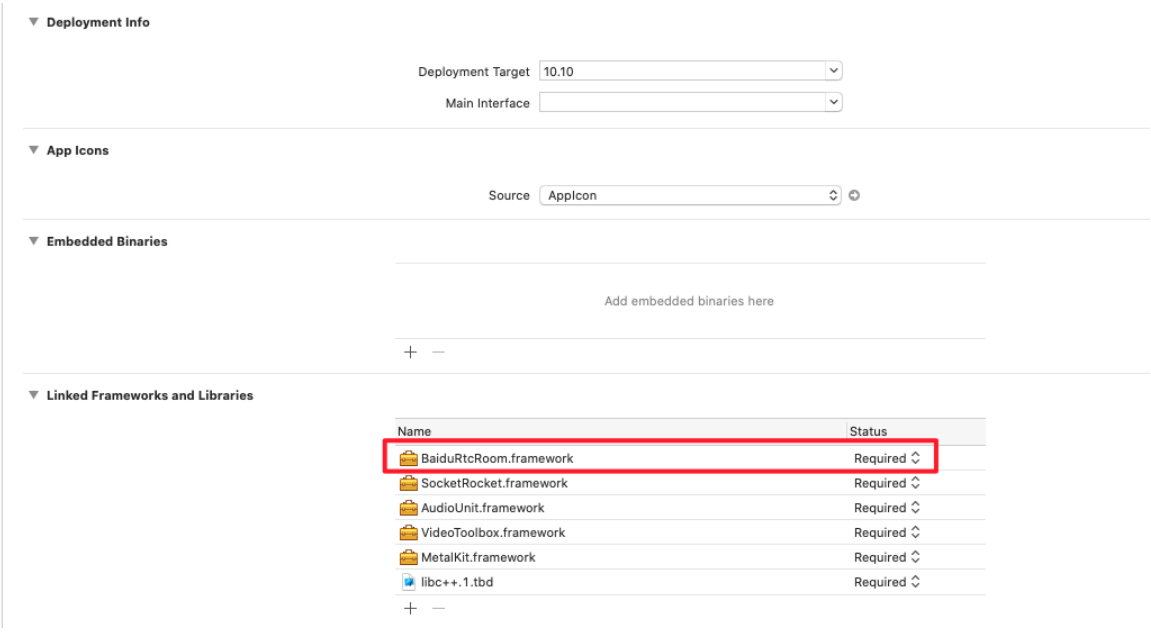
1. 将文件夹内的 BaiduRtcRoom.framework 文件复制到项目文件夹下
2. 打开 Xcode，使用 `Add Files to "xxx"（xxx 为用户的项目名）` ，添加 SDK 静态库文件到项目。



导入SDK

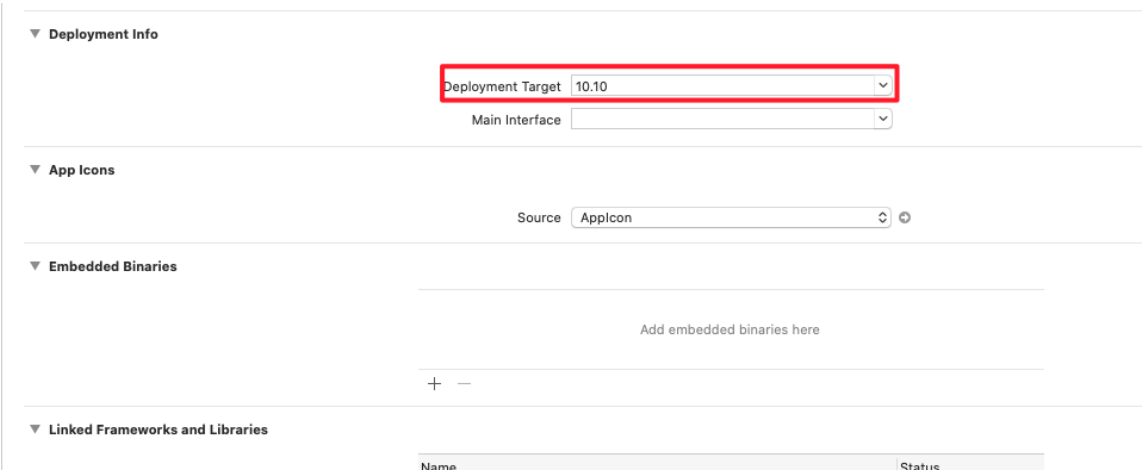
注意，在下面的设置步骤中，请选择符合开发要求的 framework 文件。

1. 打开 Xcode，选择：项目 TARGET -> General -> Link Binary With Libraries，添加 BaiduRtcRoom.framework。Status 设置为 Requied。

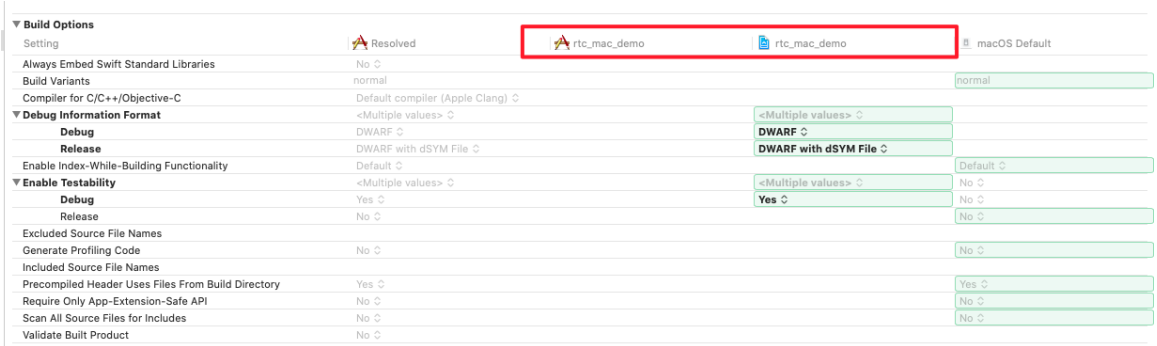


项目设置

1. 打开 Xcode，选择：项目 TARGET -> General -> Deployment Target，设置 10.10 或以上版本。

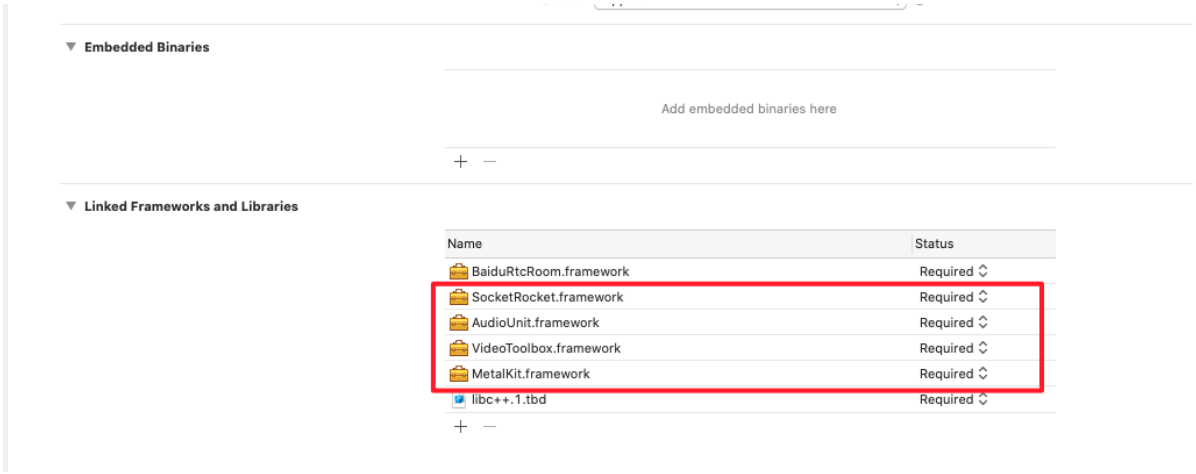


2. 继续在 Xcode 中，选择：项目 TARGET -> Build Settings -> Build Options -> Setting，设置要编译的项目 TARGET。

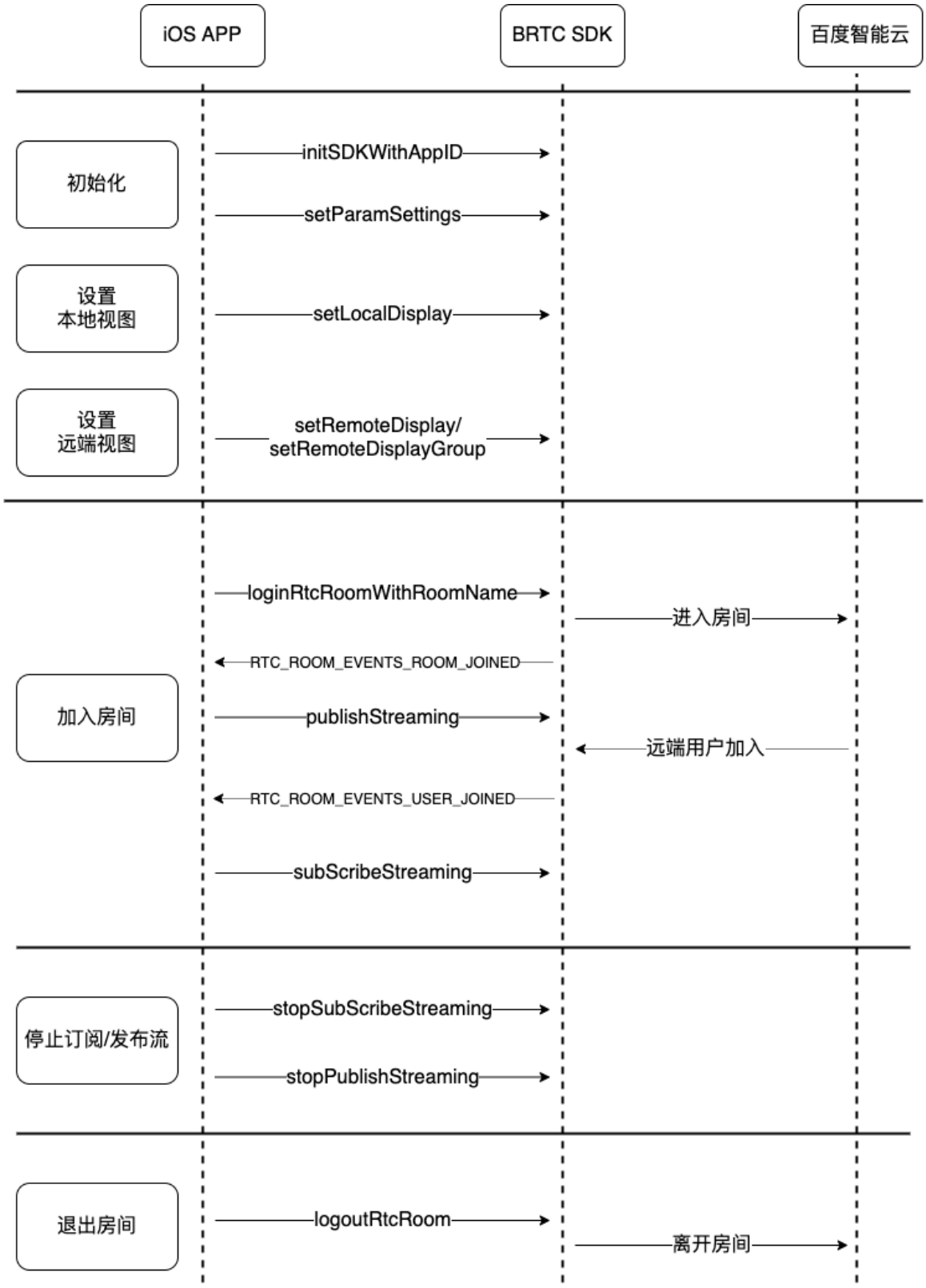


添加系统依赖库

集成 SDK 静态库，需要额外添加如下依赖库：



实现音视频通话 本节介绍如何实现音视频通话。视频通话的 API 调用时序见下图：



代码示

例 1.在您的开发项目中导入头文件

```
 #import <BaiduRtcRoom/RTCNSGLVideoView.h>
#import <BaiduRtcRoom/RTCVideoRenderer.h>
#import <BaiduRtcRoom/RTCMTLNSVideoView.h>
#import <BaiduRtcRoom/BaiduRtcRoom.h>
```

2.定义 rtc room hanlde 变量

```
@property (nonatomic, strong) BaiduRtcRoomApi *rtcRoomApi;
```

3.初始化 sdk, 返回 rtc room handle, 初始化的时候要带上 appid，token 串，和 delegate

```
_rtcRoomApi = [[BaiduRtcRoomApi alloc] initSDKWithAppID:_appId
                                        tokenStr:_tokenStr
                                        delegate:nil];
```

4.音视频参数设置：

```
RtcParameterSettings *rps = [[RtcParameterSettings alloc] init];
rps.videoFps = 29;
rps.videoWidth = 192;
rps.videoHeight = 144;
rps.videoBitrate = 300;
//使用外部采集。若 enable 外部采集，需实现 BaiduVideoCaptureFactory 和 BaiduVideoCaptureDevice 两种 protocol；其中 BaiduVideoCaptureFactory delegate 需要通过 setVideoCaptureFactory 设置给 SDK.
rps.isEnableExternalCapturer = YES;
//使用外部渲染，外部渲染远端视频画面。若 enable 外部渲染，需实现 BaiduRtcApiRenderDelegate protocol, 并通过 setRenderDelegate 设置给 SDK.
rps.isEnableExternalRender = YES;
//使用声音录制, 连麦时混合后的声音抛给用户处理。若 enable 本地采集和远端声音混功能，需实现 BaiduRtcApiAudioRecordDelegate protocol, 并通过 setAudioRecordDelegate 设置给 SDK
[_rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_ALL];
```

5. 视频显示 view 设置

```
[_rtcRoomApi setLocalDisplay:mainView.localVideoView];
[_rtcRoomApi setRemoteDisplay:mainView.remoteVideoView];
```

6. 登录房间

```
[_rtcRoomApi loginRtcRoomWithRoomName:roomId
                               userID:_userId
                          displayName:@"sj_mac"];
```

7. 发布流

```
[_rtcRoomApi publishStreaming];
```

8. 登出房间，结束音视频通话

```
[_rtcRoomApi logoutRtcRoom];
```

集成微信小程序SDK

开发环境

请确保开发环境满足以下技术要求：

- 一个经过企业认证的微信小程序账号

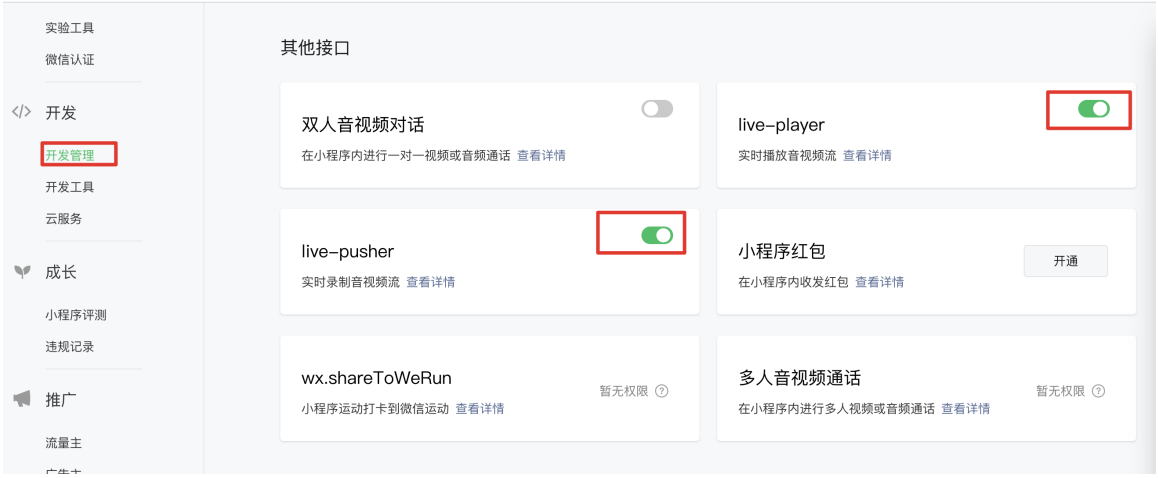
调试 RTC 微信小程序 Demo 过程中，需要使用小程序的 live-pusher 和 live-player 功能标签，用于支持音视频上行和下行（播放）能力，目前微信只向特定行业的认证企业账号开放这两个标签的使用。

- 下载并安装最新版本的微信开发者工具
- 至少一台安装有微信 App 的移动设备
- 微信 App iOS 最低版本要求：6.5.21

- 微信 App Android 最低版本要求：6.5.19
- 小程序基础库最低版本要求：1.7.0

开通小程序组件权限

进入微信公众平台 -> 【开发】 -> 【开发管理】 -> 【接口设置】 -> 【接口权限】，打开 实时播放音视频流 和 实时录制音视频流 的开关。



配置服务器域名 进入微信公众平台 -> 【开发】 -> 【开发管理】 -> 【开发设置】 -> 【服务器域名】，将如下域名配到服务器域名里

request 合法域名：以 https 开头的域名：

`https://rtc-log.cdn.bcebos.com`

socket 合法域名 ：以 wss 开头的域名：

`wss://rtc.exp.bcelive.com`

SDK下载

请前往RTC文档中心“下载专区>SDK&Demo下载”进行下载。

SDK目录说明

1. libs目录 ------ 放置RTC SDK库程序的目录
2. pages目录 ------ 放置小程序的demo代码目录
3. utils目录 ------ 放置工具代码目录

SDK集成

1. libs目录中的js文件复制到项目中
2. 使用 require 将小程序 SDK 集成到你的项目中

```
 const brtc = require('../../libs/baidu.rtc.mp.sdk.js');
var BRTCClient = brtc.BRTC;
```

完成以上步骤后，即可开始调用小程序 RTC的API。

API调用

- 小程序SDK和Web SDK 类似， 仅需使用BRTC_Start()进行登录房间，接收事件回调

API codes使用样例

```
// 获取应用实例
```

```
const brtc = require('../../libs/baidu.rtc.mp.sdk.js');
var BRTCClient = brtc.BRTC;

// 登录房间， 并接收事件回调
BRTCClient.BRTC_Start({
    server: 'wss://rtc.exp.bcelive.com/janus',
    appid: AppId,
    token: Token,
    roomname: RoomName,
    userid: UserID,
    displayname: display_name,
    aspublisher: true,
    autosubscribe: true,
    usingvideo: true,
    usingaudio: true,
    remotevideoon: function(idx) {
        console.log('remotevideoon, index:' + idx);
    },
    remotevideooff: function(idx) {
        console.log('remotevideooff, index:' + idx);
    },
    remotevideocoming: function(id, display, attribute, pullurl) {
        // pullurl 是远端视频的rtmp 地址，设置给live-player进行播放
        console.log('remotevideocoming, feedid:' + id + ' display:' + display
        + ' a:' + attribute + ' Pull URL is:' + pullurl);
        // rtmp://endpoint/%a/%rn/%f?sdkJoined=true&userId=%u&token=%t
        that.setData({
            PlayUrl: pullurl
        });
        that.remotevideo.stop();
        that.remotevideo.play();
    },
    remotevideoleaving: function(id) {
        console.log('remotevideoleaving, feedid:' + id);
    },
    userevent_joinedroom: function(id, display, attribute) {
        console.log('userevent_joinedroom id: ' + id + ', display: ' + display + ', attribute:' + attribute);
    },
    userevent_leavingroom: function(id, display) {
        console.log('userevent_leavingroom id: ' + id + ', display: ' + display);
    },
    success: function(pushurl) {
        // pushurl 是本端视频的rtmp 推流地址，设置给live-pusher进行推流
        console.log('success, Push URL is:' + pushurl);
        // rtmp://endpoint/%a/%rn/%u?sdkJoined=true&token=%t
        that.setData({
            PushUrl: pushurl
        });
        that.localvideo.start();
    },
    localvideopublishing: function() {

    },
    localvideopublished_ok: function() {

    },
    remotevideo_closed:  function(feedid) {
        console.log('remotevideo_closed(feedid: ' + feedid + ') by server, please do SubScribing again');
    },
    error: function(error) {
        wx.showModal({
            title: '提示',
```

```
            content: '登录失败,请检查AppID，Token等参数,谢谢!',
            success: function (res) {
                if (res.confirm) {
                }
            }
        });
    },
    destroyed: function(error) {

    },
    onlocalstream: function(stream,name) {
        //local stream for display sonic wave
        console.log('onlocalstream name: ' + name);
    },
    onlocalstream_end: function(name) {
        //end event of local stream
        console.log('onlocalstream_end name: ' + name);
    },
    onmessage: function(msg) {
        // event onmessage.
        console.log('onmessage id: ' + msg.id + ' data: '+ msg.data);
    },
    onattribute: function(a) {
        // event onattribute
        console.log('onattribute id: ' + a.id + ' attribute: '+ a.attribute);
    },
    reportmonitordata: false,
});

// 发送自定义消息等API
BRTCClient.BRTC_SendMessageToUser('hello word');

// 登出房间
BRTCClient.BRTC_Stop();
```

Demo Codes下载

请前往RTC下载中心进行下载。

打包应用程序-依赖的文件列表

- libs/baidu.rtc.mp.sdk.js

集成FreeRtos SDK

开发环境 请确保开发环境满足以下技术要求：

1. Windows7、Windows8、Windows10或以上版本
2. 安装ARM DS-5 v5.26.2或以上版本。
3. 安装msys2-x86_64-20210105或以上版本。
4. 安装ActivePerl5.32或以上版本。
5. 安装python3.8.7或以上版本。
6. 硬件设备：基于ASR3601的小度手表。

SDK下载 进入RTC文档中心，点击“下载专区>SDK&Demo下载”

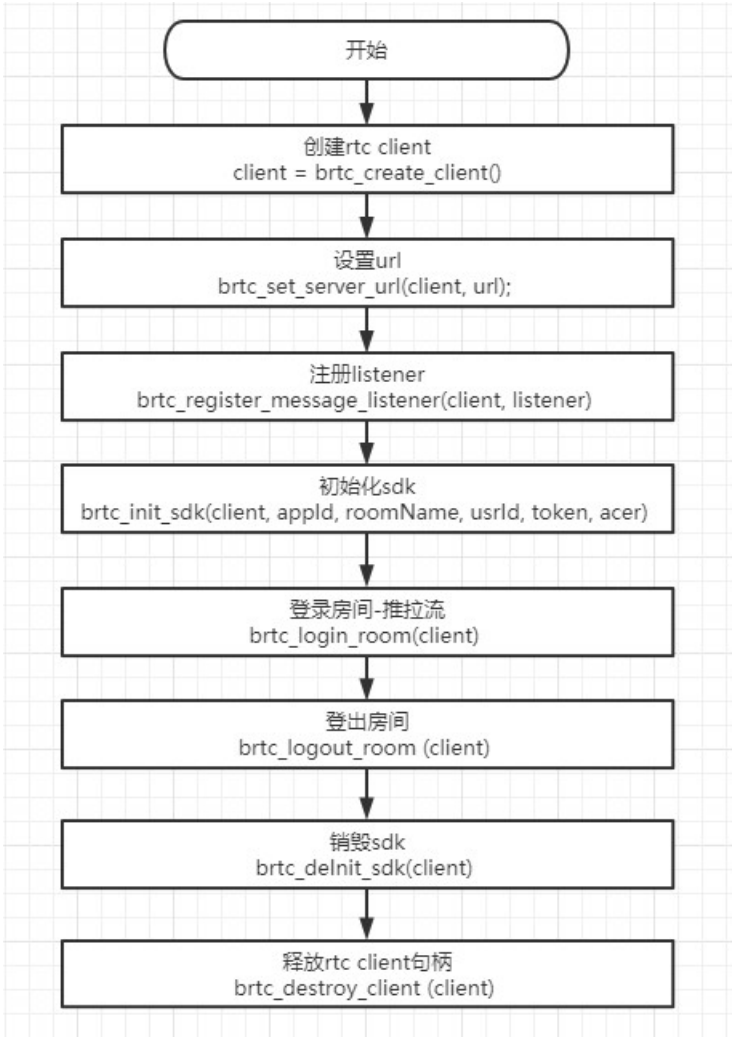
SDK目录说明

1. BRTC.FreeRtos.SDKV0.0.1/include目录 ------ 放置api头文件
2. BRTC.FreeRtos.SDKV0.0.1/lib目录 ------ 放置音视频静态库libbrtc.a
3. BRTC.FreeRtos.SDKV0.0.1/src目录 ------ 放置api调用的参考demo代码

SDK集成

1. 将BRTC.FreeRtos.SDKV0.0.1/include目录加入到CMakeLists.txt中的头文件搜索路径。
2. 将BRTC.FreeRtos.SDKV0.0.1/lib目录下的libbrtc.a放到执行目录下。
3. application调用sdk api流程可参考BRTC.FreeRtos.SDKV0.0.1/src/demo.c源代码。
4. 小度手表app调用sdk api流程可参考BRTC.FreeRtos.SDKV0.0.1/src/lv_app源代码。
5. 完成以上步骤后，即可开始调用FreeRtos RTC的API。

API调用流程 1. API 调用流程图



2. API codes使用样例

`

```
 bool rtc_joined_room = false;
static void OnRtcMessage(RtcMessage* msg)
{
 printf("myListener got Message: %d\n", msg->msgType);
 switch (msg->msgType) {
 case RTC_MESSAGE_ROOM_EVENT_LOGIN_OK: {
  printf("Login ok!\n");
```

```
 } break;
 case RTC_MESSAGE_ROOM_EVENT_LOGIN_ERROR: {
  printf("Login error!\n");
 } break;
 case RTC_MESSAGE_ROOM_EVENT_LOGIN_TIMEOUT: {
  printf("Login timeout!\n");
 } break;
 case RTC_MESSAGE_ROOM_EVENT_CONNECTION_ERROR: {
  printf("Connection error!\n");
 } break;
 case RTC_MESSAGE_ROOM_EVENT_REMOTE_COMING: {
  printf("Feed coming!\n");
 } break;
 case RTC_ROOM_EVENT_ON_PUBLISHER_JOINED_ROOM: {
  printf("Publisher joined room!\n");
  rtc_joined_room = true;
 } break;
 case RTC_ROOM_EVENT_ON_USER_MESSAGE: {
  printf("user msg callback:%s\n", msg->extra_info);
 } break;
 case RTC_ROOM_EVENT_ON_USER_JOINED_ROOM: {
  printf("User joined room!\n");
 } break;
 case RTC_MESSAGE_MEDIA_SIGNAL_ESTABLISH_OK: {
  printf("Media signal established ok!\n");
 } break;
 case RTC_MESSAGE_MEDIA_SIGNAL_ESTABLISH_ERROR: {
  printf("Media signal established error!\n");
 } break;
 case RTC_MESSAGE_MEDIA_SIGNAL_RECEIVE_FISRT_AUDIO: {
  printf("It received audio yet!\n");
 } break;
 case RTC_MESSAGE_MEDIA_SIGNAL_RECEIVE_FISRT_VIDEO: {
  printf("It received video yet!\n");
 } break;
 case RTC_MESSAGE_MEDIA_SIGNAL_NO_RTP_STREAM_TIMEOUT: {
  printf("Did't receive rtp stream because of timeout!\n");
 } break;
 case RTC_MESSAGE_ROOM_EVENT_REMOTE_LEAVING: {
  printf("Feed leaving!\n");
 } break;
 case RTC_MESSAGE_STATE_STREAM_UP: {
  printf("Stream up, send/got video now\n");
 } break;
 default:
  break;
 }
}
void brtc_video_call_put_image(int width, int height, const unsigned char *src,
    int is_semi, int img_type)  //回调解码后的视频数据显示
{
    printf("put image");
}

const char *s_url = "ws://rtc2.exp.bcelive.com:8186/janus";
static void *client = NULL;
int main(int argc, char* argv[])
{
 bool ret =false;
 const char appId[] = "APPID_GET_FROM_BAIDU"; //please get a appId from baidu
 const char token[] = "no_token";
 const char room_name[] = "8000";
 const char user_id[] = "1008";
```

```
 const char user_id[] = "1008";
 if(client == NULL) {
  client = brtc_create_client();
  if (client == NULL) {
   printf("brtc_create_client failed\n");
   return -1;
  }
 }

 if (client) {
  brtc_register_message_listener(client, OnRtcMessage);
  brtc_set_server_url(client, s_url);
  ret = brtc_init_sdk(client, appId, room_name, user_id, token, NULL);
  if (!ret) {
   printf("brtc_init_sdk failed\n");
   return -1;
  }
  while (!rtc_joined_room);

  ret = brtc_login_room(client);
  if (!ret) {
   printf("brtc_login_room failed\n");
   brtc_deInit_sdk(client);
   brtc_destroy_client(client);
   if(client)
    client = NULL;
   return -1;
  }

  while (true) {
   printf("Please input q to Quit.\n");
   char commandid[512];
   memset(commandid, 0, sizeof(commandid));
   int ret = scanf("%s", commandid);
   printf("scanf %d.\n", ret);
   if (ret < 0) {
    continue;
   }
   if (commandid[0] == 'q') {
            break;
         }
  }

  ret = brtc_logout_room(client);
  if (!ret) {
   printf("brtc_logout_room failed\n");
   brtc_deInit_sdk(client);
   brtc_destroy_client(client);
   if(client)
    client = NULL;
   return -1;
  }
  brtc_deInit_sdk(client);
  brtc_destroy_client(client);
  if(client)
   client = NULL;
  rtc_joined_room = false;
 }

 return 0;
}
```

打包应用程序-依赖的文件列表

- libbrtc.a

快速跑通RTC Demo

iOS 快速接入

iOS 快速实现音视频通话

实时音视频通话能够拉近人与人之间的距离，帮助你的 APP 提高用户黏性。 本文介绍如何通过少量代码集成百度 RTC SDK ，快速实现高质量、低延迟的视频通话功能。

实现视频通话的步骤如下：

1. 初始化

调用 initWithAppID初始化SDK

2.设置显示视图

设置本地与远端显示视图

3.加入房间

调用loginRtcRoomWithRoomName 加入房间

说明：在 APP ID 一致的前提下，传入相同房间名的用户会进入同一个房间通话。

4.发布/订阅

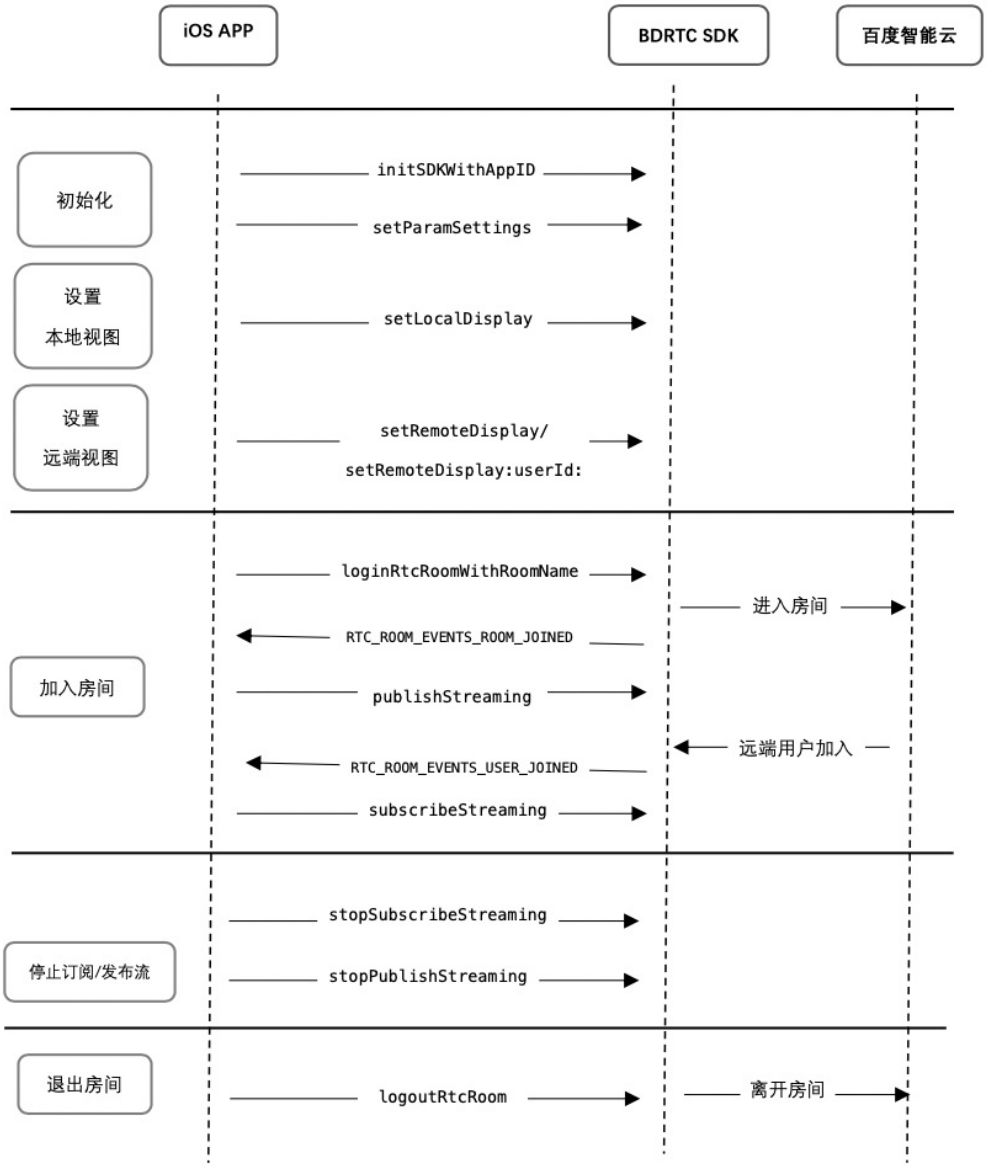
加入房间后，两位主播可以手动发布音视频并互相订阅

如果设置自动发布 或 自动订阅 ，则自动发布/订阅音视频

5.退出房间

挂断，退出房间

视频通话的 API 调用时序见下图：



前期准备

- Xcode 9.0 或以上版本
- 支持 iOS 9.0 或以上版本的 iOS 设备
- 两台支持音视频功能的模拟器或真机
- 申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html
- 下载精简SDK https://doc.bce.baidu.com/bce-documentation/RTC/rtc_iOS_release_fastdemo_20230407.zip
- 或下载SDK https://cloud.baidu.com/doc/RTC/s/Hk29sqi42
- 可以访问互联网的计算机。确保你的网络环境没有部署防火墙，否则无法正常使用百度服务。

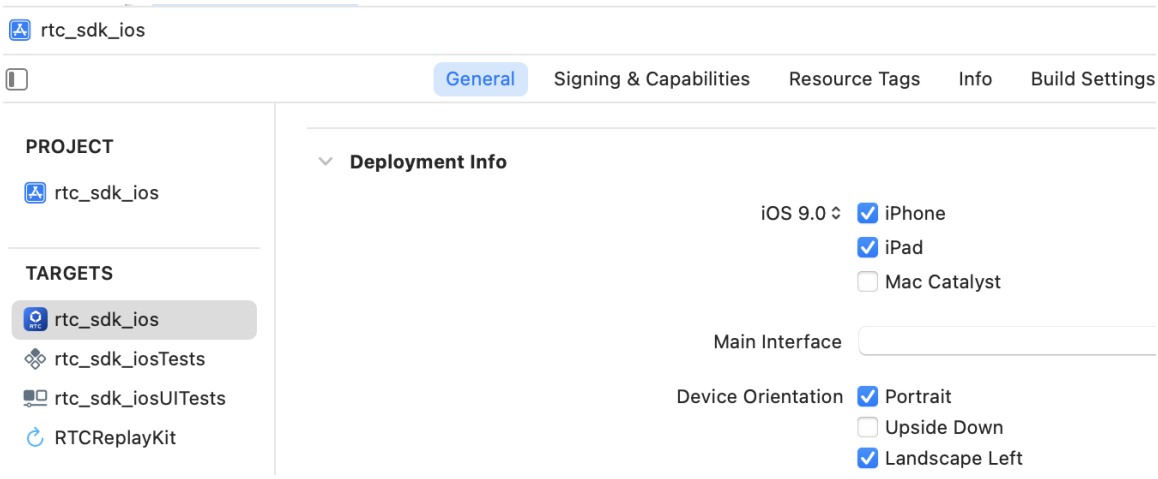
在 Xcode 中进行以下操作，在你的 app 中实现视频通话功能：

1. 创建一个新的项目，**Application** 选择 **App**，**Interface** 选择 **Storyboard**，**Language** 选择 **Objective_C**。如果你没有添加过开发团队信息，会看到 **Add account…** 按钮。点击该按钮并按照屏幕提示登入 **Apple ID**，点击 **Next**，完成后即可选择你的 Apple 账户作为开发团队。
2. 为你的项目设置自动签名。
3. 设置部署你的 app 的目标设备。

说明：使用已有的项目也可以集成SDK。

项目设置

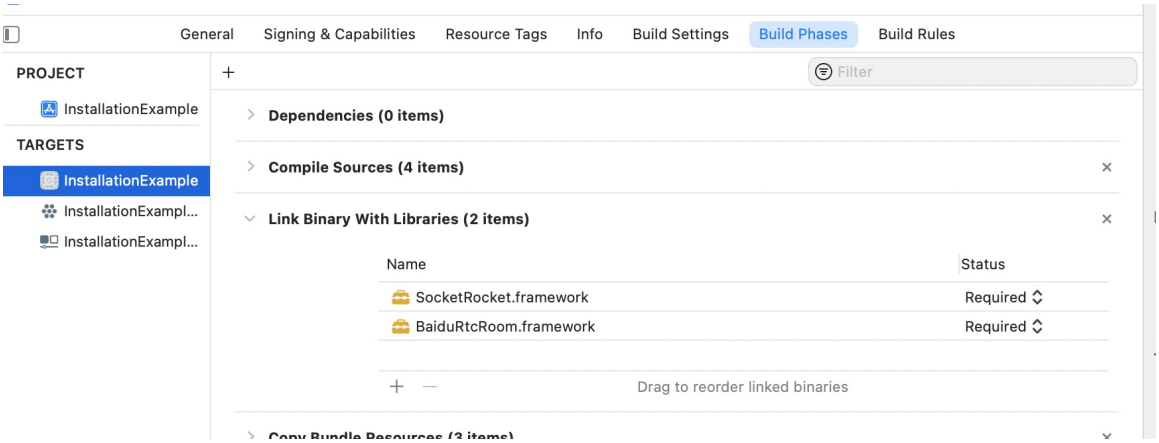
1. 打开 Xcode，选择：项目 TARGET -> General -> Deployment Target，设置 9.0 或以上版本。

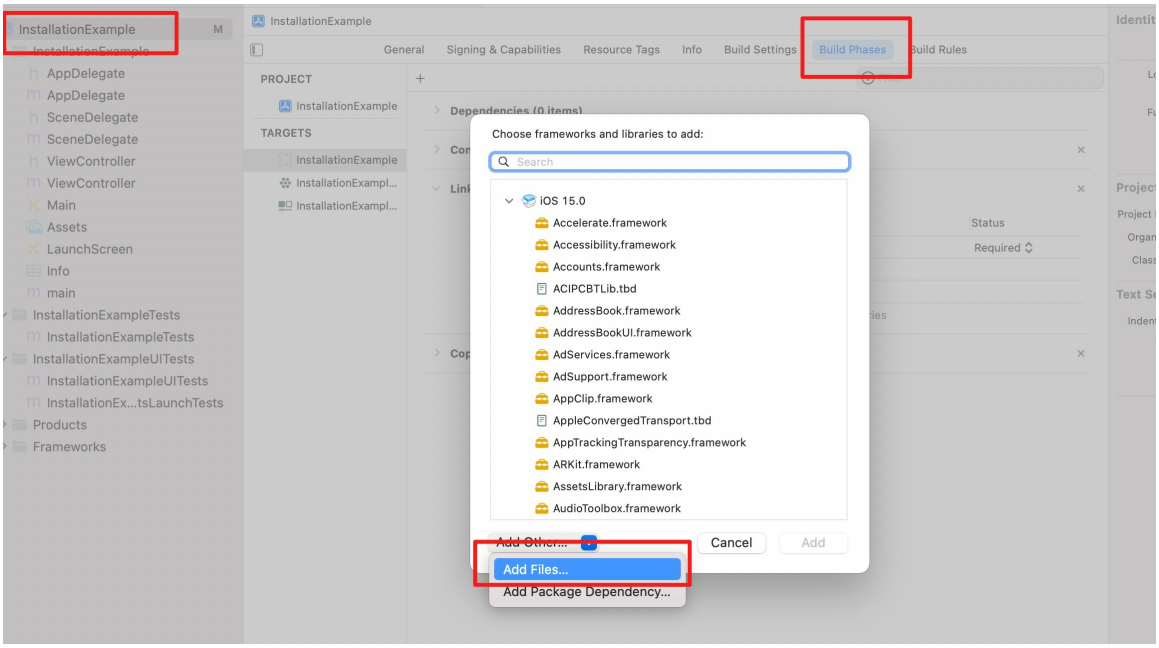


添加 SDK 静态库文件

1. 将下载的SDK中lib文件夹内的文件复制到项目文件夹下

2. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，将 BaiduRtcRoom.framework SocketRocket.framework添加 SDK 静态库文件到项目。

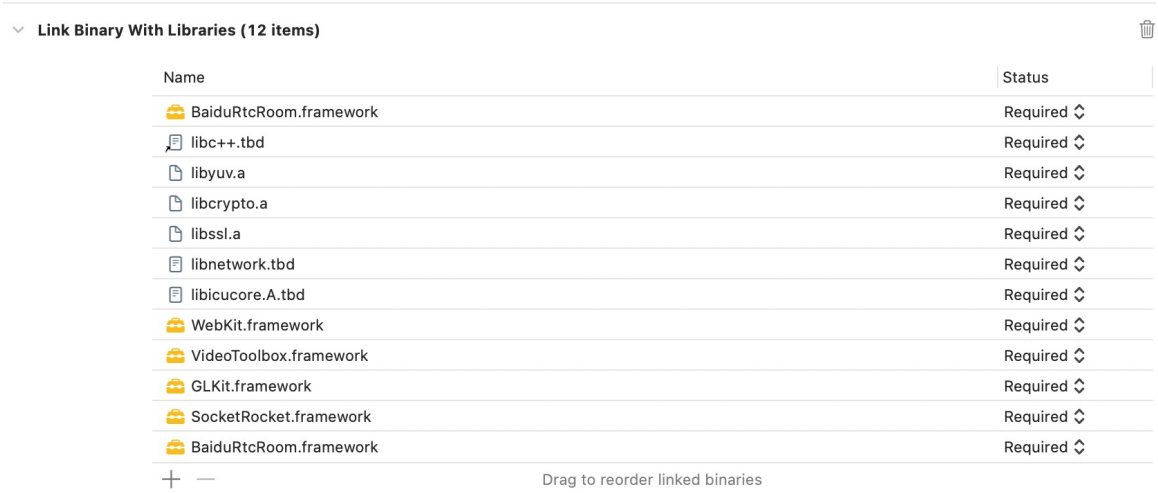




3. 选择：项目 TARGET -> Build Phases ->Link Binary With Libraries，将libyuv.a，libcrypto.a，libssl.a添加到项目中

添加系统依赖库

将系统库libc++.tbd，libnetwork.tbd，libicucore.A.tbd，WebKit.framework，VideoToolbox.framework，GLKit.framework添加到项目中



说明 ：

BaiduRtcRoom 为 BRTC SDK 主依赖库，必须添加；

SocketRocket 为三方依赖库，必须添加；

其中libyuv.a，libcrypto.a，libssl.a为百度内部依赖库，必须添加；

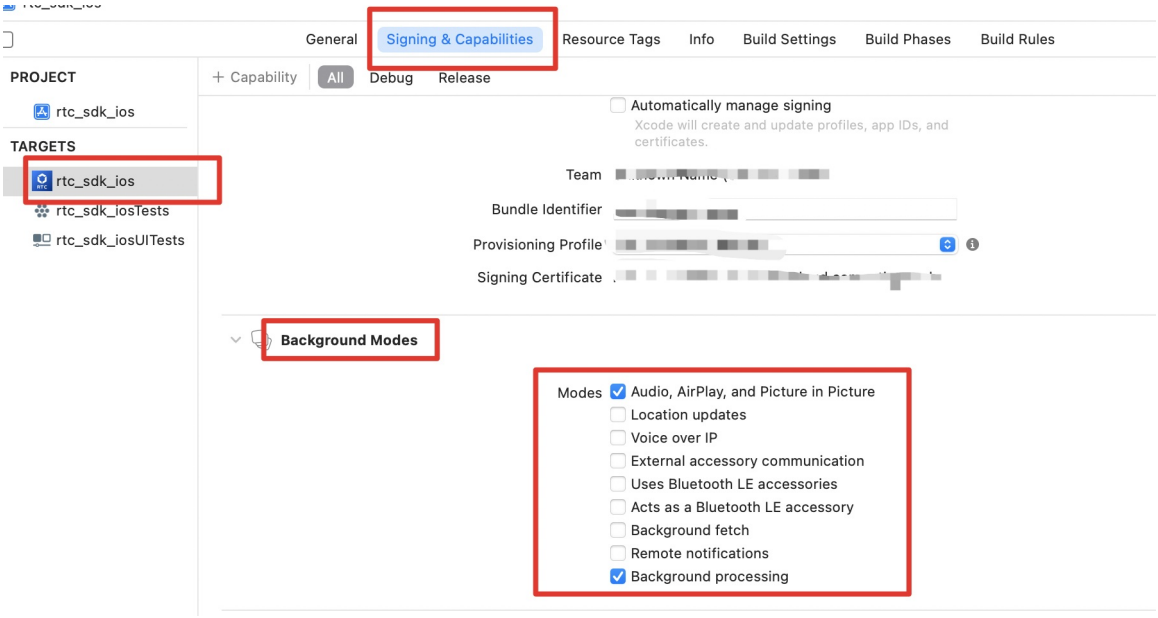
其余为系统依赖库，必须添加。

添加相机/麦克风权限

在Info.plist文件中添加Privacy相机/麦克风权限

添加Background Modes

在Target -> Signing & Capabilities -> Background Modes中添加如下Modes



代码示例

1.以新建项目为例，在ViewController.m文件中导入头文件

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

2.定义 rtc room hanlde 变量

```
@property (nonatomic, strong) BaiduRtcRoomApi *rtcRoomApi;
```

3.初始化 sdk, 返回 rtc room handle, 初始化的时候要带上 appid，token 串，和 delegate

3.1 内部采集

```
self.rtcRoomApi = [[BaiduRtcRoomApi alloc] initSDKWithAppID:self.appId
                                                  tokenStr:self.tokenStr
                                                  delegate:self];
```

3.2 自定义采集

```
/**
  视频自定义采集
 FooCameraCapturer 是您自己的视频源实现类，需要实现 id<BaiduVideoCapturer> 协议。
 如果您的视频源是系统相机源，并且想要通过 BRTC 接口实现对相机的部分控制，则需要同时实现
id<BaiduCameraController> 协议
 */
FooCameraCapturer *capturer = [FooCameraCapturer new];
[self.rtcRoomApi setVideoCapturer:capturer];

  /**
   音频自定义采集
   FooAudioExternalDevice 是您自己的音频采集源实现类，需要实现 id<BaiduRtcRoomApiAudioExternalDeviceDelegate>
协议
   */
  FooAudioExternalDevice *audioDevice = [[FooAudioExternalDevice alloc] init];
  rps.isEnableExternalAudioDevice = YES;
  [self.rtcRoom setParamSettings:rps paramType:RTC_AUDIO_PARAM_SETTINGS_DEVICE_MODE];
  [self.rtcRoom setAudioExternalDeviceDelegate:audioDevice];
```

4.音视频参数设置：

```
 RtcParameterSettings *rps = [[RtcParameterSettings alloc] init];
//是否开启多人模式（YES：多人模式；NO：两人模式）
rps.isMultiPlayerModel = YES;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_SESSION_MODE];

//以下参数按需设置
// 视频采集参数
RtcVideoBaseParams *captureParams = [[RtcVideoBaseParams alloc] init];
captureParams.videoFps = 15;
captureParams.videoWidth = 720;
captureParams.videoHeight = 1280;
rps.videoCaptureParams = captureParams;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_CAPTURE_PARAMS];

// 视频采集参数 - 前置摄像头
BOOL frontCamera = YES;
[self.rtcRoomApi setCameraFace:frontCamera];

// 视频编码参数
RtcVideoEncodeParams *videoParams = [[RtcVideoEncodeParams alloc] init];
videoParams.videoFps = 15;
videoParams.videoWidth = 720;
videoParams.videoHeight = 1280;
videoParams.videoBitrate = 1500;

// 视频编码参数设置
rps.videoEncodeParams = @{
    RTC_MEDIA_TARGET_VIDEO_DEFAULT: videoParams,
};
[self.rtcRoomApi setParamSettings:rps paramType:RTC_VIDEO_PARAM_SETTINGS_TARGET_ENCODE_DICT];

//是否自动发布流
rps.isAutoPublish = YES;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_AUTO_PUBLISH];

//是否自动订阅流
rps.isAutoSubscribe = YES;
[self.rtcRoomApi setParamSettings:rps paramType:RTC_PARAM_SETTINGS_AUTO_SUBSCRIBE];
```

5.视频显示 view 设置

```
@interface ViewController () <
    BaiduRtcRoomDelegate, RTCRemoteVideoViewDelegate
>
@property (nonatomic, strong) BaiduRtcRoomApi *rtcRoomApi;
//声明本地与远端用户显示的视图
@property (nonatomic, strong) RTCLocalVideoView *localView;
@property (nonatomic, strong) RTCRemoteVideoView *remoteView;
@end

- (void)viewDidLoad {
    //初始化本地用户显示视图
    self.localView = [[RTCLocalVideoView alloc] initWithDelegate:self];
    self.localView.videoView.frame = CGRectMake(0, 0, 100, 200);
    [self.view addSubview:self.localView.videoView];
    //初始化远端用户显示视图
    self.remoteView = [[RTCRemoteVideoView alloc] initWithDelegate:self];
    self.remoteView.videoView.frame = CGRectMake(0, 120, 100, 200);
    [self.view addSubview:self.remoteView.videoView];

    //设置本地视频显示view
    [self.rtcRoomApi setLocalDisplay:self.localView];

    //设置远端用户视频显示view
    //当多人模式rps.isMultiPlayerModel = NO时，通过此方法设置远端画面
    [self.rtcRoomApi setRemoteDisplay:self.remoteView];

    /**
    当多人模式rps.isMultiPlayerModel = YES时，在BaiduRtcRoomDelegate代理回调里通过以下方法设置远端画面。
    可以在提供的demo里面搜索此方法
    [self.rtcRoomApi setRemoteDisplay:videoInfo.videoView userId:rtcStreamInfo.userId];
    */

}
```

6. 登录房间

```
uint32_t tempUserId = [self getRandomNumber:1000 to:9000]; //随机生成数或者输入读取
[self.rtcRoomApi loginRtcRoomWithRoomName:self.roomName
                                   userID:tempUserId
                              displayName:@"James"];
```

7. 发布流

```
// 不自动发布流时，可使用该方法推流
// 注意需要等待加入房间成功，即代理方法onRoomEventUpdate回调RTC_ROOM_EVENTS_ROOM_JOINED事件之后调用
[self.rtcRoomApi publishStreaming];
```

8. 订阅流

```
// 不自动订阅流时，可使用该方法拉流
// 注意需要等待加入房间成功，即代理方法onRoomEventUpdate回调RTC_ROOM_EVENTS_ROOM_JOINED事件之后调用
[self.rtcRoomApi subsribeStreaming:@[@(123)]];
```

9. 登出房间，结束音视频通话

```
[self.rtcRoomApi logoutRtcRoom];
```

Android快速接入

Android快速实现音视频通话

实时视频通话能够拉近人与人之间的距离，帮助你的 APP 提高用户黏性。 本文介绍如何通过少量代码集成百度 RTC SDK ，快速实现高质量、低延迟的视频通话功能。

实现视频通话的步骤如下：

1. 加入房间 调用 initWithAppID初始化SDK，并loginRtcRoomWithRoomName 加入房间。在 APP ID 一致的前提下，传入相同房间名的用户会进入同一个房间通话。
2. 发布 / 订阅 加入房间后，两位主播可以手动发布音视频并互相订阅。 如果设置自动发布 或 自动订阅 ，则自动发布/订阅音视频。

1. 前期准备 在实现 Android 平台音视频 Demo 之前，你需要有以下准备：

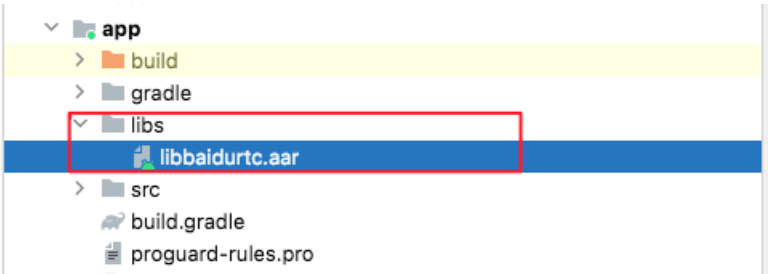
- Android Studio 本文档基于2021.2.1 Patch 2 版本
- Android SDK
- gradle
  - 本文档基于 com.android.tools.build:gradle:3.4.1
  - gradle-6.1.1-all.zip
- 申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html
- 下载SDK https://cloud.baidu.com/doc/RTC/s/Hk29sqi42
- 两台运行 Android 4.4 或以上版本的移动设备。
- 可以访问互联网的计算机。确保你的网络环境没有部署防火墙，否则无法正常使用百度服务。
- 快速Demo 下载地址：https://doc.bce.baidu.com/bce-documentation/RTC/rtc_Android_release_fastdemo_20230413.zip

2. 创建项目 可以使用官网下载的Demo进行参考，也可自行创建项目导入SDK。下面是快速实现音视频通话的几个步骤：

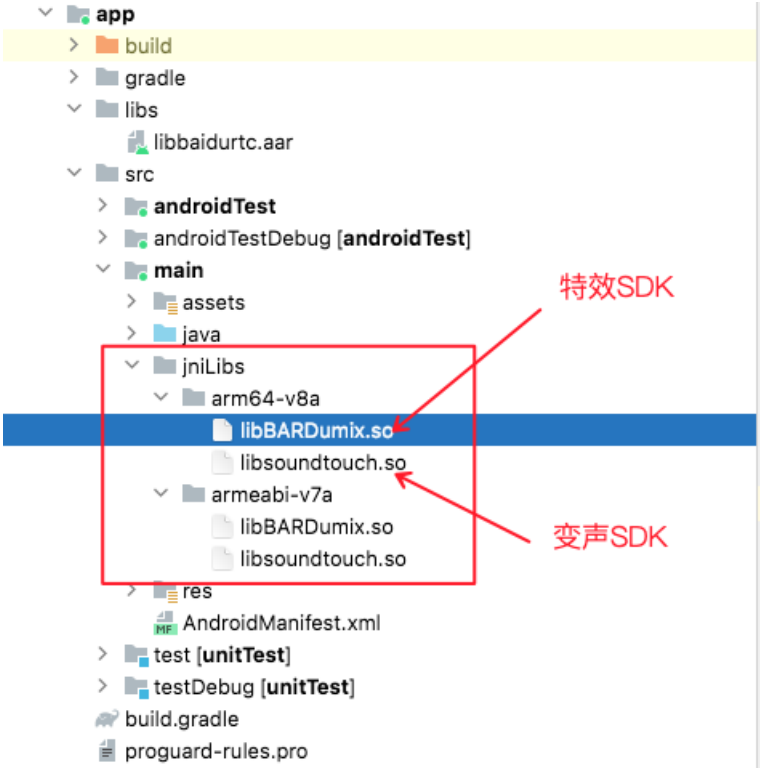
（1）创建Android项目 按照以下步骤准备开发环境：如需创建新项目，在 Android Studio 里，依次选择 Phone and Tablet > Empty Activity，创建 Android 项目。 创建项目后，Android Studio 会自动开始同步 gradle, 稍等片刻至同步成功后再进行下一步操作。

（2）导入SDK

- 在项目的libs目录中增加 libbaidurtc.aar



- 如果需要使用变声能力/ 特效能力，需要在jniLibs目录中，增加 libsoundtouch.so / libBARDumix.so



- 修改你的项目中app module 下的 build.gradle，增加如下相关依赖配置项 apply plugin: 'com.android.application'

```
 dependencies {
   implementation fileTree(dir: 'libs', include: ['*.aar'])
}
```

（3）添加网络及设备权限，在 /app/src/main/AndroidManifest.xml 文件中，在 <application 前面添加如下权限：

```
   <uses-permission android:name="android.permission.READ_PHONE_STATE" android:maxSdkVersion="19" />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission android:name="android.permission.READ_LOGS" />
  <uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
  <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
  <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

3. 使用SDK实现音视频通话 (1) 初始化BaiduRtcRoom 需要调用initWithAppID 设置申请的APPID ，token； 调用 setMediaServerURL 设置服务器地址，可使用Demo中默认的URL地址；也可用自己部署的信令服务器地址； 具体可参考 SDK 初始化

```
 // 初始化,如果公有云可以用默认token，如果私有云需要使用申请APPID时获取的token
mVideoRoom = BaiduRtcRoom.initWithAppID(this, mAppId, tokenStr);
// 设置是否开启控制台log输出，true输出到控制台，false不输出
BaiduRtcRoom.setVerbose(true);
// 状态上报
mVideoRoom.enableStatsToServer(true, "online");
// 设置回调
mVideoRoom.setBaiduRtcRoomDelegate(this);
// 设置url
mVideoRoom.setMediaServerURL(url);

// 是否房间模式，默认是 BDRTC_ROOM_NORMAL
mVideoRoom.setRoomMode(BaiduRtcRoom.BdRtcRoomMode.BDRTC_ROOM_NORMAL);
```

(2) 视频通话的用户界面中，通常有两个视图框，分别用于展示本地预览和远端视频 可直接在layout xml 文件中增加两个RTCVideoView

```
 <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.baidu.rtc.RTCVideoView
        android:id="@+id/local_rtc_video_view"
        android:layout_width="240dp"
        android:layout_height="320dp"
        android:gravity="center_vertical" />

    <com.baidu.rtc.RTCVideoView
        android:id="@+id/remote_rtc_video_view"
        android:layout_width="240dp"
        android:layout_height="320dp"
        android:layout_marginTop="10dp"
        android:gravity="center_vertical" />

</LinearLayout>
```

```
 RTCVideoView rtcLocalVideoView = findViewById(R.id.local_rtc_video_view);
RTCVideoView rtcRemoteVideoView = findViewById(R.id.remote_rtc_video_view);
```

或者也可通过新建RTCVideoView的方式创建

```
 // 创建local
RTCVideoView rtcLocalVideoView = new RTCVideoView(mContext);
// 设置视频显示类型，SCALE_ASPECT_FIT 填充，其他显示类型，请查看官方文档
rtcVideoView.setScalingType(RTCVideoView.ScalingType.SCALE_ASPECT_FIT);
rootView.addView(rtcLocalVideoView);

// 创建remote
RTCVideoView rtcRemoteVideoView = new RTCVideoView(mContext);
// 设置视频显示类型，SCALE_ASPECT_FIT 填充，其他显示类型，请查看官方文档
      rtcVideoView.setScalingType(RTCVideoView.ScalingType.SCALE_ASPECT_FIT);
rootView.addView(rtcRemoteVideoView);
```

(3) 设置本地预览 调用BaiduRtcRoom setLocalDisplay(rtcVideoView) 接口，具体可参考 视频本地渲染 开始预览，在Activity onCreate 或者 onResume()中添加开始预览

```
  mVideoRoom.setLocalDisplay(rtcLocalVideoView);
mVideoRoom.startPreview();
```

(4) 配置BaiduRtcRoom参数 具体可参考 媒体参数配置

```
 RtcParameterSettings cfg = RtcParameterSettings.getDefaultSettings();
cfg.EnableMultistream = true;
cfg.inputAudioChannel = 1;
// 音频输出通道数
cfg.outputAudioChannel = 1;
// 音频采样率
cfg.AudioFrequency = 48000;
// 自动发布
cfg.AutoPublish = true;
// 自动订阅用户
cfg.AutoSubScribe = true;
// 是否有视频
cfg.HasVideo = true;
// 是否有音频
cfg.HasAudio = true;

// 分辨率
cfg.VideoResolution = "640x480-800kbps";
// 或者用VideoWidth
cfg.VideoHeight = 640;
cfg.VideoWidth = 480;
// 视频最大码率
cfg.VideoMaxkbps = 800;
// 帧率
cfg.VideoFps = 15;

// 链接超时时间
cfg.ConnectionTimeoutMs = 5000;
// 链接读超时时间
cfg.ReadTimeoutMs = 5000;

if (Build.MANUFACTURER.contains("Ainemo")
            || Build.MODEL.contains("NV6001")
            || Build.MODEL.contains("NV6101")
            || Build.MODEL.contains("NV2001")
            || Build.MODEL.contains("NV5001")) {
    cfg.AudioFrequency = 16000;
    cfg.inputAudioChannel = 2;
    cfg.outputAudioChannel = 2;
    cfg.audioContentType = AudioAttributes.CONTENT_TYPE_MUSIC;
}

if (Build.MODEL.contains("ONEPLUS")) {
    cfg.AudioSource = MediaRecorder.AudioSource.DEFAULT;
}

// 是否有远端音频，默认true
cfg.HasRemoteAudio = true;
// 是否有远端视频，默认true
cfg.HasRemoteVideo = true;

// 重连，cfg.enableAutoReconnect 如果设置true (默认是false)，则SDK自动重连，如需业务侧实现重连，则置为false
// 若使用自动重连功能 cfg.enableListenNetwork 需为true（默认是true）
cfg.enableAutoReconnect = false;

// enable jitter deley can be stretched by retransmission packets
cfg.enableJitterRetransmission = true;

// 设置参数
mVideoRoom.setParamSettings(cfg, RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);
```

(5) 登录房间 具体可参考接口文档 登录房间接口

```
// roomName：房间名； userId：用户Id； userName：用户名
mVideoRoom.loginRtcRoomWithRoomName(roomName, userId, userName);
```

(6) 设置远端画面预览 通过（2）获取到显示远端画面的RTCVideoView，调用BaiduRtcRoom setRemoteDisplay 接口，即可. setRemoteDisplay 可绑定userId，也可以不绑定userId，具体参考接口文档 远端画面渲染

```
// 绑定远端用户画面
mVideoRoom.setRemoteDisplay(rtcRemoteVideoView, userId);
```

完成以上步骤即可实现音视频通话，也可根据Demo查看多人布局效果，更多能力，请查看官网文档 https://cloud.baidu.com/doc/RTC/s/HkOgh4qck ，或联系客服。

操作指南

控制台功能简介

RTC控制台为您提供**概览**、**应用管理**、**房间管理**、**质量监控**、**统计分析**、**通信记录**、**多用户访问控制**等功能。

功能	说明
概览	为您展示本月累计通话时间、昨日通话时间、昨日活跃用户数、并发通信等统计信息，以及近7天的通话时长折线图
应用管理	提供应用的生命周期管理，包括创建应用、启用/停用应用、编辑应用信息，查看并管理应用下的房间列表等功能
房间管理	您可通过房间管理查看应用下的房间列表，以及房间中通话用户的通话情况
质量监控	您可通过质量监控查看应用下每个房间的实时通话质量以及历史通话质量
统计分析	查看某个应用在一定时间范围内的实用情况，包括通话时长、活跃用户、房间数、并发通信峰值等
通信记录	查看某个应用下全部频道的通信记录，包括开始/结束时间、通话统计和用户记录
多用户访问控制	帮助用户管理云账户下资源的访问权限，适用于对企业内不同角色赋予不同权限

应用管理

创建应用

概述

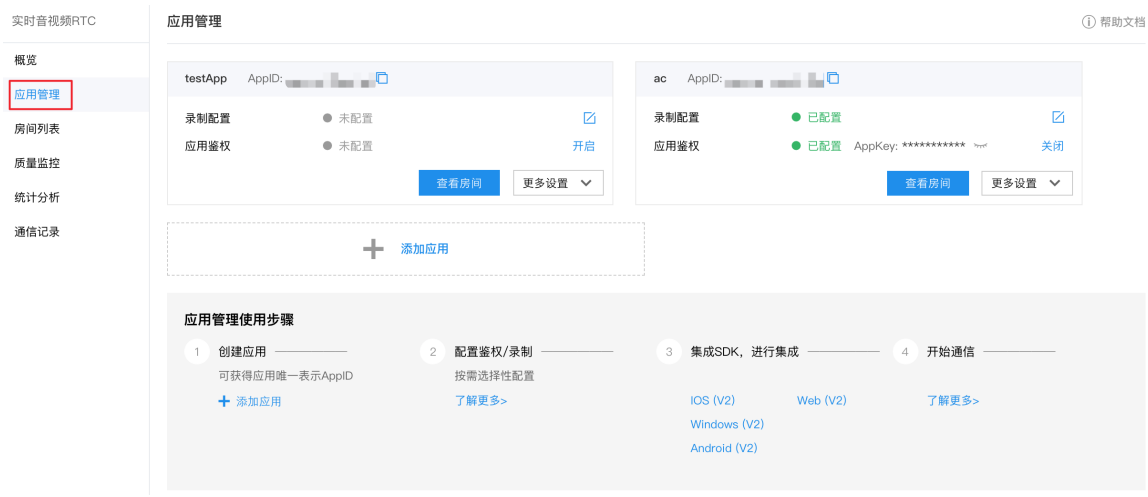
您可通过控制台创建应用并获取应用的AppID。本文将为您介绍如何通过控制台创建一个应用。

前提条件

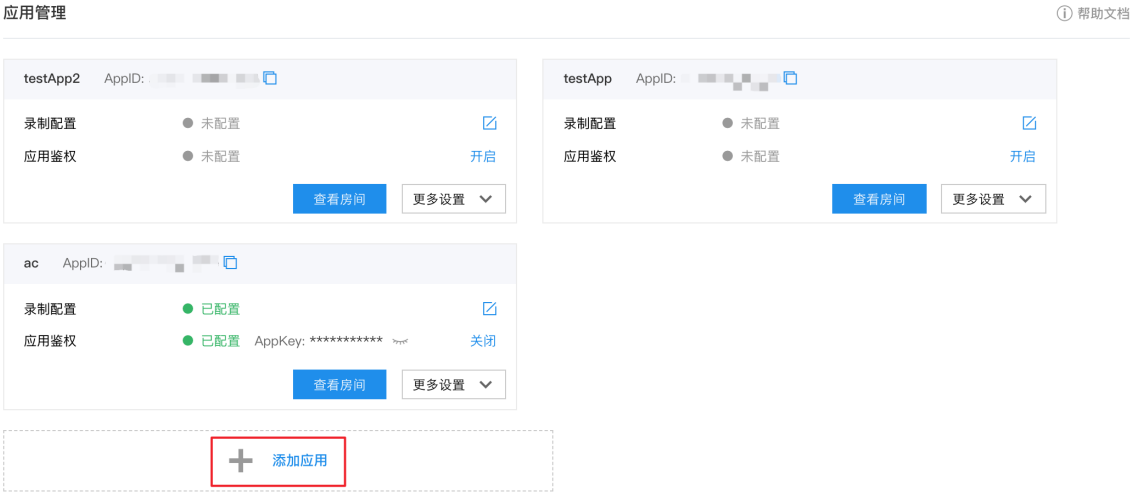
已注册百度智能云账号并完成实名认证。具体操作请参见 注册百度智能云账号 和 实名认证。

创建应用

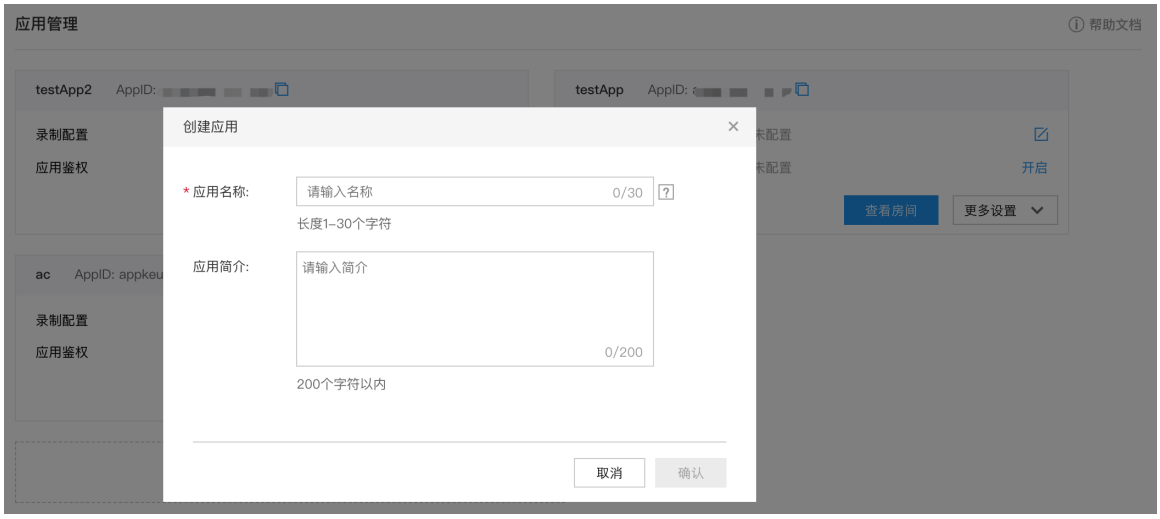
1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**应用管理**。



3. 在应用管理页，点击**添加应用**。



4. 在弹出的**创建应用**框中，完成应用名称、应用简介设置。



项目	是否必填	填写说明
应用名称	必填	应用名称必须唯一，支持汉字、英文字母、数字、英文格式的下划线，必须以英文字母或汉字开头，1~30个字符。
应用简介	非必填	您可以通过添加应用简介标记应用的用途，便于对应用的管理。

5. 点击**确认**，完成应用创建。

录制配置

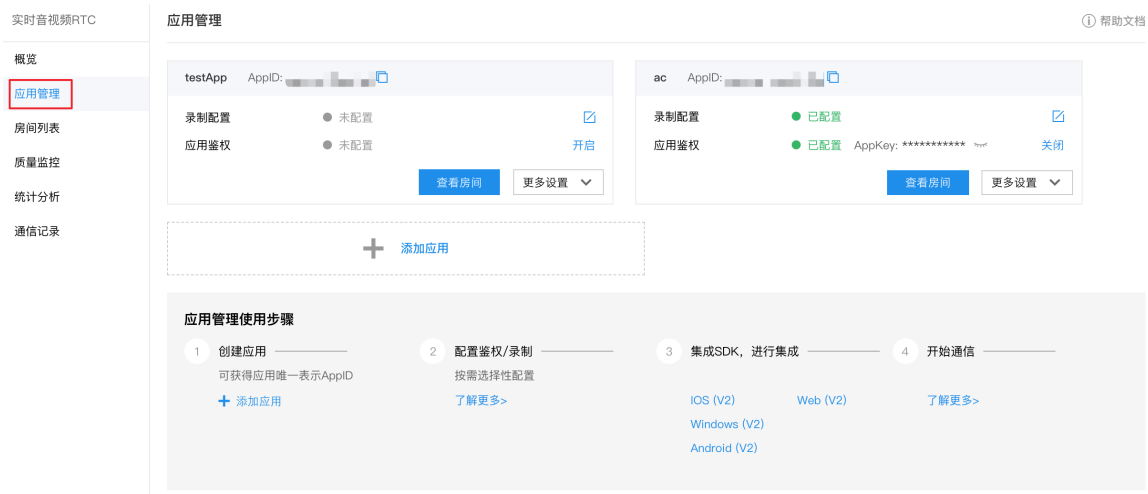
概述

本文将为您介绍如何使用控制台开启或关闭服务端录制。

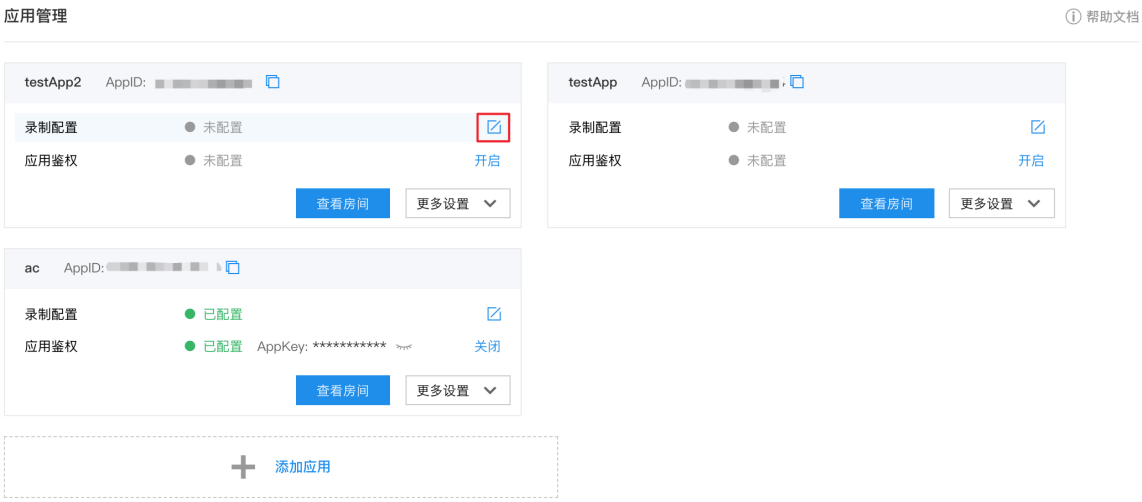
开启录制配置

1. 登录 实时音视频 RTC 控制台。

2. 在左侧导航选择**应用管理**。



3. 选择您需要开启录制配置的应用，点击该应用录制配置右侧的 。



4. 在弹出的**录制配置**框，配置录制信息。

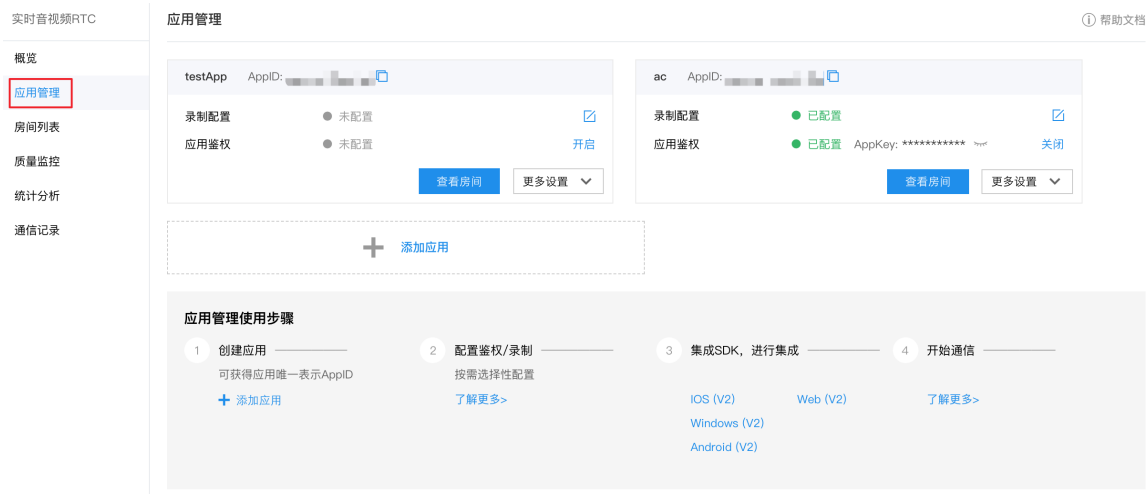
配置项	配置说明
录制配置	选择开启，开启录制配置。
录制模式	提供混流录制和单独录制两种模式。 **混流录制**：将房间内所有通话用户混合为一路进行录制，最大支持6路混合。支持选择纯音频或音视频录制 • **混流录制 - 纯音频**：只录制房间内容音频，不保留视频。 • **混流录制 - 音视频**：完整录制房间内的音频及视频。 **单路录制**：支持将房间内通话的用户分别录制为单独的文件。
录制布局	选择录制模式为**混流录制-音视频**时，可选择录制布局。 • **画中画**：画中画模式，第一路为大画面，其余画面位于底部；大画面为主画面；以主画面尺寸为基准，其余画面按比例自适应叠加在主画面中，其余画面为方形；最大支持6路。 • **平铺**：平铺模式，所有画面大小相等平铺；左上画面为主画面；以主画面尺寸为基准平铺，设置对齐的小画面大小；最大支持6路。 • **主次平铺**：主次平铺模式，第一路为大画面，其余画面在右侧及底部平铺；大画面为主画面；以主画面尺寸为基准平铺，设置对齐的小画面大小；最大支持6路。
编码参数	选择录制布局后，需要选择录制的编码参数。您可选择360P（16:9）、标清360P（4:3）、标清480P（16:9）、标清480P（4:3）、标清720P（16:9）或标清720P（4:3）。
录制格式	可选择 FLV 或 MP4 格式。
单文件录制时长	可输入 1～360 分钟，必须输入整数。
录制文件命名模式	系统默认pattern=%a/%r/%d/recording_%t.%f；%t和%f为必填项，不可删除； 单路录制时录制文件名系统默认pattern=%a/%r/%d/recording_%t_%u.%f，%t_%u和%f为必填项，不可删除；其中， %a：AppId %r: RoomName %y: Year（2019） %m: Month（2019-09） %d: Day（2019-09-02） %u：UserId %t: 录制开始时间，CST时区，精确到秒 %f: 录制文件格式 例如，生成的录制文件名形如AppId/RoomName/2019-09-02/recording_20190902120931.flv
储存模式	默认为标准BOS存储，不可修改。
存储所属区域	支持选择华北 - 北京 、华东 - 苏州 、华南 - 广州。
选择储存Bucket	选择您要存储的Bucket。如果您尚未创建Bucket，您可参考创建Bucket创建一个Bucket。当您选择录制存储到BOS bucket时，系统将授权RTC服务访问该bucket。

5. 点击**确认**，完成录制配置。

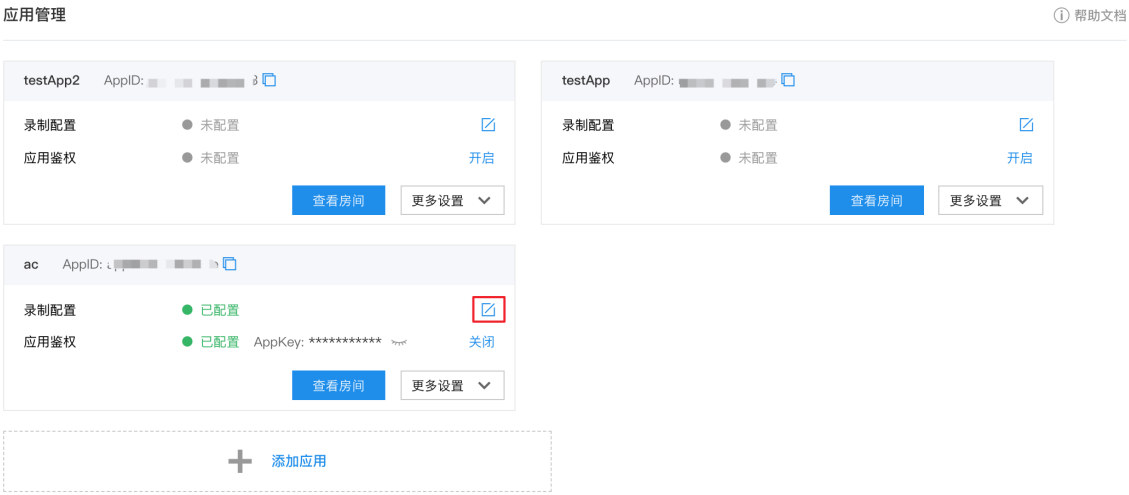
关闭录制配置

1. 登录 实时音视频 RTC 控制台。

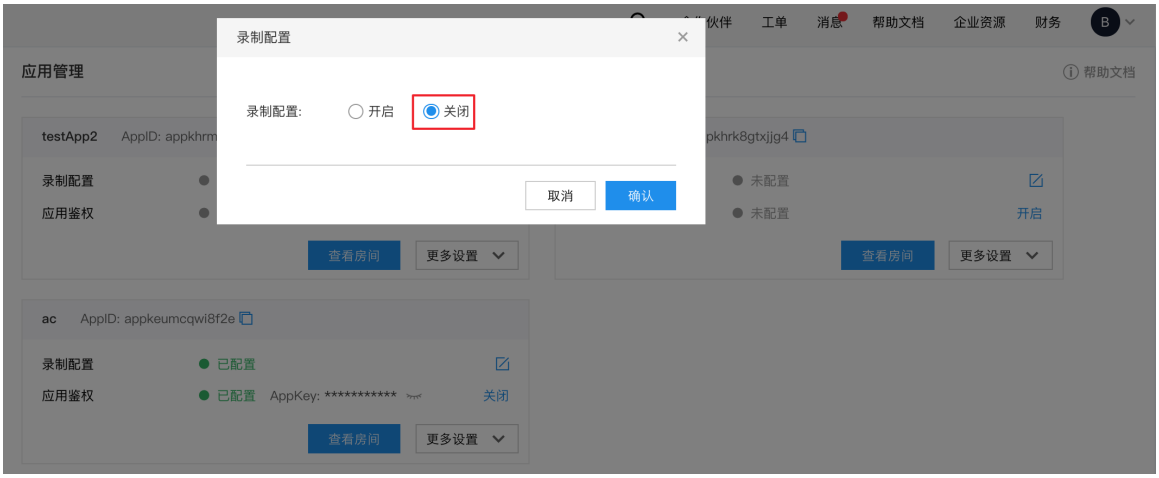
2. 在左侧导航选择**应用管理**。



3. 选择您需要关闭录制配置的应用，点击该应用录制配置右侧的 。



4. 在弹出的**录制配置**弹框中，勾选录制配置为**关闭**。



5. 点击**确认**，即可关闭录制配置。

应用鉴权

概述

开启应用鉴权会使得您的服务更加安全，启用应用鉴权后每次请求服务时须带上动态秘钥，否则服务请求将被拒绝。本文将为您介绍如何使用控制台开启或关闭应用鉴权。

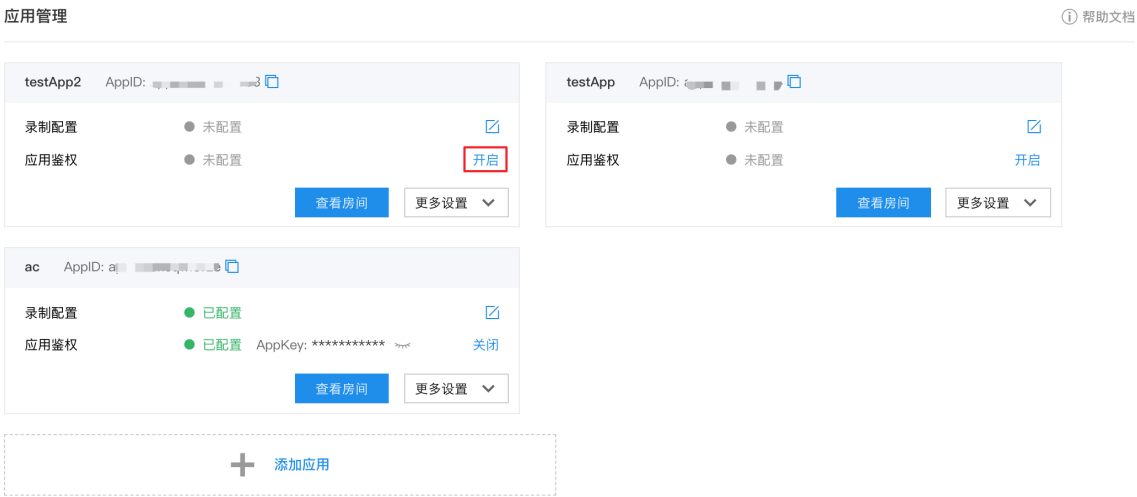
前提条件

您需要在AppServer上搭建Token服务器，详情请参见 在AppServer上部署token生成服务。

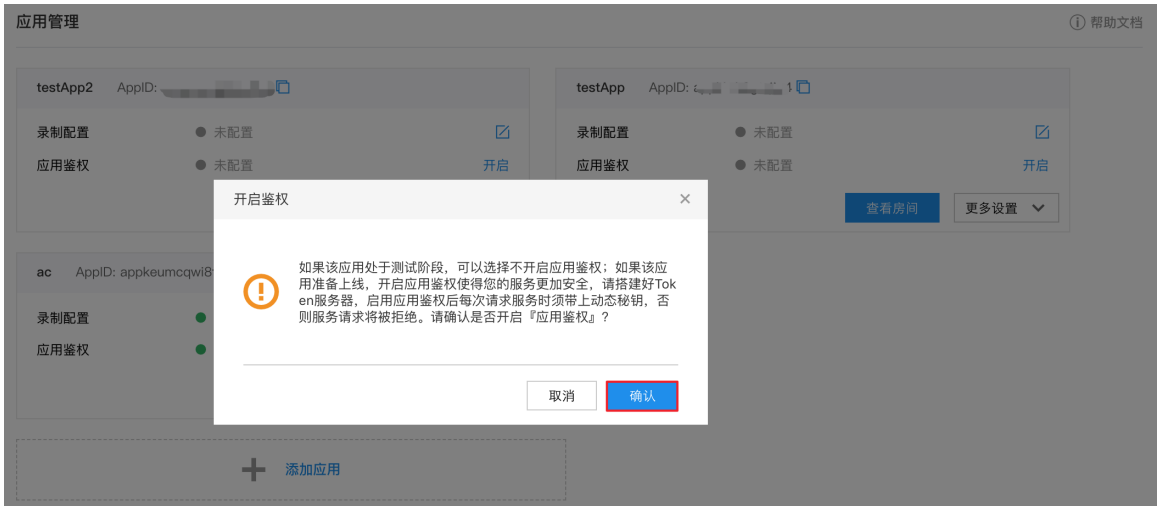
开启应用鉴权

1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**应用管理**。

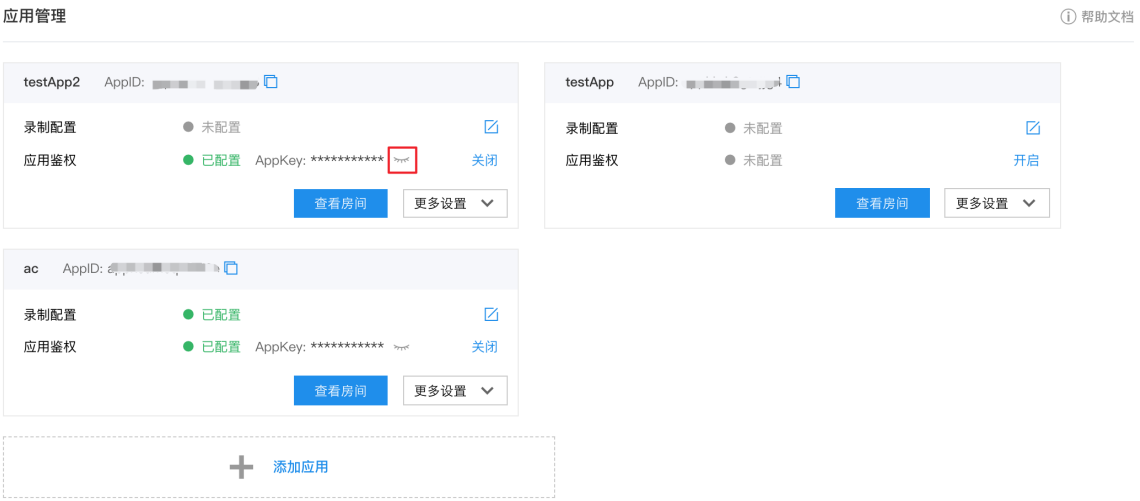
3. 选择需要开启应用鉴权的应用，点击应用鉴权右侧的**开启**。

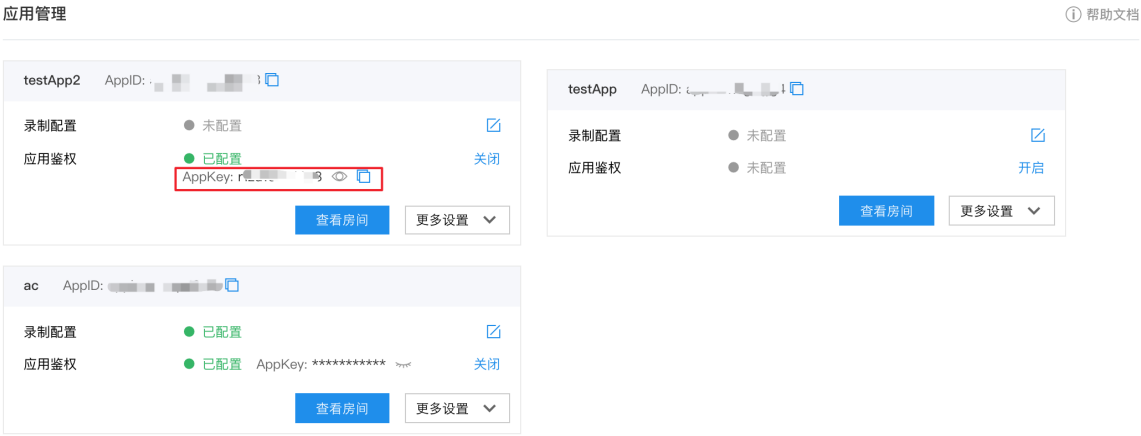


4. 在弹出的**开启鉴权**弹框中，点击**确认**。



5. 开启应用鉴权后，点击AppKey右侧的 ，即可查看AppKey。



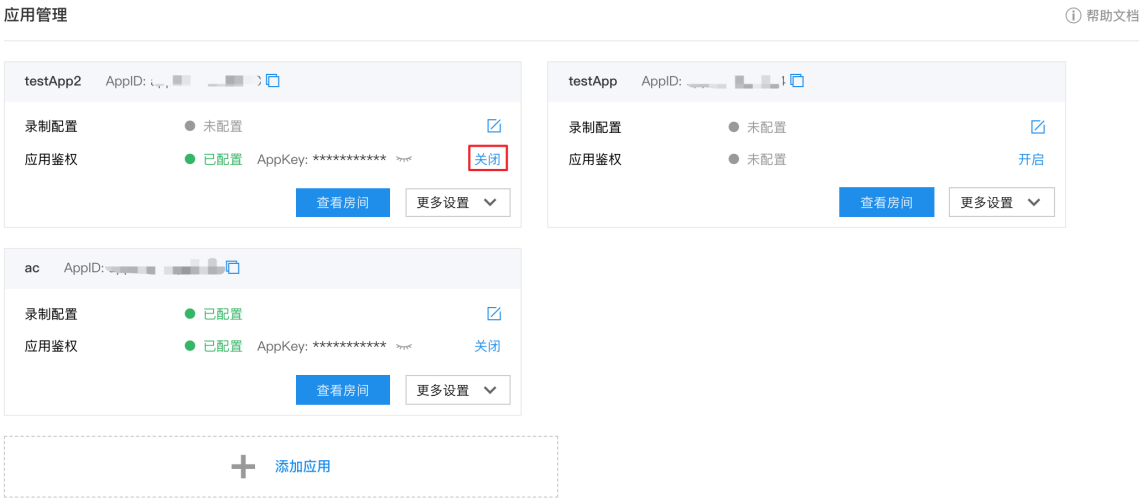


关闭应用鉴权

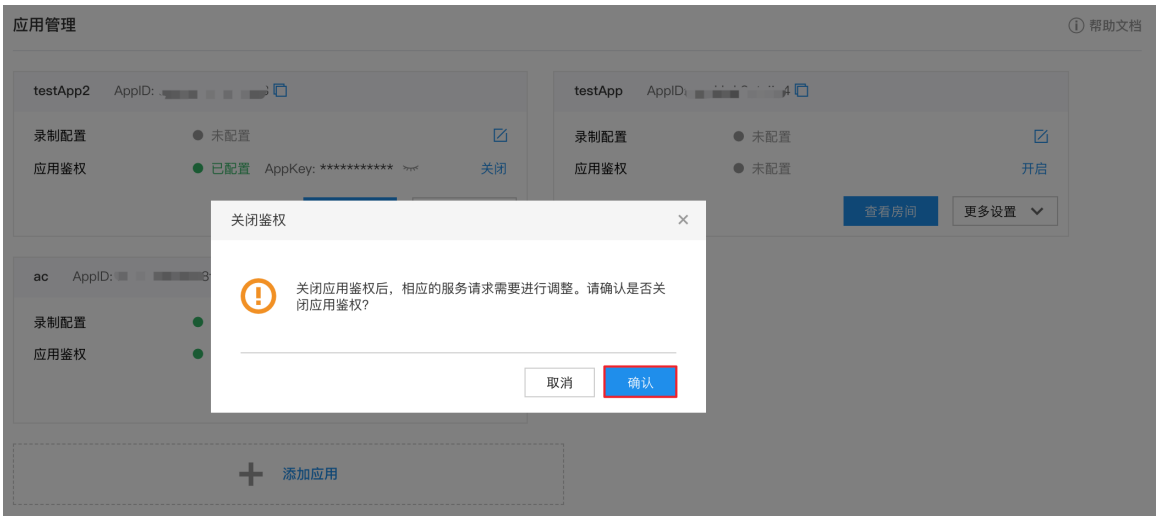
1. 登录 实时音视频 RTC 控制台。

2. 在左侧导航选择**应用管理**。

3. 选择需要开启应用鉴权的应用，点击应用鉴权右侧的**关闭**。



4. 在弹出的**关闭鉴权**弹框中，点击**确认**完成应用鉴权的关闭。



5. 此时该应用的应用鉴权状态为**未配置**。

房间管理

概述

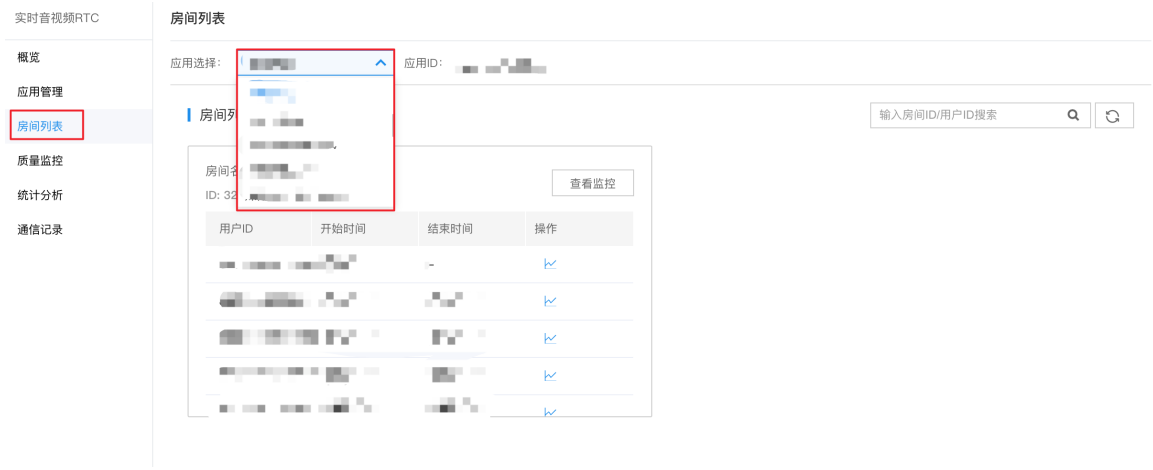
您可通过房间管理查看应用下的房间列表，以及房间中通话用户的通话情况。

注意事项

仅显示通话中房间，追溯历史通话记录请到 通信记录 页面查看。

查看房间列表

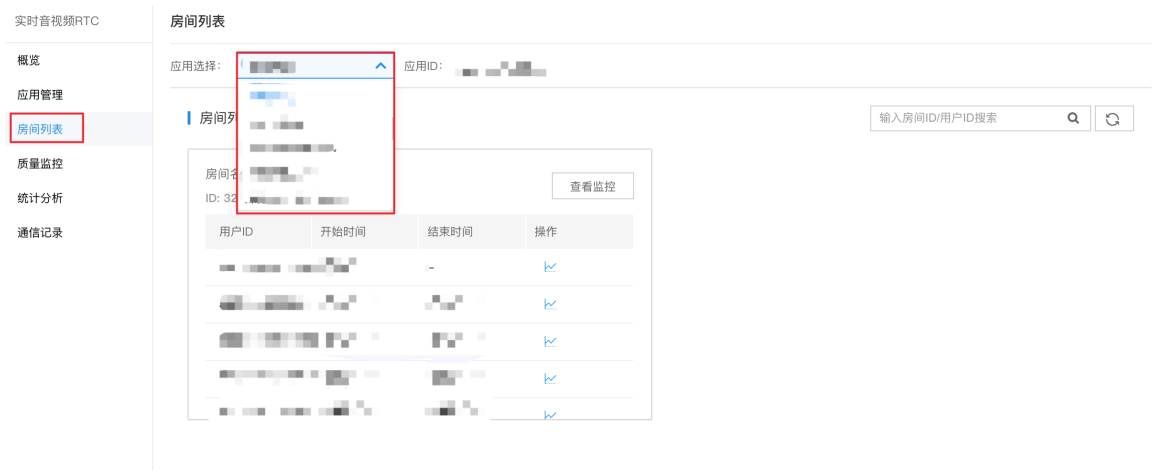
1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**房间列表**。
3. 在**房间列表**页选择要查看的**应用**。



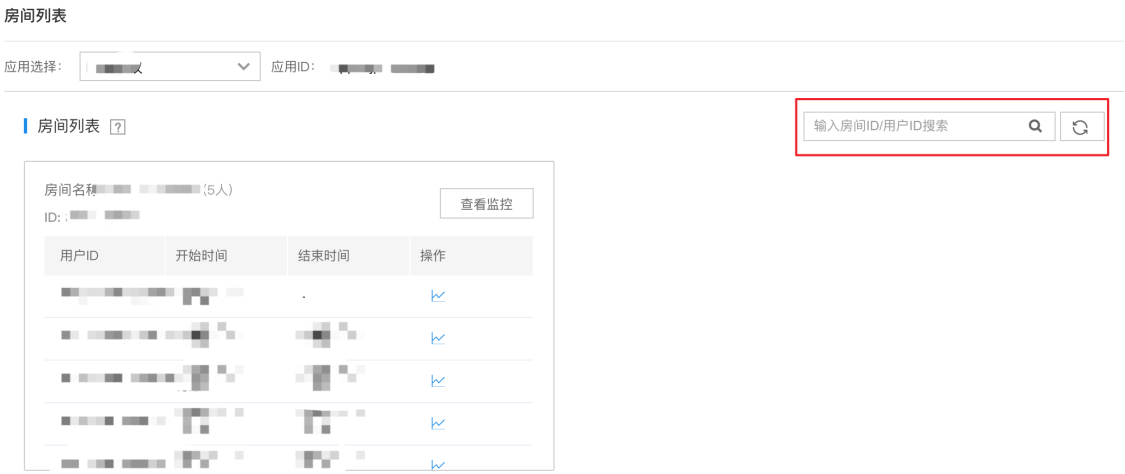
4. 此时即可查看该应用下的房间列表。

筛选房间

1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**房间列表**。
3. 在**房间列表**页选择要查看的**应用**。

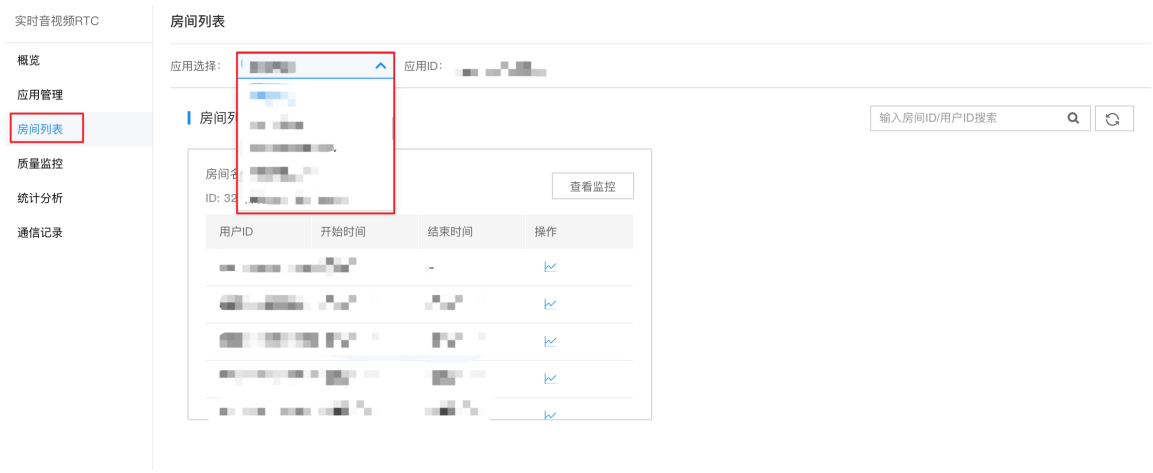


4. 在右侧搜索框中您可根据**房间ID**或**用户ID**搜索房间。

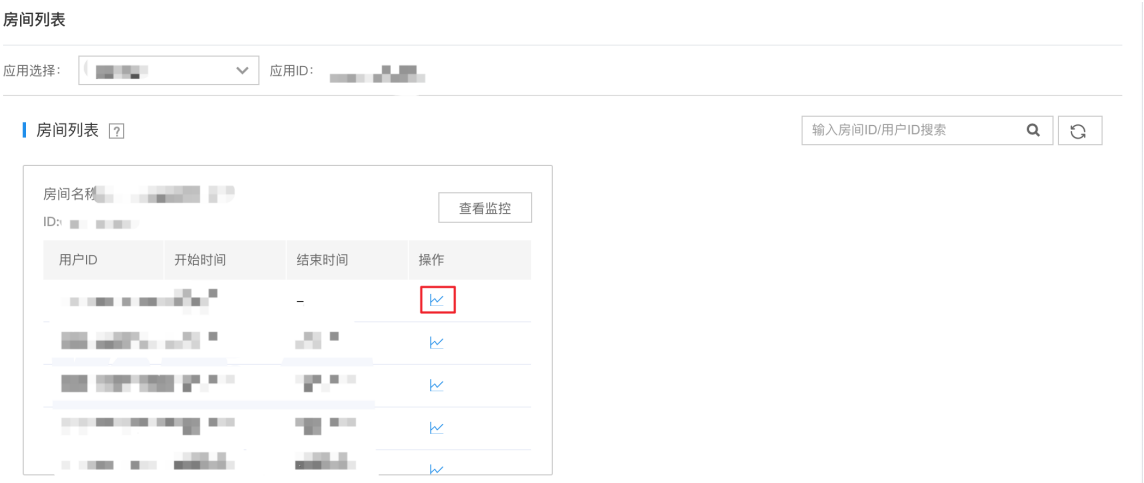


查看用户通话质量

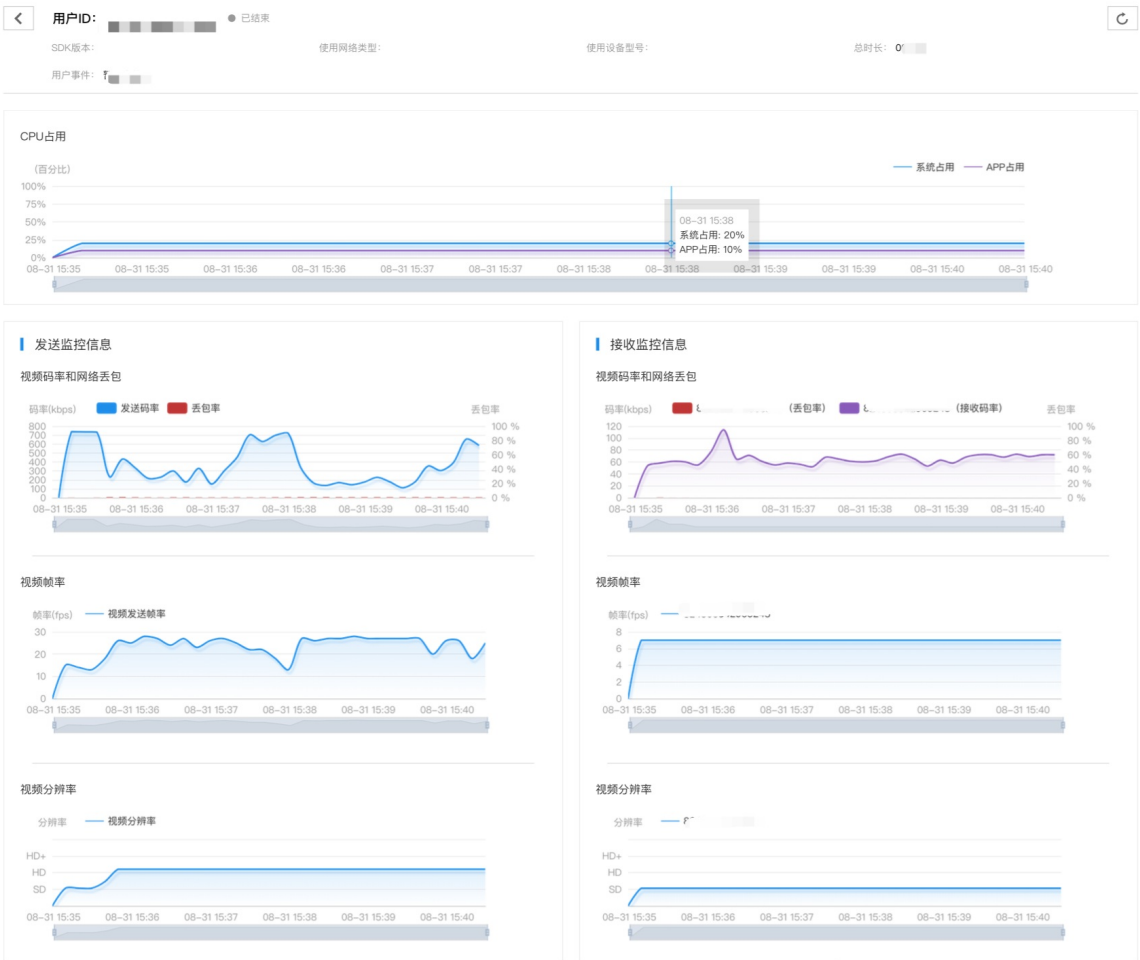
1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**房间列表**。
3. 在**房间列表**页选择要查看的**应用**。



4. 选择需要查看的用户，点击该用户ID右侧的  。



5. 此时即可查看当前用户的通话状况。



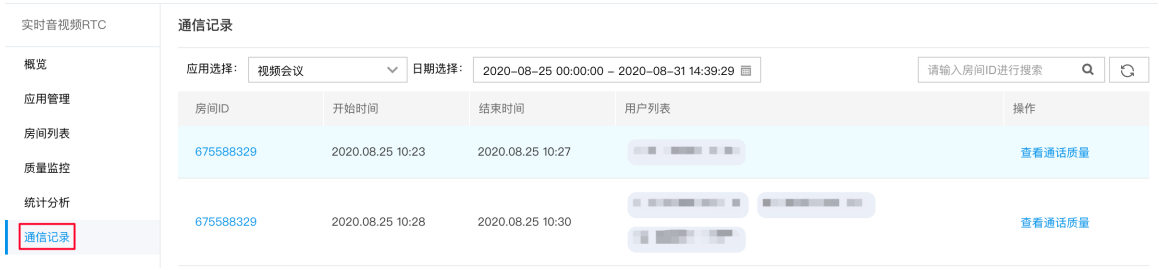
通信记录

概述

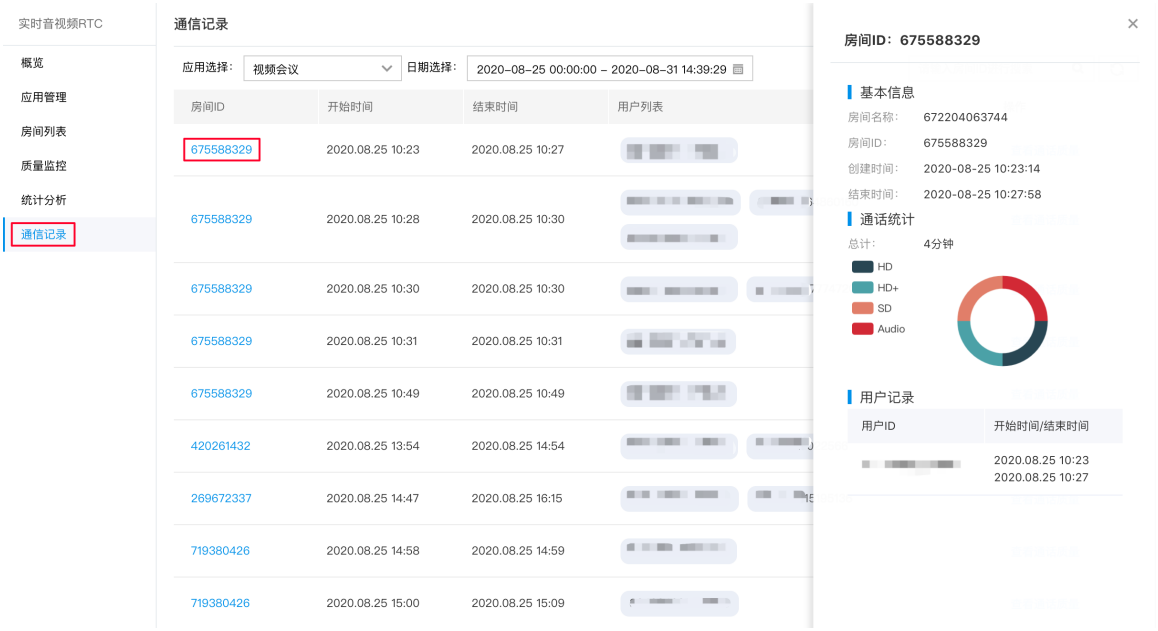
通信记录中可查询每一个通信的详情，根据应用、时间、房间ID、通话人ID进行检索。

查询通信记录

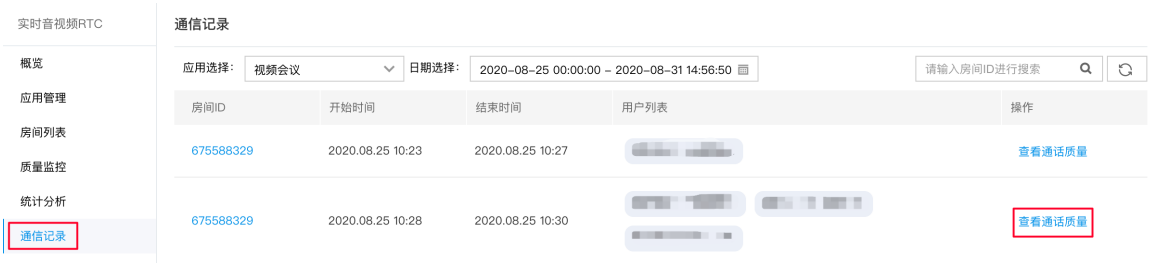
1. 登录 实时音视频 RTC 管理控制台。
2. 点击左侧导航栏 **通信记录** 按钮，进入通信记录页面。
3. 每个房间的创建和结束生成一条通信记录，在通信记录页面，可查看历史通信记录。



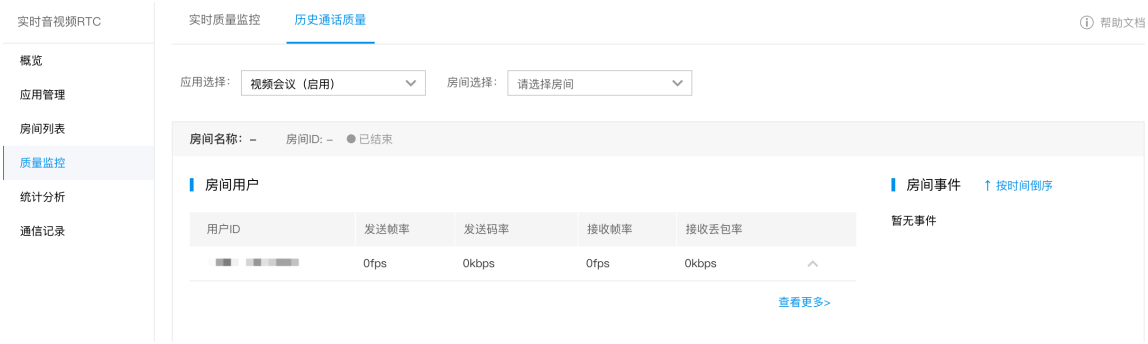
4. 您可以根据 **应用名称、日期** 等条件，对列表展示的信息进行筛选，也可以搜索**房间 ID**查找指定的通信记录。
5. 单击**房间ID**即可查看该次通信的详情。通信详情包括：房间名称、房间ID、创建时间、结束时间、通话统计（Audio、SD、HD、HD+），以及每个用户在该房间的通信记录。



6. 在目标“房间 ID”的操作列，单击**查看通话质量**，进入质量监控页面。



7. 在此页面可以查看该房间的历史通话质量。



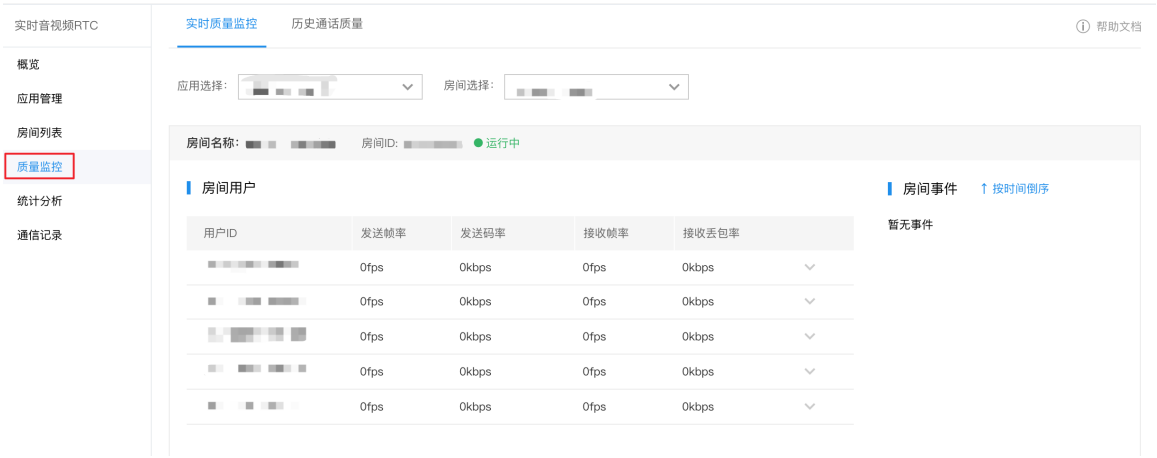
质量监控

概述

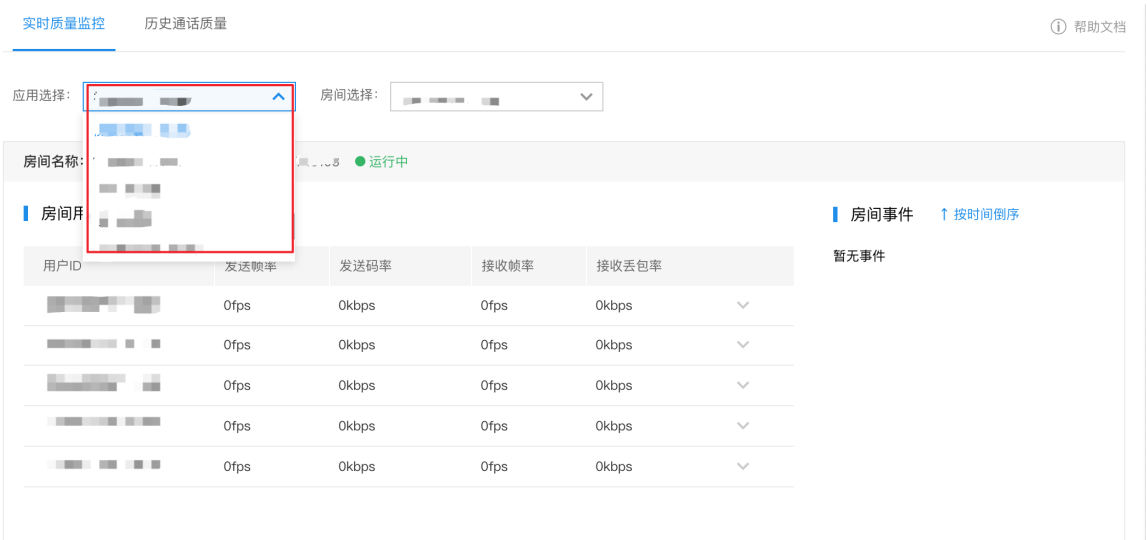
您可通过质量监控查看应用下每个房间的实时通话质量以及历史通话质量。

实时质量监控

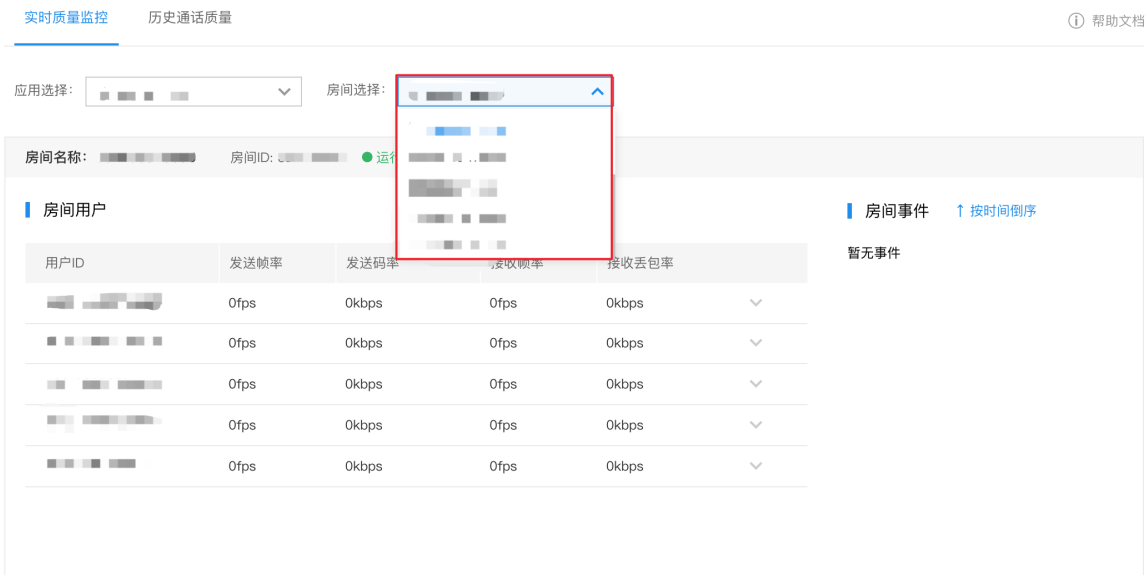
1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**质量监控**。



3. 在**实时质量监控**页签，选择要查看实时质量监控的应用。



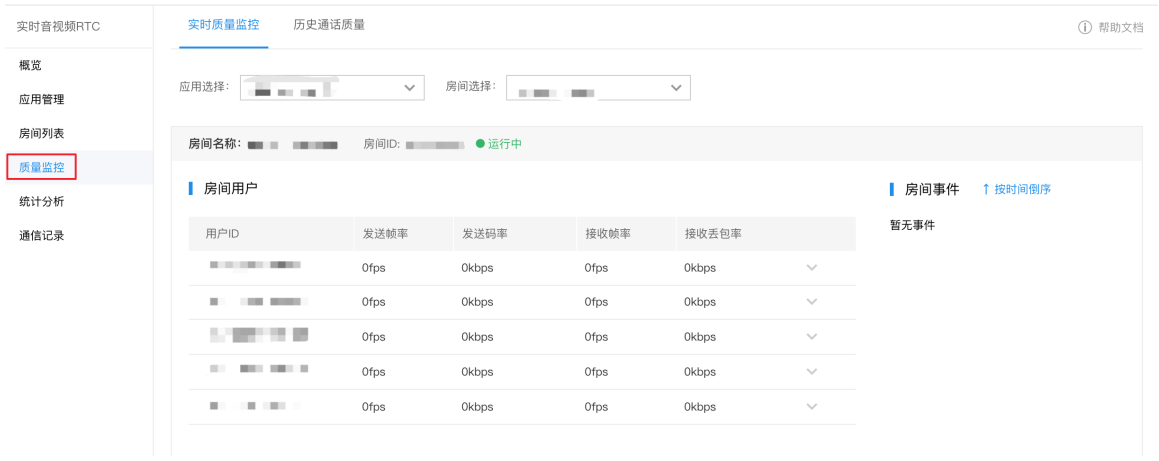
4. 选择要查看实时质量监控的房间。



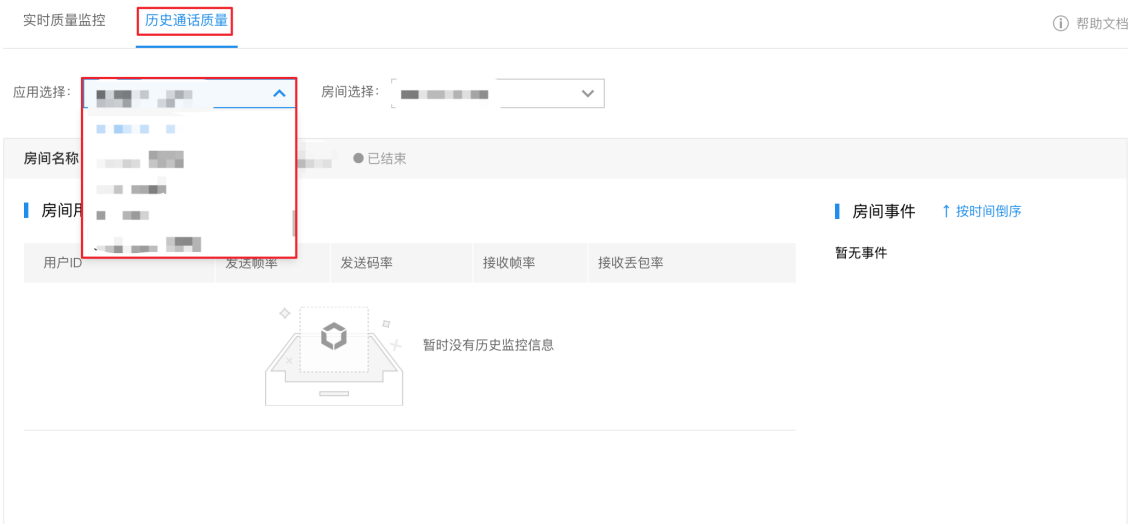
5. 此时即可查看该房间用户的**发送帧率**、**发送码率**、**接收帧率**、**接收丢包率**。

历史通话质量

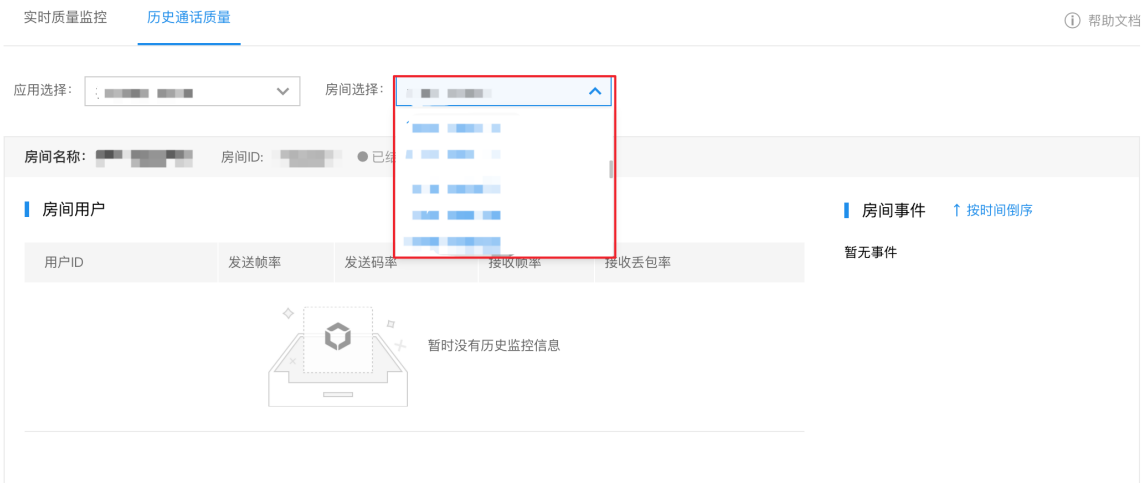
1. 登录 实时音视频 RTC 控制台。
2. 在左侧导航选择**质量监控**。



3. 在**历史通话质量**页签，选择要查看历史通话质量的应用。



4. 选择要查看历史通话质量的房间。



5. 此时即可查看该房间用户的**发送帧率**、**发送码率**、**接收帧率**、**接收丢包率**。

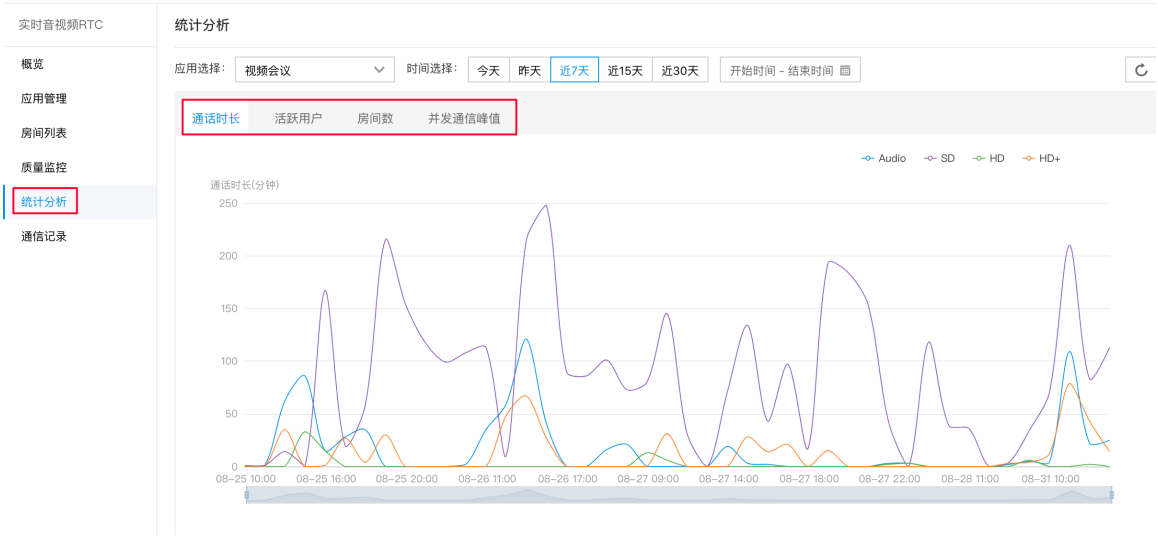
统计分析

概述

统计分析能够查看分应用的通话分钟数（音频、标清、高清）、活跃用户、房间数、并发通信峰值。

查询统计分析

1. 登录 实时音视频 RTC 管理控制台。
2. 点击左侧导航栏 **统计分析** 按钮，进入统计分析页面。
3. 在此页面可查看各个应用的统计数据，可以根据 **应用名称**、**时间** 查询 **通话时长**、**活跃用户**、**房间数**、**并发通信峰值** 的信息。



> **说明：**统计数据以 1h 为粒度展示，以应用为最小单位统计数据。

多用户访问控制

介绍

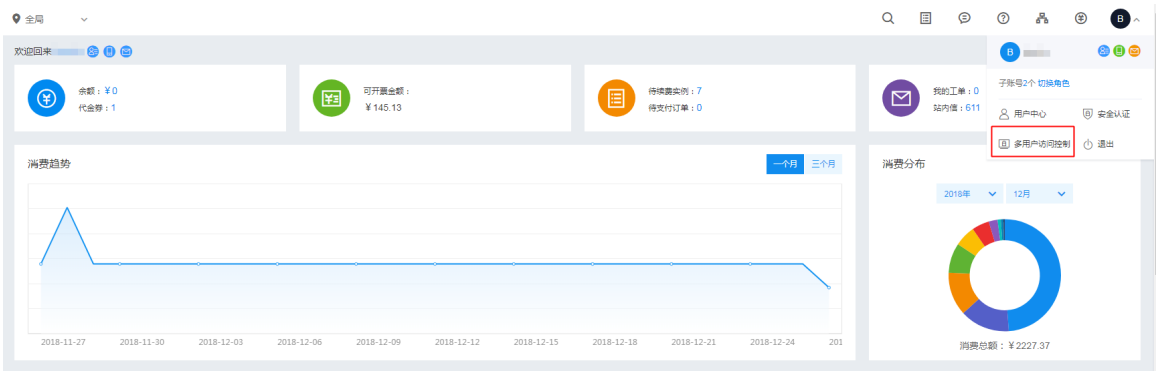
多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限，适用于对企业内不同角色赋予不同权限。当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。

多用户访问控制适用于下列使用场景：

- 中大型企业客户：对公司内多个员工授权管理；
- 偏技术型vendor或SAAS的平台商：对代理客户进行资源和权限管理；
- 中小开发者或小企业：添加项目成员或协作者，进行资源管理。

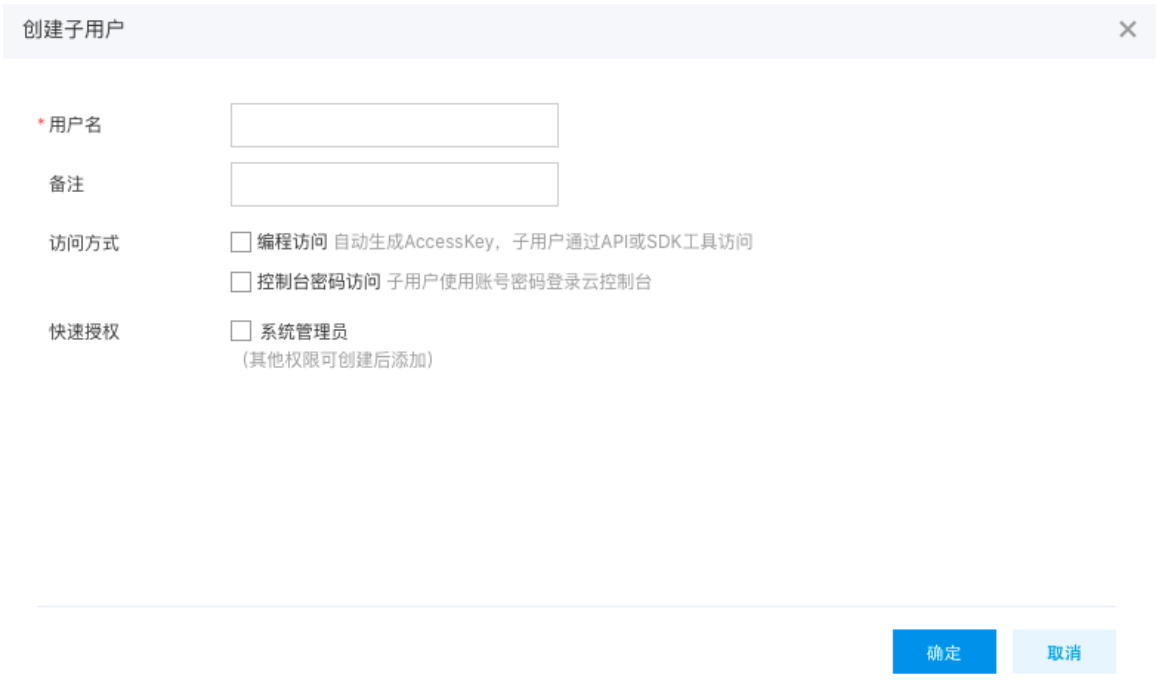
创建用户

1. 主账号用户登录百度智能云控制台，选择“多用户访问控制”进入用户管理页面。



2. 点击左侧导航栏“用户管理”->“子用户”，点击“创建子用户”。

3. 在弹出的“新建用户”对话框中，完成填写“用户名”填写。点击确认，返回“子用户管理列表”区，可以查看到刚刚创建的子用户。



配置策略

RTC支持系统策略实现产品级权限控制。

系统策略：百度智能云系统为管理资源而预定义的权限集，这类策略可直接为子用户授权，用户只能使用而不能修改。

系统策略包含管理权限、只读权限2种策略，权限范围详细如下：

策略名称	权限说明	权限范围
RTCFullControlAccessPolicy	完全控制管理RTC的权限	创建应用、启用/停用应用、编辑应用信息、录制配置、管理房间列表、查看房间列表、查看通话时长、查看活跃用户数、查看房间数、查看并发通信峰值、查看通信记录、查看通话状况
RTCReadAccessPolicy	只读访问RTC的权限	查看应用列表、查看应用信息、查看房间列表、查看通话时长、查看活跃用户数、查看房间数、查看并发通信峰值、查看通信记录、查看通话状况

用户授权

在“用户管理->子用户管理列表页”的对应子用户的“操作”列选择“添加权限”，并为用户选择系统权限进行授权。

子用户登录

主账号完成对子用户的授权后，将链接发送给子用户。子用户通过IAM用户登录链接登录主账号的管理控制台，根据被授权的策略对主账户资源进行操作和查看。

其他详细操作参考：多用户访问控制 IAM

RTC 客户端SDK

Android SDK

概述

概述 本文档主要介绍如何将RTC Andriod SDK集成到您的项目中，并介绍各类功能的使用方法。在使用本文档前，您需要先了解RTC的一些基本知识，并已经开通了RTC服务。若您还不了解RTC，可以参考产品描述。

RTC Andriod SDK 能够帮助您实现登录音视频通信房间并开始通信，并可以对发布/订阅、音视频参数、设备参数进行设置。

接口概览 RTC Android SDK提供以下接口。

初始化接口 | API | 描述 | |--- | --- | |initWithAppID| 初始化SDK | |setParamSettings| 设置媒体参数 | |setAudioProfile| 设置音频编码配置 | |setBaiduRtcEventHandler| 设置回调代理 | |setEngineStateStatistics|开启/关闭引擎统计信息|

房间相关接口 | API | 描述 | | --- | --- | |loginRtcRoomWithRoomName| 登录房间 | |loginRtcRoomWithRoomName:isCompulsive| 强制登录房间 | |logoutRtcRoom| 登出房间 | |startRoomMediaRelay| 启动跨房间通信 | |stopRoomMediaRelay| 停止跨房间通信 | |stopRoomMediaRelayAll| 停止所有跨房间通信 | |disbandRoom| 解散房间 | |kickOffUserWithId| 踢出某成员| |shutUpUserWithId|禁言某成员| |queryUserListOfRoom|获取房间成员ID列表|

发布/订阅流相关接口 | API | 描述 | | --- | --- | | startPublish |开始直播推流| | stopPublish |停止直播推流| | publishStreaming |发布媒体流| | stopPublish |停止直播推流| | subscribeStreaming |订阅某成员媒体流 | | stopSubscribeStreaming | 停止订阅某成员媒体流 |

消息相关接口 | API | 描述 | | - | - | |sendMessageToUser|向某成员发送消息| |setUserAttribute|设置用户属性| |getUserAttribute|获取用户属性| |sendDataMessage|发送数据|

视频相关接口 | API | 描述 | | --- | --- | | startPreview |开启本地预览| | stopPreview | 停止本地预览| |setLocalDisplay|设置本地渲染窗口| |setRemoteDisplay| 设置远端渲染窗口 | |removeRemoteDisplay| 移除远端用户渲染窗口 | |enableExternalVideoCapturer| 设置本地视频采集器 | |addExternalRenderer| 设置某成员外部渲染器 | |removeExternalRenderer| 移除某成员外部渲染器 | |setExternalSurface| 设置某成员外部渲染Surface | |changeSurfaceSize| 更新外部渲染Surface尺寸 | |destroyExternalSurface| 销毁指定用户对应的外部渲染surface | |setRemoteVideoPlayState| 设置是否拉取某成员视频流 | |getRemoteVideoDimension| 获取远端视频尺寸 |

摄像头相关接口 |API|描述| |-|-| |switchCamera|切换摄像头| |muteCamera|关闭/开启本地视频采集| |setCameraID|设置摄像头ID|

音频相关接口 | API | 描述 | | --- | --- | | muteMicphone |设置是否静音采集麦克风数据（不关闭麦克风） | |enableMicCapture|设置是否使用麦克风采集音频（关闭麦克风） | |switchLoundSpeaker| 切换扬声器/听筒 | |presetLoudSpeaker|预置听筒/扬声器| |muteSpeaker|禁止/开启音频输出| |setUserPlaybackVolume|设置远端用户音频播放音量| |isSpeakerOn|查询扬声器是否开启| |getRemoteAudioLevels|获取房间成员语音激励列表| |setRemoteAudioPlayState|设置是否拉取某成员音频流| |setSoundMod|设置音频输出设备| | setAuidoSamplesReadyCallback | 设置本地音频数据回调 | | enableAgc | 开启默认音频自动增益 | | enableAns | 开启默认音频自动噪声抑制 | | enableAec | 开启默认音频回声消除 |

转推配置相关接口 |API|描述| |-|-| |configLiveServerWithUrl|转推配置接口|

通知相关接口 | API | 描述 | | --- | --- | |onLoginSuccess| 用户登录成功 | |onError| 错误信息更新回调 | |onConnectionStateChanged| 链接状态变更 | |onRoomDataMessage| 房间数据接收回调 |

其它接口 | API | 描述 | |-| - | |version|获取SDK版本| |setVerbose| 开启/关闭调试信息输出 | |enableStatsToServer|开启/关闭质量数据上报|

API

初始化接口

SDK初始化

```
public static synchronized BaiduRtcRoom initWithAppID(Context context, String appId, String token)
```

SDK初始化。

初始化SDK并创建RTC房间实例 初始化 SDK 时调用，初始化 SDK 失败会导致 SDK 功能异常。

参数

参数	类型	描述
context	Context	Android上下文环境
appId	String	百度为App签发的 App ID，用于识别应用，全局唯一，详见创建应用。
token	String	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。 详见应用鉴权。

返回

BaiduRtcRoom 实例对象：成功； null：失败；

设置媒体参数

```
public abstract void setParamSettings(RtcParameterSettings paramSettings,RtcParameterSettings.RtcParamSettingType paramType);
```

音视频参数设置。

设置音视频相关的参数。该函数在登录房间前调用，主要用于设置音视频采集，编解码相关的参数。

参数

参数	类型	描述
paramSettings	RtcParameterSettings	该参数封装了音视频的一些参数,如video分辨率，fps, bitrate，音频采样率等
paramType	RtcParamSettingType	参数类型，可指定设置某一项，或者所有参数都设置。

参数类型 RtcParamSettingType说明：

- RTC_PARAM_SETTINGS_ALL：设置全部paramSettings参数集，一般情况下采用该方式设置全部参数；
- RTC_VIDEO_PARAM_SETTINGS_BITRATE：仅设置最高视频码率；

RtcParameterSettings 定义

- 通用属性

类型	属性	默认值	描述
boolean	HasVideo	true	【含义】是否采集、发送视频流 【说明】涉及与服务端视频协商，通信后无法修改。开启后采集摄像头媒体数据。
boolean	HasAudio	true	【含义】是否采集、发送音频流 【说明】涉及与服务端音频协商，通信后无法修改。
boolean	HasScreen	false	【含义】是否采集、发送屏幕共享 【说明】涉及与服务端音频协商，通信后无法修改。
boolean	HasRemoteVideo	true	【含义】配置是否拉取远端视频流
boolean	HasRemoteAudio	true	【含义】配置是否拉取远端音频流
boolean	HasData	false	【含义】是否开启数据通道 【说明】涉及与服务端数据通道协商，通信后无法修改。开启可向其它用户发送数据。
String	VideoCodec	"h264"	【含义】视频编码类型 【说明】推荐使用默认值。
String	AudioCodec	"opus"	【含义】音频编码类型 【说明】推荐使用默认值。
int	AudioFrequency	48000	【含义】音频采样率 【说明】推荐使用默认值。
int	AudioChannel	1	【含义】音频通道 【说明】推荐使用默认值。
int	VideoWidth	640	【含义】发送视频宽 【说明】推荐使用32位对齐的视频采集宽度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。
int	VideoHeight	480	【含义】发送视频高 【说明】推荐使用32位对齐的视频采集高度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。

int	PreviewWidth	0	【含义】预览视频宽 【说明】推荐使用32位对齐的视频采集宽度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常，若未设置，使用VideoWidth作为预览视频宽。
int	PreviewHeight	0	【含义】预览视频高 【说明】推荐使用32位对齐的视频采集高度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常，若未设置，使用VideoHeight作为预览视频高。
int	VideoFps	20	【含义】视频初始帧率 【说明】通信时，实际帧可能会根据当前带宽上下波动。
boolean	MicPhoneMuted	false	【含义】是否发送本端音频流 【说明】与HasAudio不同， 该值仅控制是否发送音频数据，通信后可切换。
boolean	CameraMuted	false	【含义】是否发送本端视频流 【说明】与HasVideo不同， 该值仅控制是否发送视频数据，通信后可切换
int	VideoMaxkbps	1000	【含义】视频最高码率 【说明】通信时，实际发送的视频码率不超过最高码率。
int	VideoMinkbps	0	【含义】视频最低码率 【说明】通信时，实际发送的视频码率不低于最低码率。
int	AudioMaxkbps	-1	【含义】音频最大码率 【说明】默认值 -1 表示使用自适应码率，推荐使用默认值。
int	AudioSource	VOICE_COMMUNICATION	【含义】音频输入源类型 【说明】输入源类型定义可参考系统接口：android.media.AudioSource，通信场景推荐使用默认值。
boolean	EnableMultistream	true	【含义】开启多流模式 【说明】在多人通信场景，多路媒体流复用同一个连接，系统开销更小， 推荐使用。
boolean	AutoPublish	true	【含义】自动发布媒体流 【说明】 房间登录成功后自动发布本端媒体流，设置自动发布后不应再调用startPublish接口手动发布媒体流。
boolean	AutoSubScribe	true	【含义】自动订阅媒体流 【说明】房间登录成功后自动订阅房间内其它用户媒体流，设置自动订阅后不应再调用subscribeStreaming接口手动订阅媒体流。
boolean	enableAutoReconnect	true	【含义】自动重连接 【说明】房间登录成功后，由于网络原因导致链接断开自动重连接。
boolean	disableBluetooth	false	【含义】禁止监听蓝牙设备监听 【说明】房间登录后监听蓝牙设备链接和断开通知。
int	keyAgreementProto	0	【含义】密钥交换方式 【说明】 参数包含： 0：KeyAgreementProtocol.BRTC_DTLS 1：KeyAgreementProtocol.BRTC_SDES 2：KeyAgreementProtocol.BRTC_NONE 推荐使用默认值。
boolean	enablePruneSignal	false	【含义】极简信令 data-channel 【说明】开启极简信令模式。
			【含义】极简信令订阅模式 【说明】

int	subscribeMode	2	参数包含： 1：WebSocketData.SubscribeMode.AUTO_MODE 音频视频自动订阅 2：WebSocketData.SubscribeMode.MANUAL_MODE 手动订阅 3：WebSocketData.SubscribeMode.MEETING_MODE 音频自动订阅 视频手动订阅。
int	subscribeMaxCount	3	【含义】极简信令订阅最大人数。
RtcSignalChannelMode	signalChannelMode	RTC_SIGNAL_CHANNEL_MODE_TCP	【含义】信令传输模式。 【说明】 参数包含: 0: RTC_SIGNAL_CHANNEL_MODE_TCP 传统TCP模式传输 1: RTC_SIGNAL_CHANNEL_MODE_QUIC quic协议模式

- 扩展属性

扩展属性推荐使用默认值，您也可以根据具体应用场景选择性使用。

类型	属性	默认值	描述
int	AudioContentType	AudioAttributes.CONTENT_TYPE_SPEECH	【含义】设置音频输出类型，完整Audio Content Type定义参考系统接口 android.media.AudioAttributes
RtcVideoRenderMode	VideoRenderMode	RTC_VIDEO_RENDER_MODE_INTERNAL	【含义】设置渲染模式：内部渲染/外部渲染
boolean	EnableFixedResolution	false	【含义】是否使用固定分辨率
boolean	EnableRequiredResolutionAligment32	false	【含义】
boolean	DisableBuiltInAEC	false	【含义】禁用内置AEC
boolean	EnableHisiH264HW	true	【含义】是否打开Hisi平台H.264硬件编解码
boolean	EnableMTKH264Decode	true	【含义】是否打开MTK平台H.264硬件解码
boolean	EnableAacCodec	false	【含义】是否开启AAC 解码 (仅低延时播放场景)
boolean	enableJitterRetransmission	false	【含义】是否开启平滑渲染，可优化下行丢包
int	EncodeBitrateMode	RTC_VIDEO_CONTROLRATECONSTANT	【含义】设置编码模式
RtcAudioBitrateMode	audioBitrateMode	RTC_AUDIO_BITRATE_CBR	【含义】设置opus音频编码模式：CBR/VBR;
RtcVideoDegradationPreference	degradationPreference	RtcVideoDegradationPreference.MAINTAIN_FRAMERATE	【含义】保持fps 当网络变差时候降编码低分辨率来达到流畅 【说明】 参数包含：1、MAINTAIN_FRAMERATE 保持framerate 2、MAINTAIN_RESOLUTION 保持分辨率 3、BALANCED 平衡分辨率和framerate
int	ConnectionTimeoutMs	5000	【含义】信令服务器连接超时时长
int	ReadTimeoutMs	5000	【含义】信令读取超时时长
boolean	EnableAudioLevel	false	【含义】开启服务端按音频增益混流
int	AudioLevelTopCount	3	【含义】与EnableAudioLevel 配合使用,控制服务端转发的音频混流路数（根据音频增益大小）

- 屏幕分享属性

屏幕分享相关属性

类型	属性	默认值	描述
boolean	HasScreen	false	【含义】是否采集、发送屏幕媒体流 【说明】涉及与服务端视频协商，通信后无法修改，开启后采集屏幕媒体数据。
Intent	screenIntentData	null	【含义】设置通过系统权限获取到的屏幕Intent数据
Map<String, RtcVideoEncodeParams>	videoEncodeParams	null	【含义】Video编码参数集合,当前主要是屏幕分享使用，用于配置屏幕分享的分辨率、帧率、码率等。 Key一般设置为RtcParameterSetting.RtcMediaTarget，Value详细查看RtcVideoEncodeParams定义的属性
BRTCScreenShareParams	screenShareParams	null	【含义】设置屏幕分享参数详细配置，可替代在videoEncodeParams配置屏幕分享视频参数

其中RtcParameterSetting.RtcMediaTarget定义如下：

类型	属性	默认值	描述
String	TARGET_VIDEO_DEFAULT	video	【含义】默认视频。
String	TARGET_VIDEO_SCREEN	video_screen	【含义】分享屏幕。
String	TARGET_AUDIO_DEFAULT	audio	【含义】默认音频。

其中 RtcVideoEncodeParams属性定义如下：

类型	属性	默认值	描述
int	videoWidth	640	【含义】采集视频宽 【说明】推荐使用32位对齐的视频采集宽度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。
int	videoHeight	480	【含义】采集视频高 【说明】推荐使用32位对齐的视频采集高度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。
int	videoFps	20	【含义】视频初始帧率 【说明】通信时，实际帧可能会根据当前带宽上下波动。
int	videoMaxkbps	1000	【含义】视频最高码率 【说明】通信时，实际发送的视频码率不超过最高码率。

其中 BRTCScreenShareParams属性定义如下：

类型	属性	默认值	描述
boolean	mEnableVideoCapture	true	【含义】是否开启系统屏幕采集。
boolean	mEnableAudioCapture	false	【含义】是否开启系统音频采集。
BRTCScreenShareVideoParams	mVideoCaptureParams	-	【含义】屏幕采集参数，继承自RtcVideoEncodeParams兼容旧版配置，参数定义与之相同，默认值 videoWidth=720，videoHeight=1280，videoFps=10，videoMaxkbps=1500。
BRTCScreenShareAudioParams	mAudioCaptureParams	-	【含义】系统音频参数，包含channel、sampleRate配置。

其中 AudioCodec 属性定义如下：

类型	属性	默认值	描述
String	RtcParameterSettings.VideoCodecId.OPUS	"opus"	【含义】opus编码
String	RtcParameterSettings.VideoCodecId.PCMA	"pcma"	【含义】pcma编码
String	RtcParameterSettings.VideoCodecId.PCMU	"pcmu"	【含义】pcmu编码 【备注】PCMU格式对比opus cpu消耗更低
String	RtcParameterSettings.VideoCodecId.G722	"g722"	【含义】g722编码
String	RtcParameterSettings.VideoCodecId.AMRWB	"amr-wb"	【含义】arm-wb编码 【备注】需要集成全功能SDK

其中 VideoCodec 属性定义如下：

类型	属性	默认值	描述
String	RtcParameterSettings.VideoCodecId.H263	"h263"	【含义】H263编码 【备注】需要集成全功能SDK
String	RtcParameterSettings.VideoCodecId.H264	"h264"	【含义】H264编码
String	RtcParameterSettings.VideoCodecId.H265	"h265"	【含义】H265编码
String	RtcParameterSettings.VideoCodecId.JPEG	"jpeg"	【含义】jpeg编码 【备注】需要集成全功能SDK

设置音频编码配置

```
public abstract int setAudioProfile(BRTCAudioProfileType profile, BRTCAudioScenario scenario);
```

音频编码及应用场景配置。

音频编码及应用场景配置。主要为简化设置媒体参数接口对音频参数的设置，使用该方法时将覆盖设置媒体参数接口对音频相关的参数设置，如果不使用该接口， 则设置媒体参数对音频参数的设置有效。另外，同设置媒体参数一样， 该函数在登录房间前调用，否则无效。

参数

参数	类型	描述
profile	BRTCAudioProfileType	该参数设置音频采样率、码率、编码模式及声道数。
scenario	BRTCAudioScenario	音频应用场景。

BRTCAudioProfileType 定义 设置音频采样率、码率及声道数。

类型	值	描述
BRTC_AUDIO_PROFILE_DEFAULT	0	默认配置：同BRTC_AUDIO_PROFILE_HIGH_QUALITY
BRTC_AUDIO_PROFILE_LOW_QUALITY	1	低音质配置：采样率8KHz, 单声道, 最大编码码率12Kbps
BRTC_AUDIO_PROFILE_STANDARD	2	标准配置：采样率16KHz, 单声道, 最大编码码率32Kbps
BRTC_AUDIO_PROFILE_HIGH_QUALITY	3	高音质配置：采样率48KHz, 单声道, 最大编码码率48Kbps.
BRTC_AUDIO_PROFILE_STEREO_HIGH_QUALITY	4	立体声高音质配置：采样率48KHz, 双声道, 最大编码码率80Kbps
BRTC_AUDIO_PROFILE_SUPER_QUALITY	5	超高音质配置：采样率48KHz, 单声道, 最大编码码率96Kbps
BRTC_AUDIO_PROFILE_STEREO_SUPER_QUALITY	6	立体声超高音质配置：采样率48KHz, 双声道, 最大编码码率128Kbps

BRTCAudioScenario 定义 音频应用场景。

类型	值	描述
BRTC_AUDIO_SCENARIO_DEFAULT	0	默认场景：同BRTC_AUDIO_SCENARIO_SPEECH
BRTC_AUDIO_SCENARIO_SPEECH	1	语音场景：语音清晰及传输稳定流畅，适用于双向语音通信场景
BRTC_AUDIO_SCENARIO_MUSIC	2	音乐场景：高品质音乐音质，适用于高音质的场景
BRTC_AUDIO_SCENARIO_METAVERSE	3	元宇宙场景：支持高音质及立体声

设置回调代理

```
public abstract boolean setBaiduRtcEventHandler(IRtcEventHandler handler);
```

代理设置。

设置RTC Room 代理对象。使用 RTC Room 功能，初始化相关组件时需要设置代理对象，代理对象用户事件回调。未设置代理对象，或对象设置错误，可能导致无法收到相关回调。

参数

参数	类型	描述
handler	IRtcEventHandler	遵循 IRtcEventHandler 协议的代理对象

返回

true 成功，false 失败

开启/关闭引擎统计信息

```
public abstract void setEngineStateStatistics(boolean bOnStatistics);
```

引擎统计信息开关。

当打开开关时 onEngineStatisticsInfo 函数会每隔2秒返回引擎的统计信息，并且可通过queryEngineStatisticsInfo主动函数查询到引擎统计信息

参数

参数	类型	描述
bOnStatistics	bool	是否打开RTC引擎统计信息， bOnStatistics = true,打开开关，bOnStatistics = false, 关闭引擎统计信息

房间相关接口

登录房间

```
public abstract boolean loginRtcRoomWithRoomName(String roomName, long userId, String displayName);
```

登录房间。

登录成功后同一个房间的的成员能够互相看到和听到。

参数

参数	类型	描述
roomName	String	房间名，长度不可超过 255 byte
userId	long	用户ID，每个房间的用户ID必须唯一
displayName	String	用户显示名

返回

true 成功，false 失败

注意

如果失败，会通过错误信息更新回调返回错误信息。

强制登录房间

```
public static class LoginOptions {
  /** 强制登录标志 */
  public boolean isCompulsive = true;

  /** 更新token */
  public String tokenStr;
  /** 角色，仅仅在V4 场景使用 */
  public @Constants.RoleType int roleType = Constants.RoleType.RoleAnchor;
}


public abstract boolean loginRtcRoomWithRoomName(String roomName, long userId, String displayName,
                                                 Constants.LoginOptions options);
```

强制登录房间。

options.isCompulsive = true;

登录房间成功，在同一个房间的人能进行相互音视频聊天。开启强制登录，若房间内之前存在同一userID用户，将被踢出，建议在断网重连或者初次登录失败时调用。

参数

参数	类型	描述
roomName	String	房间名，长度不可超过 255 byte， 不可包括特殊字符及中文
userId	int	用户ID,每个房间的用户 ID 必须唯一
displayName	String	用户显示名
options	Constants.LoginOptions	options.isCompulsive = true; 强制登录； options.isCompulsive = true; 正常登录 默认正常登录

返回

true 成功，false 失败

注意

如果失败，会通过错误信息更新回调返回错误信息。

登出房间

```
public abstract boolean logoutRtcRoom();
```

退出房间。

执行logoutRtcRoom后，会停止音视频采集，断开与房间服务器的连接，取消音视频的传输，销毁音视频传输通道以及释放其他资源。

返回

true 成功，false 失败

启动跨房间通信

```
abstract void startRoomMediaRelay(String destRoomName, long userId, String token)
```

启动跨房间通信。

注意 默认情况下，SDK允许同一房间内的用户间进行通信，若要与其它房间的用户进行通信，则需要需要该接口进行跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	long	加入目标房间时使用的userId，必须在加入房间中是唯一的
token	String	App server 派发的token字符串

停止跨房间通信

```
abstract void stopRoomMediaRelay(String destRoomName, long userId, String token)
```

停止跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	long	加入目标房间时使用的userId，必须在加入房间中是唯一的

停止所有跨房间通信

```
public abstract void stopRoomMediaRelayAll()
```

停止所有跨房间通信。

解散房间

```
public abstract boolean disbandRoom();
```

解散房间。 房间管理员有权利解散整个房间，解散后，房间中的每个用户都被动退出房间。

踢出某成员

```
public abstract void kickOffUserWithId(int userId);
```

踢出某用户。

房管／主播／会议主持 把某用户踢出聊天室。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户ID

禁言某成员 （1/2）

```
public abstract void shutUpUserWithId(int userId);
```

禁言某人。

房管/主播／会议主持 禁止某人发言。

参数

参数	类型	描述
userId	int	在房间中的用户的用户ID

弃用

v1.3.2 后推荐使用shutUpUserWithId(int userId, bool disable)替换。

禁言某成员 (2/2)

```
public abstract void shutUpUserWithId(int userId, bool disable);
```

是否禁言某成员。

房管/主播／会议主持 禁止某成员发言。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户ID
bool	disable	禁言/取消禁言

获取房间成员ID列表

```
public abstract UserList queryUserListOfRoom()
```

查询房间内成员ID列表，已废弃，请使用新接口 queryUserList

获取房间中所有成员ID信息。

```
public abstract void queryUserList(BRtcCallback.QueryRoomUsersCallback queryRoomUsersCallback);
```

查询房间内成员ID列表, 通过回调接口返回相关数据。

获取房间中所有成员ID信息。

返回

用户ID信息列表

获取房间成员详细列表

```
public abstract RtcRoomUserInfo[] getUserListOfRoom()
```

查询房间用户详细信息。

获取房间中所有参与者的详细信息列表。

返回

用户详细信息列表

发布/订阅相关接口

开始直播推流

```
public abstract void startPublish();
```

开始直播推流。

发布媒体流，开始直播。与直播转推接口configLiveServerWithUrl（推流地址通过该接口设置）配合使用。
调用时机：接收到房间登录成功事件IRtcEventHandler#onLoginSuccess之后；

注意 该接口仅直播场景使用。与stopPublish接口配对使用。

开始视频推流 public abstract void startVideoPublish();

开始视频推流，仅仅作用于 RtcParameterSettings.enablePruneSignal = true

停止直播推流

```
public abstract void stopPublish();
```

停止直播推流。

停止发布流，结束直播。

注意 该接口仅直播场景使用。与startPublish接口配对使用。

停止视频推流 public abstract void stopVideoPublish();

注意：停止视频推流，仅仅作用于 RtcParameterSettings.enablePruneSignal = true

角色转换

主播角色Constants#RoleType#RoleAnchor 支持推流和拉流

观众角色Constants#RoleType#RoleAudience仅仅支持拉流

```
public abstract void switchRole(@Constants.RoleType int roleType);
```

注意：仅仅作用于 RtcParameterSettings.enablePruneSignal = true，角色转换后会有 IRtcEventHandler#onSwitchRole事件回调

订阅某成员媒体流

```
public abstract void subscribeStreaming(long feedId);
```

订阅远端成员订阅流。

仅当RtcParameterSettings.AutoSubscribe设置为false（非自动订阅）时，手动订阅远端用户媒体流。渲染窗口通过 setRemoteDisplay(RTCVideoView remoteVideoView, long userId) 动态设置。

参数

参数	类型	描述
feedId	long	远端用户ID

订阅所有远端用户视频流 public abstract void subscribeAllRemoteVideoStreams();

可在加入房间前、后调用；加入房间前调用等同于配置HasRemoteVideo字段

订阅所有远端用户音频流 public abstract void subscribeAllRemoteAudioStreams();

可在加入房间前、后调用；加入房间前调用等同于配置HasRemoteAudio字段

订阅某成员视频流 public abstract void subscribeVideoStreaming(long feedId);

手动订阅-开始订阅指定用户视频流 or 自动订阅时，开始订阅指定用户视频流

参数

参数	类型	描述
feedId	long	远端用户ID

订阅某成员音频流 public abstract void subscribeAudioStreaming(long feedId);

手动订阅-开始订阅指定用户视频流 or 自动订阅时，开始订阅指定用户音频流

参数

参数	类型	描述
feedId	long	远端用户ID

停止订阅某成员媒体流

```
public abstract void stopSubscribeStreaming(long feedId);
```

停止订阅。

停止订阅流。

参数

参数	类型	描述
feedId	long	远端用户ID

停止订阅所有远端用户视频流

```
public abstract void stopSubscribeAllRemoteVideoStreams();
```

可在加入房间前、后调用；加入房间前调用等同于配置HasRemoteVideo字段

停止订阅所有远端用户音频流

```
public abstract void stopSubscribeAllRemoteAudioStreams();
```

可在加入房间前、后调用；加入房间前调用等同于配置HasRemoteAudio字段

停止订阅某成员视频流

```
public abstract void stopSubscribeVideoStreaming(long feedId);
```

停止订阅指定用户视频流 **参数**

参数	类型	描述
feedId	long	远端用户ID

停止订阅某成员音频流

```
public abstract void stopSubscribeAudioStreaming(long feedId);
```

停止订阅指定用户音频流 **参数**

参数	类型	描述
feedId	long	远端用户ID

消息相关接口

向某成员发送消息

```
public abstract void sendMessageToUser(String msg, long id);
```

发送消息。

id=0，房间内广播消息， id = userID，指定用户发送消息

参数

参数	类型	描述
msg	String	消息
id	long	流的ID号

设置用户属性

```
public abstract void setUserAttribute(String attribute);
```

用户属性设置。

设置用户属性。

参数

参数	类型	描述
attribute	String	用户属性

获取用户属性

```
public abstract void getUserAttribute(long feedid);
```

用户属性获取。

获取用户属性。

参数

参数	类型	描述
feedid	long	流ID

发送数据 （1/2）

```
public abstract void sendDataMessage(String data, boolean reliable);
```

发送Data数据。

Data 数据通道保证顺序性和完整性。 使用数据通道能够同时传输音视频之外的相关数据。

参数

参数	类型	描述
data	String	要发送的数据
reliable	boolean	是否可靠传输

注意 发送Data 数据， 需要先打开datachannel 功能： RtcParameterSettings cfg = RtcParameterSettings.getDefaultSettings(); cfg.HasData = true; mVideoRoom.setParamSettings(cfg,RtcParameterSettings.RtcParamSettingType.RTC_PARAM_SETTINGS_ALL);

发送数据 (2/2)

```
public abstract void sendDataMessage2(String data, long uid, boolean reliable);
```

发送Data数据。

Data 数据通道保证顺序性和完整性。 使用数据通道能够同时传输音视频之外的相关数据。 **注意** uid用于发送给指定用户消息，当开启enablePruneSignal = true; 情况下有效；

发送SEI消息

```
public abstract void sendSEIMsg(byte[] data, int repeatCount);
```

发送SEI消息。

Data 待发送的数据，最大支持 1KB（1000字节）的数据大小。

注意 发送频率不宜过多，防止影响正常音视频通讯

视频相关接口

开启本地预览

```
public abstract void startPreview();
```

本地预览。

打开camera，开始预览。

停止本地预览

```
public abstract void stopPreview();
```

停止预览。

关闭camera, 停止本地预览。

设置本地渲染窗口

```
public abstract void setLocalDisplay(RTCVideoView localVideoView);
```

设置本地视频预览窗口。

设置本地相机预览窗口。 外部采集模式下勿需调用。 调用时机：登录房间之前。

参数

参数	类型	描述
localVideoView	RTCVideoView	本地显示窗口,用于显示camera采集的视频数据

RTCVideoView 主要接口

接口	描述
setScalingType(ScalingType scalingType)	设置画面裁剪模式： SCALE_ASPECT_FIT：等比缩放，保持视频比例，保留视频全部内容，未被填满区域填充背景色。 SCALE_ASPECT_FILL:全屏裁剪，保持视频比例，铺满整个视图，视频超出部分被裁剪。 SCALE_ASPECT_BALANCED:保留视频全部内容，不裁剪，填充整个视图，画面可能变形。
setMirror(boolean mirror)	设置显示镜像
clearImage()	清屏，显示区域置黑。

设置远端渲染窗口 (1/2)

```
public abstract void setRemoteDisplay(RTCVideoView remoteVideoView, long feedId)
```

设置指定远端用户显示窗口。

可适用于1 v 1、1 v N等多种场景，支持RTCVideoView与User的动态绑定与更新，若不调用removeRemoteDisplay(long userId)主动释放VideoView(可缺省调用)，则Videoview在远端用户离开或本端登出房间时自动释放。 调用时机： 本端登录房间前或远端用登入房间并获取到对应的feedId均可，推荐在 `IRtcEventHandler#onStreamChangedState` 事件回调处调用。

参数

参数	类型	描述
remoteVideoView	RTCVideoView	远端画面显示窗口, 用于显示远端用户传输过来的视频数据
feedId	long	远端成员ID

设置远端渲染窗口 (2/2)

```
public abstract void setRemoteDisplay(RTCVideoView remoteVideoView, long feedId, String viewMediaTarget);
```

设置指定远端用户指定流的显示窗口。

可适用于1 v 1、1 v N等多种场景，支持RTCVideoView与User的指定流绑定，主要扩展一个用户同时推多路视频流时，viewMediaTarget指定具体用户的哪路流，常用于多流模式的屏幕分享，指定具体是用户的哪一路视频流。 调用时机：推荐在 `IRtcEventHandler#onStreamChangedState` 事件回调处调用。

参数

参数	类型	描述
remoteVideoView	RTCVideoView	远端画面显示窗口，用于显示远端用户传输过来的视频数据
feedId	long	远端成员ID
viewMediaTarget	String	远端成员ID推的哪一路流，可使用RtcParameterSettings.RtcMediaTarget做类型匹配

移除远端用户渲染窗口 (1/2)

```
public abstract void removeRemoteDisplay(long userId)
```

移除远端用户渲染窗口。

移除并释放远端用户所对应的渲染窗口，可通过setRemoteDisplay(RTCVideoView, long)接口再次设置渲染窗口。若不主动调用该接口释放渲染,则相应的渲染视图将在远端用户退出或本端登出房间时释放。

注意 仅与setRemoteDisplay(RTCVideoView remoteVideoView, long feedId)配对使用，可缺省调用。
窗口视图不见时，通过该接口释放渲染视图可优化系统资源占用。

参数

参数	类型	描述
userId	long	用户ID

移除远端用户渲染窗口 (2/2)

```
public abstract void removeRemoteDisplay(long userId, String viewMediaTarget);
```

移除远端用户渲染窗口。

移除并释放远端用户所对应的渲染窗口，可通过setRemoteDisplay(RTCVideoView, long)接口再次设置渲染窗口。若不主动调用该接口释放渲染,则相应的渲染视图将在远端用户退出或本端登出房间时释放。

参数

参数	类型	描述
userId	long	用户ID
viewMediaTarget	String	视图类型 一般使用 RtcParameterSettings.RtcMediaTarget.TARGET_VIDEO_DEFAULT

开启/关闭本地视频外部采集/编码

```
public abstract int setExternalVideoSource(boolean enable,
                                           boolean useTexture,
                                           Constants.ExternalVideoSourceType sourceType);
```

使能视频外部采集/外部编码。

参数

参数	类型	描述
isEnable	boolean	true：外部采集；false：内部采集
useTexture	boolean	true：使用纹理；false：不使用纹理
sourceType	Constants.ExternalVideoSourceType	ExternalVideoSourceType.VIDEO_FRAME：视频裸数据； ExternalVideoSourceType.ENCODED_VIDEO_FRAME：视频编码数据

```
public abstract int sendExternalVideoFrame(RTCVideoFrame frame);
```

发送外部采集视频数据

```
public abstract int sendExternalEncodedImage(Constants.RtcEncodedImage image);
```

发送外部编码视频数据

参数

参数	类型	描述
encodedWidth	int	视频宽
encodedHeight	int	视频高
captureTimeNs	long	视频采集时间戳
frameType	RtcEncodedImage.FrameType	FrameType.VideoFrameKey 关键帧，其他非关键帧
trackId	int	外部编码多路流唯一标识

设置某成员外部渲染器

```
public abstract void addExternalRenderer(long userId, RTCVideoExternalRender render);
```

设置外部渲染。

为指定用户设置外部视频渲染器,外部采集模式若不设置外部渲染器则使用默认外部渲染器。

参数

参数	类型	描述
userId	long	用户ID
render	RTCVideoExternalRender	渲染器

移除某成员外部渲染器

```
public abstract void removeExternalRenderer(long userId);
```

移除外部渲染器。

移除指定用户的外部渲染器。

参数

参数	类型	描述
userId	long	用户ID

设置某成员外部渲染Surface

```
public abstract void setExternalSurface(long userId, Surface remoteSurface);
```

设置外部渲染Surface。

设置外部渲染Surface，该接口需在主线程调用。

参数

参数	类型	描述
userId	long	用户ID
remoteSurface	Surface	surface

更新外部渲染Surface尺寸

```
public abstract void changeSurfaceSize(long userId, int width, int height);
```

外部surface 尺寸变更。

外部surface 尺寸变更。

参数

参数	类型	描述
userId	long	用户ID
width	int	宽
height	int	高

销毁指定用户对应的外部渲染surface

```
public abstract void destroyExternalSurface(long userId, Surface remoteSurface);
```

销毁指定用户对应的surface。

销毁指定用户对应的surface 及 外部渲染器。

参数

参数	类型	描述
userId	long	用户ID
remoteSurface	Surface	surface

获取远端视频尺寸

```
public abstract Constants.VideoDimension getRemoteVideoDimension(long userId);
```

获取视频video dimension。

当远端视频达到后，可通过该接口获取远端视频流的Dimension信息。

参数

参数	类型	描述
userId	long	用户ID

返回

RtcRoomVideoDimension

截屏相关

```
 /** SnapShotHelper 对象构建 */
public SnapShotHelper(RTCVideoView renderer, Handler handler);

/** 开始截屏 */
public void takeSnapShot(final String filePath, SnapShotCallback callback);
```

参数

参数	类型	描述
render	RTCVideoView	需要截屏的VideoView，截屏本地输入LocalDisplay,截屏远端输入RemoteDisplay
handler	Handler	暂时默认输入null
filePath	String	文件保存路径地址默认是jpg格式，如果使用外部存储需要有写权限
callback	SnapShotCallback	截屏事件回调

```
/**
 * 截图回调
 */
public interface SnapShotCallback {
    /**
     * 截图
     * @param success 是否截图成功
     * @param description 截图成功则该参数返回文件路径,否则返回出错描述
     */
    void onSnapShotTake(boolean success, String description);
}
```

视频前处理相关

设置背景分割、虚化

```
public abstract void enableHumanSeg(boolean enable, BRTCEffectParams humanSegParams);
```

启停背景分割或虚化功能，可在通话前、通话中设置。

参数

参数	类型	描述
enable	boolean	启动或停止背景分割功能
humanSegParams	BRTCEffectParams	背景分割或虚化配置

BRTCEffectParams 定义

其中BRTCEffectParams属性定义如下：

类型	属性	默认值	描述
String	resourcePath	-	特效资源地址，通常背景分割或虚化会提供一个特效包，此参数为特效包在本地的解压后的路径。
String	resourceId	-	特效资源ID。

设置实时视频水印

```
public abstract void enableWatermark(boolean enable, BRTCWatermarkParams watermarkParams);
```

启停水印功能，可在通话前、通话中设置。

参数

参数	类型	描述
enable	boolean	启动或停止实时视频功能
watermarkParams	BRTCWatermarkParams	水印参数配置

BRTCWatermarkParams 定义

其中BRTCWatermarkParams属性定义如下：

类型	属性	默认值	描述
List	watermarkParamList	-	水印参数配置集合，多条水印可共存显示。

每一条水印存储在BRTCWatermarkParam中。

BRTCWatermarkParam 定义

类型	属性	默认值	描述
WatermarkType	watermarkType	-	类型：image、string、time。
Rect	rect	-	水印位置，左上右下，像素值。 当类型是图片时，默认以rect计算的width和height为准，若未设置right、bottom，则以图片宽高为准。 当类型是字符串，将忽略bottom，由文字大小来决定最终的高度。 当类型是时间时，将忽略right和bottom，以生成的时间字符串为准。
String	watermarkResource	-	资源地址。 image（固定资源图片（png） 或 绝对目录文件路径）。 字符串（静态字符串 （字符长度有最大限制，可根据设置的宽度，自动换行，居左对齐））。 时间（默认如果不设置时，使用yyyy-MM-dd HH:mm:ss format，可配置 yyyy-MM-dd\nHH:mm:ss 其中增加\n来进行换行）。
float	textSize	13	文字大小，像素。
int	textColor	Color.rgb(255, 255, 255)	文字颜色。
float	shadowRadius	1f	文字阴影半径。
float	shadowDx	0f	文字阴影x偏移。
float	shadowDy	1f	文字阴影y偏移。
int	shadowColor	Color.DKGRAY	文字阴影颜色。

WatermarkType 定义

类型	值	描述
IMAGE	0	图片，可为本地drawable资源图或者存储在本地的图片
STRING	1	字符串
TIME	2	时间

白板

初始化

```
    // Constant.Settings 白板配置
    public static class Settings {
        public String loadUrl;

        public String channelName;
        public String appId;

        public String token;

        public boolean debug = false;

        /** 标记观众 */
        public boolean isAudience = false;

        public long uid;
    }

    // 创建白板对象
    public static RtcWhiteBoard RtcWhiteBoard.init(Context context, RtcWhiteBoardHandler handler, Constant.Settings
settings)
```

获取白板View

```
    View getView();
```

- 注意 初始化后可以通过该接口后获取到白板view

加载白板

```
    void load();
```

- 注意 初始化后可以通过该接口加载白板

设置画笔

```
// 设置画笔类型，支持自由绘画，剪头绘画、直线、矩形、菱形、椭圆、文本、激光笔
void  selectToolType(@Constant.ToolType int type);

// 设置画笔颜色
void selectBrushColor(@Constant.ColorType int color);
```

清除白板

```
    void clear();
```

- 注意 加载后可以清除

销毁版本

```
    void destroy();
```

- 注意 销毁对象

本地录制相关接口

获取录制功能入口

```
public abstract IRtcMediaRecorder getBRTCMediaRecorder();
```

录制功能通过此接口获取到IRtcMediaRecorder后，调用相关接口进行录制操作。

返回

IRtcMediaRecorder

IRtcMediaRecorder定义 | 接口 | 描述 | | --- | --- | | int startRecording(BRTCMediaRecorderParams params)|启动录制，BRTCMediaRecorderParams为启动录制的参数配置，接口返回值0调用接口成功，否则调用接口失败 | | int stopRecording()| 停止录制，返回值0调用接口成功，否则调用接口失败 | | void release()| 释放录制资源，调用后不可再调用其他接口| | void setMediaRecorderCallback(BRTCMediaRecorderCallback callback)| 设置录制回调，详细参见BRTCMediaRecorderCallback定义| | boolean isReleased()| 获取是否已释放状态，true表示已释放，false表示未释放|

BRTCMediaRecorderParams定义 |类型 | 属性 |默认值|描述 | | --- | --- | --- | --- | |BRTCMediaRecorderType |mediaRecorderType|BRTCMediaRecorderType.AUDIO_VIDEO|录制类型。包含纯音频、纯视频、音视频、混流纯音频、混流纯视频、混流音视频。| |BRTCMediaMixType |mediaMixType|BRTCMediaMixType.PIP|混流录制画面布局类型。包含画中画、主次布局、平铺布局。| |String |storagePath|-|录制文件存储位置。| |BRTCMediaEncodeParams |encodeParams|-|录制参数，参见BRTCMediaEncodeParams定义。| |long |maxRecordDuration|-|录制最大时长，单位毫秒。| |long |infoUpdateInterval|-|录制信息更新间隔，单位毫秒。|

BRTCMediaRecorderType定义 |类型 |值|描述 | | ---|--|--- | |AUDIO_VIDEO|0|音视频| |AUDIO_ONLY |1|纯音频| |VIDEO_ONLY |2|纯视频| |MIX_AUDIO_VIDEO |3|混流音视频| |MIX_AUDIO_ONLY |4|混流纯音频| |MIX_VIDEO_ONLY |5|混流纯视频|

BRTCMediaMixType定义 |类型 |值|描述 | | ---|--|--- | |PIP|0|画中画布局：远端第一路为大画面，其余画面悬浮于画布底部，子画面从右向左依次平铺，子画面的宽为整个画布宽的1/5，高为整个画布高的1/5。| |PAS |1|主次平铺布局：远端第一路为大画面，其余画面自上而下依次垂直排列于右侧，左侧大画面的宽为整个画布宽的4/5，左侧大画面的高为整个画布高；
右侧小画面的宽为整个画布宽的1/5，右侧小画面的高为整个画布高的1/5。| |TILE |2|平铺布局：本地画面作为第一路，根据流数量自动调整每个画面的大小，每个画面大小一致；
画面数为1时，画面的宽和高分别为整个画布宽和高；
画面数为2时，每个小画面的宽为整个画布宽的1/2，每个小画面的高为整个画布高；
画面数大于2小于等于4时，每个小画面的宽和高分别为整个画布宽和高的 1/2；
画面数大于4小于等于6时，每个小画面的宽为整个画布宽的1/2，每个小画面的高为画布高的1/3|

BRTCMediaRecorderParams定义 |类型 | 属性 |默认值|描述 | | --- | --- | --- | --- | |String |videoCodec|MediaFormat.MIMETYPE_VIDEO_AVC|视频编码格式，默认H264。| |int |videoWidth|-|视频宽度。| |int |videoHeight|-|视频高度。| |int |videoFps|-|视频帧率。| |int |videoIFrameInterval|5|I帧间隔。| |int |videoBitrate|-|视频码率。| |String |audioCodec|MediaFormat.MIMETYPE_AUDIO_AAC|音频编码格式，默认AAC。| |int |audioSampleRate|-|采样率。| |int |audioChannel|-|声道数。| |int |audioBitrate|-|音频码率。|

BRTCMediaRecorderCallback定义

接口	描述
void onRecordStateChanged(int state, int code)	录制状态改变回调，参见下面的state及code定义
void onRecordInfoUpdate(long durationMs, long fileSize)	录制信息回调更新，durationMs录制时长， fileSize文件大小

state回调状态

类型	值	描述
RECORDER_STATE_ERROR	-1	录制状态：错误
RECORDER_STATE_START	1	录制状态：开始
RECORDER_STATE_STOP	2	录制状态：停止

code状态码

类型	值	描述
RECORDER_CODE_SUCCESS	0	成功
RECORDER_CODE_ERROR_PARAMS_INVALID	-100	参数无效
RECORDER_CODE_ERROR_INITIALIZED	-101	初始化异常
RECORDER_CODE_ERROR_OVER_MAX_DURATION	-102	超过最大时长
RECORDER_CODE_ERROR_RECORDING_STATE	-103	录制状态异常
RECORDER_CODE_ERROR_ENGINE_RELEASED	-104	BRTC引擎已释放
RECORDER_CODE_ERROR_WRITE_FAILED	-105	写入异常
RECORDER_CODE_ERROR_RECORDER_RELEASED	-106	录制已释放
RECORDER_CODE_ERROR_MUXER_STOP	-107	合成器停止异常
RECORDER_CODE_ERROR_MUXER_START	-108	合成器启动异常

摄像头相关接口

切换摄像头

```
public abstract void switchCamera();
```

切换摄像头。

前/后摄像头切换。

设置摄像头ID

```
public abstract void setCameraID(int cameraId);
```

设置指定的摄像头ID。 调用时机： 登录房间之前。

参数

参数	类型	描述
cameraId	int	摄像头ID

关闭/开启本地视频采集

```
public abstract void muteCamera(boolean muted);
```

关闭/打开摄像头。

在关闭摄像头后，本地无法预览，且不传输本地视频数据给对方。

参数

参数	类型	描述
muted	boolean	是否打开摄像头

音频相关接口

关闭/开启麦克风静音 public abstract void muteMicphone(boolean muted);

开启/关闭 麦克风静音（不关闭手机麦克风设备）。

如果需要关闭麦克风设备，请查看enableMicCapture接口。

参数

参数	类型	描述
muted	boolean	true表示要静音， false表示取消静音

关闭/开启麦克风采集 public abstract void enableMicCapture(boolean enableMic);

开启/关闭 麦克风采集（关闭手机麦克风设备）。

参数

参数	类型	描述
enableMic	boolean	true表示要打开麦克风设备采集， false表示关闭麦克风设备采集

切换扬声器/听筒

```
public abstract void switchLoundSpeaker();
```

开关扬声器。

听筒与扬声器切换(免提功能)

预置听筒/扬声器

```
public abstract void presetLoudSpeaker(boolean isPresetLoudSpeaker);
```

预置听筒/扬声器。

参数

参数	类型	描述
isPresetLoudSpeaker	boolean	true表示扬声器，false表示听筒

禁止/开启音频输出

```
public abstract void muteSpeaker(boolean mute)
```

开启/关闭音频输出。

参数

参数	类型	描述
mute	boolean	true：禁止音频输出， false：开启音频输出 （默认）

设置远端用户音频播放音量

```
public abstract void setUserPlaybackVolume(long userId, int volume)
```

设置远端用户音频播放音量。

参数

参数	类型	描述
userId	long	远端用户id
volume	int	设置的音量范围 [0,400] ，>100可增益

设置远端所有用户音频播放音量

```
public abstract void setPlaybackVolume(int volume)
```

设置所有远端用户播放音量。

参数

参数	类型	描述
volume	int	设置的音量范围 [0,400] ，>100可增益

设置采集或自定义采集的音量

```
public abstract void setAudioCaptureVolume(int volume)
```

设置采集音量或者自定义采集的音量。

参数

参数	类型	描述
volume	int	设置的音量范围 [0,400] ，>100可增益

查询扬声器是否开启

```
public abstract boolean isSpeakerOn()
```

查询扬声器是否开启。

获取房间成员语音激励列表

```
public abstract RtcRoomAudioLevel[] getRemoteAudioLevels();
```

语音激励接口。

获取用户语音激励列表。

返回

RtcRoomAudioLevel[]

启停用户音量提示回调

```
public abstract void enableAudioVolumeIndication(boolean enable, int interval)
```

启停用户音量提示，启用后，会通过onAudioVolumeIndication回调，与getRemoteAudioLevels接口的差别在于，getRemoteAudioLevels为主动调用一次，此接口会周期性回调。

参数

参数	类型	描述
enable	boolean	是否启动音量提示回调
interval	int	音量提示回调周期，单位毫秒

设置是否拉取某成员音频流

```
public abstract void setRemoteAudioPlayState(boolean stats, long userId);
```

指定远端音频暂停/恢复播放。

通过控制音频流拉取，控制指定远端用户音频暂停/恢复播放。

参数

参数	类型	描述
stats	boolean	true：拉取，false：停止拉取。默认拉取
userId	long	用户ID

设置是否拉取某成员视频流

```
public abstract void setRemoteVideoPlayState(boolean stats, long userId);
```

指定远端画面暂停/恢复播放。

通过控制流的拉取与否，指定远端画面暂停/恢复播放。

参数

参数	类型	描述
stats	boolean	true：拉取；false：停止拉取
userId	long	用户ID

设置音频输出设备

```
public abstract void setEnableSpeakerphone(boolean enable)
```

设置音频输出设备，扬声器/听筒。

参数

参数	类型	描述
soundMod	boolean	音频输出方式：扬声器/听筒

设置自定义音频采集 // 设置外部音频采集 public abstract void enableExternalAudioRecord(boolean isEnable);

```
// 获取外部音频采集状态
public abstract boolean isExternalAudioRecord();

// 获取外部音频采集回调， 用于输入音频数据 【1/2】 每次输入音频长度为对应音频采样率10ms （48k采样率每次输入960）
public abstract int pushExternalAudioFrame(byte[] data, long timestamp, int sampleRate, int channels);

// 获取外部音频采集回调， 用于输入音频数据 【2/2】 每次输入音频长度为对应音频采样率10ms （48k采样率每次输入960）
public abstract int pushExternalAudioFrame(ByteBuffer data, long timestamp, int sampleRate, int channels);
```

开启自定义音频采集（外部音频采集）

设置自定义音频渲染 // 设置外部音频渲染 public abstract void enableExternalAudioRender(boolean isEnable);

```
// 获取外部音频渲染状态
public abstract boolean isExternalAudioRender();

// 设置远端音频渲染回调, obs.onPlaybackAudioFrame(RTCAudioSamples sample);
public abstract void registerAudioFrameObserver(Constants.IAudioFrameObserve obs);
```

开启自定义音频渲染（外部音频渲染）

设置本地音频数据回调

```
// 设置本地音频渲染回调, obs.onRecordAudioFrame(RTCAudioSamples sample);
public abstract void registerAudioFrameObserver(Constants.IAudioFrameObserve obs);
```

设置本端音频数据回调。

在 登录房间前设置，进入通信后通过该接口回调本端PCM音频数据。

开启默认音频自动增益

```
public abstract void enableAgc(boolean enable)
```

开启音频自动增益。

参数

参数	类型	描述
enable	boolean	开启/关闭

开启默认音频自动噪声抑制

```
public abstract void enableAns(boolean isAns)
```

开启默认音频自动噪声抑制。

参数

参数	类型	描述
enable	boolean	开启/关闭

开启默认音频回声消除

```
public abstract void enableAec(boolean enable)
```

开启默认音频回声消除。

参数

参数	类型	描述
enable	boolean	开启/关闭

开启/关闭变声能力

```
public abstract void enableVoiceChange(boolean enableVoiceChange)
```

开启/关闭变声能力。

参数

参数	类型	描述
enableVoiceChange	boolean	开启/关闭变声

设置变声类型

```
public abstract void setVoiceChangeType(BdRTCVoiceChangeType voiceChangeType)
```

设置变声类型。

参数

参数	类型	描述
voiceChangeType	BdRTCVoiceChangeType	变声类型

BdRTCVoiceChangeType 类型	描述
AUDIO_EFFECT_ORIGIN	原声
AUDIO_EFFECT_LUOLI	萝莉
AUDIO_EFFECT_DASHU	大叔
AUDIO_EFFECT_ZHENGTAI	正太
AUDIO_EFFECT_FEIZAI	肥仔
AUDIO_EFFECT_KTV	KTV
AUDIO_EFFECT_FOZU	佛祖
AUDIO_EFFECT_SINGER_FINE	K歌

播放本地音频文件

```
public abstract IRtcAudioManager getBRTCAudioManager();
```

获取音频控制器
包括混音、播放本地音频文件（支持MP3，AAC等Android系统解码器可解码的音频文件格式，并推送到远端）。

IRtcAudioManager 定义 | 接口 | 描述 | | --- | --- | | void enableAudioMix(boolean isEnable)|是否启用音频混音，原BaiduRtcRoom混音接口，移到该音频管理接口下
| | int setExternalMixAudio(ByteBuffer music, int length)| 原BaiduRtcRoom外部音频混音接口，移到该音频管理接口下 | | int setExternalMixAudio(String filePath, boolean pushRemote, int cycle, long startTime);| 本地音频文件混音推流 | ||filePath: 本地音频文件地址| ||pushRemote: 是否推流到远端
值：
true：推流，同时本地播放；
false：不推流，只本地播放；| ||cycle: 播放次数
值：
0：不播放；
1：播放1次；
-1（负数）：循环播放；| ||startTime: 开始播放位置，默认是0| |void pauseExternalMixAudio()|暂停外部混音（停止播放，停止推流）| |void resumeExternalMixAudio()|继续外部混音（从之前pause的位置继续）|

动态加载SO

RTCLoadManager SO加载管理帮助类

设置cpu类型

```
public void setDefaultCpuType(String cpuType);
```

参数

参数	类型	描述
cpuType	String	cpu类型，armeabi-v7a、arm64-v8a

注册回调

```
public void registerCallback(LoadListener callback);
```

参数

参数	类型	描述
callback	LoadListener	加载回调接口

检查当前是否已经加载

```
public void queryLibsDownloaded(Context context, LibsStateListener libsStateListener);
```

参数

参数	类型	描述
context	Context	上下文
libsStateListener	LibsStateListener	状态回调

演示代码

```
 // so 外部提前加载 需必须
final RTCLoadManager loadManager = RTCLoadManager.getInstance(this);
loadManager.setDefaultCpuType("arm64-v8a");
loadManager.registerCallback(mLoadListener);
loadManager.queryLibsDownloaded(this,
        new RTCLoadManager.LibsStateListener() {
            @Override
            public void onLibsHasDownloaded(boolean hasDownload) {
                Log.i(TAG, "onLibsHasDownloaded :" + hasDownload);
                if (!hasDownload) {
                    Log.i(TAG, "loadLibraries :" + RTCLoadManager.getDefaultDownloadUrl());
                    loadManager.loadLibraries("arm64-v8a", null);
                }
            }
        });
```

外部编码

开启外部编码 public abstract void enableExternalVideoEncode(boolean isEnable);

开启外部编码

`enable=true`视频外部编码, 开启外部编码需要同步设置编码的宽高和码率, 参考`cfg.VideoResolution = "1280x720-1500kbps`

发送外部编码数据 public abstract void sendExternalEncodedImage(RtcEncodedImage image);

视频裸流发送接口, 客户自定义视频捕获,视频编码环节后, 调用本接口发送编码后的数据.

```
public static class RtcEncodedImage {
    // 编码后的H264/H265数据
    public final ByteBuffer buffer;
    // 编码视频宽, 像素
    public final int encodedWidth;
    // 编码视频高, 像素
    public final int encodedHeight;
    // 视频采集时间, 单位纳秒, 参考System.nanoTime()
    public final long captureTimeNs;
    // 编码视频帧类型, 关键帧VideoFrameKey和非关键帧VideoFrameDelta
    public final RtcEncodedImage.FrameType frameType;
    // 用于区分不同外置编码器
    public final int trackId;
}
```

转推配置相关接口

配置媒体转推参数

```
public abstract boolean configLiveServerWithUrl(String url,
                                                boolean isMix,
                                                boolean isRecording,
                                                String mixTemplate,
                                                RtcLiveTransferMode transferMode);;
```

server端推流参数配置。

该接口用于配置Server推流的参数，聊天室模式：在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间；主播转推模式：主播推向不同的直播房间

参数

参数	类型	描述
url	String	rtmp推流地址
isMix	boolean	是否做混流处理
isRecording	boolean	是否录制
mixTemplate	String	混流模板
transferMode	RtcLiveTransferMode	转推模式： 聊天室模式， 主播转推模式

混流模板名称	说明
side_by_side_primary_360p_4_3	主次平铺, 分辨率 480x360 (固定码率、帧率)
side_by_side_primary_360p_4_3-xx-xx	主次平铺, 分辨率 480x360 (自定义码率、帧率) 下面模板也可以设置
side_by_side_primary_360p_16_9	主次平铺, 分辨率640x360
side_by_side_primary_480p_4_3	主次平铺, 分辨率640x480
side_by_side_primary_480p_16_9	主次平铺, 分辨率854x480
side_by_side_primary_540p_16_9	主次平铺, 分辨率960x540
side_by_side_primary_720p_4_3	主次平铺, 分辨率960x720
side_by_side_primary_720p_16_9	主次平铺, 分辨率1280x720
side_by_side_equal_360p_4_3	平铺模式(大小相等), 分辨率480x360
side_by_side_equal_360p_16_9	平铺模式(大小相等), 分辨率640x360
side_by_side_equal_480p_4_3	平铺模式(大小相等), 分辨率640x480
side_by_side_equal_480p_16_9	平铺模式(大小相等), 分辨率854x480
side_by_side_equal_540p_16_9	平铺模式(大小相等), 分辨率960x540
side_by_side_equal_720p_4_3	平铺模式(大小相等), 分辨率960x720
side_by_side_equal_720p_16_9	平铺模式(大小相等), 分辨率1280x720
picture_in_picture_bottom_360p_4_3	画中画模式, 分辨率480x360
picture_in_picture_bottom_360p_16_9	画中画模式, 分辨率640x360
picture_in_picture_bottom_480p_4_3	画中画模式, 分辨率640x480
picture_in_picture_bottom_480p_16_9	画中画模式, 分辨率854x480
picture_in_picture_bottom_540p_16_9	画中画模式, 分辨率960x540
picture_in_picture_bottom_720p_4_3	画中画模式, 分辨率960x720
picture_in_picture_bottom_720p_16_9	画中画模式, 分辨率1280x720
side_by_side_primary_480p_9_16	主次平铺, 分辨率480x854（竖屏）
side_by_side_primary_540p_9_16	主次平铺, 分辨率540*960（竖屏）
side_by_side_primary_544p_9_16	主次平铺, 分辨率544*960（竖屏）
side_by_side_primary_720p_9_16	主次平铺, 分辨率720*1280（竖屏）
side_by_side_primary_1080p_9_16	主次平铺, 分辨率1080*1920（竖屏）
side_by_side_equal_480p_9_16	平铺模式, 分辨率480x854（竖屏）
side_by_side_equal_540p_9_16	平铺模式, 分辨率540*960（竖屏）
side_by_side_equal_544p_9_16	平铺模式，分辨率544*960（竖屏）
side_by_side_equal_720p_9_16	平铺模式，分辨率720*1280（竖屏）
side_by_side_equal_1080p_9_16	平铺模式，分辨率1080*1920（竖屏）
picture_in_picture_bottom_480p_9_16	画中画模式, 分辨率480x854（竖屏）
picture_in_picture_bottom_540p_9_16	画中画模式，分辨率540*960（竖屏）
picture_in_picture_bottom_544p_9_16	画中画模式，分辨率544*960（竖屏）
picture_in_picture_bottom_720p_9_16	画中画模式，分辨率720*1280（竖屏）
picture_in_picture_bottom_1080p_9_16	画中画模式，分辨率1080*1920（竖屏）

返回

true 设置成功， false 失败。

通知相关接口

房间事件更新回调

用户登录成功回调

```
public void onLoginSuccess(long roomId, long uid, int elapsed);
```

房间用户登录成功通知。

参数

参数	类型	描述
roomId	long	房间ID
uid	long	用户id
elapsed	int	登录耗时 毫秒

链接状态变更

```
public void onConnectionStateChanged(int state, int reason);
```

链接状态发生变化

事件	值	含义	备注
CONNECTION_STATE_CONNECTED	3	链接成功	无
CONNECTION_STATE_DISCONNECTED	1	链接断开	无
CONNECTION_STATE_CONNECTING	2	链接中	无
CONNECTION_STATE_RECONNECTING	4	重连接	无

远端用户加入 public void onRemoteUserJoinRoom(long uid, String name);

远端用户加入房间

参数

参数	类型	描述
uid	long	用户uid
name	String	用户昵称

远端用户离开

```
public void onRemoteUserLeaveRoom(long uid, String name);
```

远端用户离开房间

参数

参数	类型	描述
uid	long	用户uid
name	String	用户昵称

远端视频首帧渲染

```
public void onFirstVideoFrame(long uid, int width, int height);
```

视频首帧渲染

远端用户视频流状态变更

```
public void onRemoteUserVideoAvailable(long uid, String name, boolean available);
```

远端视频可用/不可用，不清楚远端用户是否推流视频，可以根据这个判断；

远端用户音频流状态变更

```
public void onRemoteUserAudioAvailable(long uid, String name, boolean available);
```

远端音频可用/不可用，不清楚远端用户是否推流音频，可以根据这个判断；

用户被禁言

```
public void onUserShutUp(long uid);
```

用户被禁言

用户被解除禁言

```
public void onUserDisShutUp(long uid);
```

用户被解除禁言

房间被解散

```
public void onRoomDisbanded(long roomId);
```

房间被解散

被踢出房间

```
public void onUserKickOff(long uid);
```

用户被踢出房间

网络状态变更

```
public void onNetworkChanged(boolean isAvailable);
```

网络变化状态通知

发送码率回调

```
public void onSendBitrateEstimation(int rate, String info);
```

发送码率预估回调

推流音视频状态变更

```
public void onSendMediaState(int type, int state);
```

音视频推流状态变更

参数

参数	类型	描述
type	int	Constants.MediaType.MEDIA_VIDEO_TYPE = 1 视频类型 Constants.MediaType.MEDIA_AUDIO_TYPE = 2 音频类型
state	int	Constants.SEND_MEDIA_OK_STATE = 1 发送成功 Constants.SEND_MEDIA_FAIL_STATE = 2 发送失败 Constants.SEND_MEDIA_SUSPEND_STATE = 3 发送中断

媒体链路状态变更

```
public void onMediaStateChanged(boolean uplink, int state);
```

媒体链路状态发生变更

参数

参数	类型	描述
uplink	boolean	true: 上行， false: 下行
state	int	Constants.MediaState.MEDIA_DISCONNECTED_STATE = 2 断开 Constants.MediaState.MEDIA_CONNECTED_STATE = 3 建联成功

远端流变更

```
public void onStreamChangedState(long uid, String name, int opt, ArrayList<Constants.RtcStream> list);
```

远端流状态发生变更

参数

参数	类型	描述
uid	long	用户uid
name	String	用户昵称
opt	int	Constants.ON_STREAM_CHANGE_COMING = 1 远端流到达 Constants.ON_STREAM_CHANGE_LAVING = 2 远端流离开
list	ArrayList<>	流信息

屏幕分享状态变更

```
public void onScreenCaptureStateChanged(int state, int code);
```

屏幕分享状态变更

外部渲染视频大小变更

```
public void onVideoSizeChanged(long uid, int width, int height);
```

外部渲染场景视频大小发生变化

收到用户消息/广播

```
public void onUserMessage(long uid, String message);
```

收到用户消息、或者广播消息

收到用户SEI消息

```
public void onRecvSEI(long uid, byte [] msg);
```

收到用户SEI消息

收到用户属性

```
public void onUserAttribute(long uid, String message);
```

收到用户属性

stopRealy状态

```
public void onStopMediaRelayState(int state);
```

stopRealy状态变更

startRealy状态 public void onStartMediaRelayState(int state); startRealy状态变更

直播推流状态变更

```
public void onLivePublishState(Constants.RtcLiveTransferMode mode, String url, int state);
```

直播推流状态变更

错误信息更新回调

```
 public void onError(int error, String msg, Bundle bundle);
```

SDK内部出错；

```
public void onWarning(int code, String msg, Bundle bundle);
```

SDK内部出警告；

事件	值	含义	备注
ERR_LOGIN_ROOM_PARAMS	500	登录参数错	Appid/Token等参数错误，不能重连接
ERR_CHANNEL_IO_EXCEPTION	501	信令通道网络异常断开	不可恢复错误，可进行重连。
ERR_CHANNEL_DISCARD_BY_REMOTE	502	信令通道被远端异常关闭	不可恢复错误，可进行重连。
ERR_AUDIO_DEVICE_RECORD	503	音频采集器异常	不可恢复错误，可进行重连。
ERR_JOIN_ROOM_PARAMS	504	登录房间异常	不可恢复错误，可进行重连。
ERR_SET_EXTERNAL_SURFACE	505	设置外部渲染异常	不可恢复错误
ERR_RECONNECT_OVER_MAX_COUNT	506	重连接超限50次	不可恢复错误, 需要登出再次登录
ERR_INTERNAL	507	SDK内部未知错误	不可恢复错误，不能重连接
ERR_SO_LATER_DOWNLOADING_FAIL	508	SDK下载SO失败	不可恢复错误，不能重连接
ERR_MEDIA_SESSION_DESCRIPTION	509	SDK内部异常	不可恢复错误，能重连接
ERR_PUBLISH_VIDEO_FAILED	510	推流视频失败	不可恢复错误，能重连接
ERR_PUBLISH_AUDIO_FAILED	511	推流音频失败	不可恢复错误，能重连接
WARNING_SUBSCRIBE_FAILED	600	订阅异常	不可恢复错误，能重连接
WARNING_PUBLISH_VIDEO_FAILED	601	推流视频失败	不可恢复错误，能重连接
WARNING_PUBLISH_AUDIO_FAILED	602	推流音频失败	不可恢复错误，能重连接
WARNING_REPEAT_SUBSCRIBE	603	重复订阅	不可恢复错误，能重连接

房间数据接收回调

```
public abstract void onRoomDataMessage(ByteBuffer data);
```

数据接收。

该callback返回当前RTC Engine收到的数据消息。

参数

参数	类型	描述
data	ByteBuffer	引擎返回的数据消息

用户音量提示回调

```
public abstract void onAudioVolumeIndication(Constants.RtcRoomAudioLevel[] audioLevels);
```

该callback通过enableAudioVolumeIndication接口开启或停止，启动后，按照设置的时间间隔返回房间用户音量信息。

参数

参数	类型	描述
audioLevels	RtcRoomAudioLevel[]	房间用户音量信息

统计

```
public void onEngineStatisticsInfo(boolean uplink, List<HUDStatistics> statistics);
```

该回调中获取到所有媒体流统计信息

其他接口

获取SDK版本

```
public static String version()
```

SDK 版本号获取。

获取当前SDK版本号

返回

返回当前SDK版本号

开启/关闭调试信息输出

```
public static void setVerbose(boolean bOnVerbose)
```

是否打开调试信息。

建议在初始化 SDK 前调用。建议在调试阶段打开此开关，打开此开关后，将打开日志信息，方便调试。

参数

参数	类型	描述
bOnVerbose	boolean	是否打开调试信息，true 打开，false 不打开。默认为 false

开启/关闭质量数据上报

```
public abstract void enableStatsToServer(boolean isEnable, String qualityMonitorEnv)
```

RTC质量监控数据上报。

预置接口，监控信息上报开关 当打开开关时，上报帧率、码率、分辨率、丢包率等监控信息到服务端，console可查。

参数

参数	类型	描述
isEnable	boolean	是否打开RTC质量监控数据上报，true 打开，false不打开。默认为 false
qualityMonitorEnv	String	线上环境："online" 沙盒："qa"。 默认值为 "online"

屏幕分享相关

目前支持两种屏幕分享模式：**多进程模式**、**多流模式**。

多进程模式：是在业务侧启动一个进程，创建一个新的BRTC实例，作为屏幕分享的特定用户加入房间，此进程用户只发送屏幕流，不订阅其他用户，此模式下支持服务端录制。

多流模式：是在已创建的通讯链路中，增加、删除一路视频流用于发送、停止屏幕流，此模式下不支持服务端录制。

屏幕分享参数 新增屏幕分享参数结构，用于配置屏幕分享音频、视频开启状态及详细参数信息，参考RtcParameterSettings中定义。

多进程模式屏幕分享API

屏幕进程初始化 public static synchronized BaiduRtcRoom initWithConfig(Constants.BaiduRtcRoomConfig config)

屏幕分享独立进程下的，SDK初始化。

初始化SDK并创建RTC房间实例 初始化 SDK 时调用，初始化 SDK 失败会导致 SDK 功能异常。

参数

参数	类型	描述
context	Context	Android上下文环境
appId	String	百度为App签发的 App ID, 用于识别应用, 全局唯一，详见创建应用。
token	String	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。 详见应用鉴权。
cpuType	String	设置空字符串”“
isEnableSoLaterLoad	boolean	设置false
initialAudioManager	boolean	设置false，表示是否默认初始化AudioManager，由于屏幕分享不采集音频，由推流主进程用户发送音频，则此处不需要初始化

返回

BaiduRtcRoom 实例对象：成功； null：失败；

多流模式屏幕分享API

启动屏幕分享 public abstract boolean startShareScreen();

当使用**多流模式**的屏幕分享时，可使用此接口启动屏幕分享。 若使用多进程的模式，则可直接配置RtcParameterSettings的参数：HasScreen、screenIntentData、videoEncodeParams配置后，登录房间即可。

停止屏幕分享 public abstract boolean stopShareScreen(); 当使用**多流模式**的屏幕分享时，可使用此接口停止屏幕分享。 若使用多进程的模式，则可直接登出房间。

获取远端加入的用户的流信息 public void onStreamChangedState(long uid, String name, int opt, ArrayList<Constants.RtcStream> list); 当使用**多流模式**的屏幕分享时，可使用此接口获取加入的用户有哪几路流，进而调用setRemoteDisplay关联不同流与RTCVideoView的关系。 一般在`Constants#ON_STREAM_CHANGE_COMING`事件回调处调用。

参数

参数	类型	描述
userId	long	远端加入的用户ID

两种模式通用API

更新屏幕分享参数

```
public abstract void updateScreenShareParams(BRTCScreenShareParams parameters);
```

屏幕分享中，可更新屏幕分享参数，可用于**多流和多进程两种模式**。

通话前测速相关

Lastmile初始化

Lastmile初始化, 初始化Lastmile并创建网络测速实例。

```
public static synchronized BaiduRtcLastmileImp initProbeTest(String appId,
                                                             String tokenStr,
                                                             int expectedUplinkBitrate,
                                                             int expectedDownlinkBitrate,
                                                             BaiduRtcLastmileDelegate delegate);
```

初始化网络测速时调用，初始化网络测速失败会导致网络测速功能异常。

参数

参数	类型	描述
appId	String	百度为App签发的 App ID, 用于识别应用, 全局唯一，详见创建应用。
tokenStr	String	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。详见应用鉴权。
expectedUplinkBitrate	int	用户期望的最高发送码率，单位为 bps，范围为 100000～5000000。
expectedDownlinkBitrate	int	用户期望的最高接收码率，单位为 bps，范围为 100000～5000000 。
delegate	BaiduRtcLastmileDelegate	遵循 BaiduRtcLastmileDelegate 协议的代理对象 。

返回

BaiduRtcLastmileImp 实例对象：成功； null：失败；

Lastmile结束 public abstract void stopProbeTest();

结束网络探测, 可以在网络探测回调函数onLastmileProbeResult()中调用, 也可以手动触发结束。

BaiduRtcLastmileDelegate代理对象

```
public abstract void onLastmileProbeResult(LastmileProbeResult result);
```

参数

参数	类型	描述
result	LastmileProbeResult	开始通话前网络质量探测，向用户反馈上下行网络的带宽、丢包、网络抖动和往返时延数据。

LastmileProbeResult对象参数

参数

参数	类型	描述
state	short	网络探测状态, 参考RtcLastmileState。
rtt	int	网络探测往返平均延时, 单位(ms)。
quality	int	探测网络质量, 参考RtcLastmileQuality。
uplinkReport	LastmileProbeOneWayResult	上行网络探测质量。
downlinkReport	LastmileProbeOneWayResult	下行网络探测质量。

LastmileProbeOneWayResult对象参数

参数

参数	类型	描述
lossRate	int	探测的网络丢包率。
jitter	int	探测的网络抖动，单位(ms)。
bandwidth	int	探测的网络带宽，单位(Kbps)。

RtcLastmileState 状态定义

状态	值	含义
PROBE_RESULT_COMPLETE	1	本次质量探测是完整的
PROBE_RESULT_INCOMPLETE	2	本次质量探测未进行带宽预测，因此结果不完整
PROBE_RESULT_UNAVAILABLE	3	未进行质量探测。一个可能的原因是网络连接中断

RtcLastmileQuality 质量定义

状态	值	含义
QUALITY_UNKNOWN	0	质量未知
QUALITY_EXCELLENT	1	质量极好
QUALITY_GOOD	2	用户主观感觉和极好差不多，但码率可能略低于极好
QUALITY_POOR	3	用户主观感受有瑕疵但不影响沟通
QUALITY_BAD	4	勉强能沟通但不顺畅
QUALITY_VBAD	5	网络质量非常差，基本不能沟通
QUALITY_DOWN	6	完全无法沟通
QUALITY_DETECTING	7	SDK 正在探测网络质量

iOS SDK

概述

概述

本文档主要介绍如何将 RTC iOS SDK 集成到您的项目中，并介绍各类功能的使用方法。在使用本文档前，您需要先了解RTC的一些基本知识，并已经开通了RTC服务。若您还不了解RTC，可以参考产品描述。

RTC iOS SDK 能够帮助您实现登录音视频通信房间并开始通信，并可以对发布/订阅、音视频参数、设备参数进行设置。 **接口概览** RTC iOS SDK 提供以下接口 **初始化接口** |API|描述| |-|-| |initSDKWithAppID|初始化SDK| |setBaiduRtcRoomDelegate|设置RTC回调代理| |setParamSettings|设置媒体参数| |getParamSettings|获取媒体参数| |setEngineStateStatistics|开启/关闭引擎统计信息|

房间相关接口 |API|描述| |-|-| |loginRtcRoomWithRoomName|登录房间| |loginRtcRoomWithRoomName：isCompulsive：|强制登录房间| |logoutRtcRoom|登出房间| |startRoomMediaRelay|启动跨房间通信| |stopRoomMediaRelay|停止跨房间通信| |stopRoomMediaRelayAll|停止所有跨房间通信| |kickOffUserWithId|踢出房间内某成员| |shutUpUserWithId|禁言房间内某成员| |disbandRoom|解散房间| |queryUserListOfRoom|获取房间成员ID列表| |queryMessageUserListOfRoom|获取房间成员详细列表| **发布/订阅流相关接口** |API|描述| |-|-| |startLiveServerStreaming|开始直播推流| |stopLiveServerStreaming|停止直播推流| |publishStreaming|发布媒体流| |stopPublishStreaming|停止发布媒体流| |subsribeStreaming|订阅某成员媒体流| |stopSubscribeStreaming|停止订阅某成员媒体流|

消息相关接口 |API|描述| |-|-| |sendMessage|广播消息（数据通道）| |sendMessage2|广播消息（信令通道）| |sendMessage2WithUserId|向某成员发送消息| |setUserAttribute|设置用户属性| |getUserAttribute|获取用户属性|

视频相关接口 |API|描述| |-|-| |setVideoCaptureFactory|设置本地视频采集器| |setRenderDelegate|设置外部渲染回调对象| |startPreview|开启本地预览| |stopPreview|停止本地预览| |setLocalDisplay|设置本地渲染窗口| |setRemoteDisplay|设置远端渲染窗口| |setRemoteDisplay: userId:|设置指定远端用户渲染窗口 | |updateDisplay: userId:|更新远端用户渲染窗口| |setRemoteVideoPlayState|设置是否拉取某成员视频流|

摄像头相关接口 |API|描述| |-|-| |switchCamera|切换摄像头| |muteCamera|关闭摄像头| |muteCamera:|关闭/开启本地视频采集| |setCameraFace|设置前后摄像头| |cameraFocusWithPoint|设置摄像头对焦|

音频相关接口 |API|描述| |-|-| |setAudioRecordDelegate|设置音频录制回调对象| |setAudioSessionDelegate|设置音频会话代理对象| |setAudioExternalDeviceDelegate|设置音频外部采集代理对象| |muteMicphone|关闭音频采集| |enableLocalAudio:|关闭/开启本地音频采集| |muteSpeaker:|关闭/开启扬声器| |muteMicphone:|关闭/开启音频采集| |switchLoundSpeaker|切换扬声器/听筒| |switchAudioCategaryWithSpeaker:|切换扬声器/听筒| |presetLoudSpeaker|预置听筒/扬声器| |getRemoteAudioLevels|获取房间成员语音激励列表| |setAudioSessionMode|设置音频会话模式| |enableAgc|开启默认音频自动增益| |enableAns|开启默认音频自动噪声抑制| |setSoundMode|设置音频输出设备| |setRemoteAudioPlayState|设置是否拉取某成员音频流| |setRemoteAudioPlayVolume|设置远端用户音频播放音量|

转推配置相关接口 |API|描述| |-|-| |configLiveServerWithUrl|配置媒体转推参数|

通知相关接口 |API|描述| |-|-| |onLoinSuccess|登录成功回调| |onError|错误信息更新回调| |onConnectionStateChanged|链接状态变更| |onEngineStatisticsInfo|开启/关闭引擎统计信息| |onTextMessageArrival|消息通知（信令通道）| |onTextMessageArrival2|消息通知（数据通道）| |onTextMessageAttribute|接收到获取用户属性回调|

其他接口 |API|描述| |-|-| |version|获取SDK版本号| |setVerbose|开启/关闭调试信息输出| |queryEngineStatisticsInfo|查询RTC统计信息| |enableStatsToServer|开启/关闭RTC质量监控数据上报| |enableErrorInfoReprot|开启/关闭RTC异常信息上报|

API

初始化接口

SDK初始化

```
 - (instancetype)initSDKWithAppID:(NSString *)appId tokenStr:(NSString *)tokenStr delegate:
(id<BaiduRtcRoomDelegate>)delegate;
```

初始化SDK。

初始化SDK时调用。设置 AppID、TokenStr 等。

参数

参数	类型	描述
appId	NSString*	RTC 基础业务单元的唯一标识
tokenStr	NSString*	RTC Server 端鉴权使用的字符串
delegate	id<BaiduRtcRoomDelegate>	遵循 BaiduRtcRoomDelegate 协议的对象

返回

返回 SDK 实例，nil 表示初始化失败

设置媒体参数

```
- (void)setParamSettings:(RtcParameterSettings *)paramSettings paramType:(RtcParamSettingType)paramType;
```

音视频相关参数设置。

该函数在 loginRtcRoomWithRoomName 前调用，用于设置音视频采集，编解码相关的参数。

参数

参数	类型	描述
paramSettings	RtcParameterSettings*	设置参数包括分辨率、帧率、码率、视频方向等
paramType	RtcParamSettingType	参数类型，可指定设置某一项，或设置所有参数

RtcParameterSettings 定义

```
 // Audio
@property (nonatomic, assign) BOOL hasAudio;    // 是否采集音频数据
@property (nonatomic, assign) BOOL hasRemoteAudio;  // 是否自动订阅远端音频流，默认 YES
@property (nonatomic, assign) BOOL isCreatingAecDump; // 是否开启aecdump
@property (nonatomic, assign) BOOL isUsingLevelControl; // 是否开启等级控制
@property (nonatomic, assign) BOOL isUsingManualConfig; // 是否开启手动配置
@property (nonatomic, assign) BOOL isExportAudioRecord; // 是否开启外部采集
@property (nonatomic, assign) BOOL isExportAudioPlayout; // 是否开启外部播放
@property (nonatomic, assign) BOOL isExportAudioRecordPlayoutMix; // 是否开启外部混音
@property (nonatomic, assign) BOOL isEnableExternalAudioDevice;//是否外部音频设备
@property (nonatomic, assign) BOOL disableAudioMixWithOthers;//是否混音
@property (nonatomic, assign) int audioSampleRate;//音频帧率
@property (nonatomic, assign) int audioBitrate; // 音频码率 kbps
@property (nonatomic, assign) RtcAudioBitrateMode audioBitrateMode;//音频码率模式
@property (nonatomic, assign) RtcTransportAudioChannel audioChannelTransport;//音频转推模式
@property (nonatomic, assign) BOOL isSoftNsEnable;//是否噪音抑制
@property (nonatomic, assign) int audioJitterBufferSize;//音频抖动缓冲空间
@property (nonatomic, assign) int playoutDelayAudioOnly;//音频播放延迟
@property(nonatomic, assign) int audioCodeComplex; //音频编码复杂度 value [0 - 10]
@property (nonatomic, assign) BOOL audioLevelEnable;//是否开启音频激励
@property (nonatomic, assign) int audioLevelTopCount;//音频层级最大数
@property (nonatomic, assign) RtcNetworkDetectionMode networkDetectionMode; // 带宽探测模式（用于POC环境/实验室环境测试）
//Video
@property (nonatomic, assign) BOOL hasVideo; // 是否开启视频采集
@property (nonatomic, assign) BOOL hasRemoteVideo;  // 是否自动订阅远端视频流，默认 YES

@property (nonatomic, assign) BOOL isEnableExternalRender; // 是否开启外部渲染
@property (nonatomic, assign) BOOL isEnableHighProfile; // 开启highProfile
@property (nonatomic, assign) BOOL isEnableVBR;        // 开启VBR模式保障编码清晰度更高
@property (nonatomic, assign) BOOL isEnableFixedResolution; // 开启固定分辨率
@property (nonatomic, assign) BOOL isEnableVideoDump;//是否开启视频dump
@property (nonatomic, assign) BOOL isVideoFramingEnhance;//是否视频增强
@property (nonatomic, assign) BOOL isEnableJitterRetransmission;//是否发送抖动信息
@property (nonatomic, assign) RtcVideoBgPushMode videoBgPushMode; // 后台推流模式
@property (nonatomic, assign) BOOL enableAutoReconnect;  // 网络重连接
@property (nonatomic, assign) BOOL isIgnoreUnknownSEI;  // 忽略未知SEI消息

/// 上行视频降级策略，默认 MAINTAIN_FRAMERATE
/// @discussion 注意当该配置为 MAINTAIN_FRAMERATE 或 BALANCED 时，若同时开启了 isEnableFixedResolution，分辨率动态调整策略会失效
@property (nonatomic, assign) RtcVideoDegradationPreference degradationPreference;

/// 采集视频参数设置
@property (nonatomic, copy) RtcVideoBaseParams *videoCaptureParams;
```

```
    /// 为不同媒体类型流配置不同编码参数，不同 RtcMediaTarget 参考 BaiduRtcRoomApiDefine 中的定义
    @property (nonatomic, copy) NSDictionary<RtcMediaTarget, RtcVideoEncodeParams *> *videoEncodeParams;

    @property (nonatomic, assign) BOOL hasData;  // 是否开启DataChannel

    //发布订阅流
    @property (nonatomic) BOOL isAutoPublish; // 是否自动发布流
    @property (nonatomic) BOOL isAutoSubscribe; // 是否自动订阅流
    @property (nonatomic, assign) BOOL enableMultiStream;//是否多流模式， 默认YES
    @property (nonatomic, assign) BOOL enableBeauty; // 是否启用内部美颜
    @property (nonatomic, assign) BOOL enableAudioProcess; // 是否启用内部音频处理
    @property (nonatomic, assign) LiveServerInfo *roomTransInfo; // 房间转推配置
    @property (nonatomic, assign) LiveServerInfo *anchorTransInfo; // 主播转推配置
    @property (nonatomic, assign) BOOL isMixing; // 是否混流
    @property (nonatomic, copy) NSString *rtmpUrl;  // 转推地址
    @property (nonatomic, assign) RtcLiveTransferMode liveTransferMode; // 转推模式
    // signal channel
    @property (nonatomic, assign) RtcSignalChannelMode signalChannelMode;//信令模式
    @property (nonatomic, copy) NSString *kcpServerDomain;//信令地址
    @property (nonatomic, copy) NSString *kcpSignalServerDomain;//信令地址
    @property (nonatomic, assign) RtcSignalChannelMode signalChannelMode;      // 信令模式 0: 传统TCP
RTC_SIGNAL_CHANNEL_MODE_TCP  1: quic协议 RTC_SIGNAL_CHANNEL_MODE_QUIC
    @property (nonatomic, assign) int subscribeCount;    // 精简信令模式预创建最多订阅人数
    @property (nonatomic, assign) NSSubscribeMode subscribeMode; // 精简信令模式 0：kSubscribeModeManual 手动订
阅，1：kSubscribeModeAuto 自动订阅  2：kSubscribeModeMeeting 会议模式 音频自动订阅，视频手动订阅
```

RtcVideoBaseParams 定义

```
  @property (nonatomic, assign) int videoFps;
@property (nonatomic, assign) int videoWidth;
@property (nonatomic, assign) int videoHeight;
```

RtcVideoEncodeParams 定义

```
  @interface RtcVideoEncodeParams : RtcVideoBaseParams<NSCopying>
@property (nonatomic, assign) int gopSize; // unit: frames
@property (nonatomic, assign) int videoBitrate; // unit: kbps
@property (nonatomic, assign) int videoMinBitrate;  // unit: kbps
@property (nonatomic, assign) RtcVideoEncodeType videoCodecType;
@end


typedef NS_ENUM(NSInteger, RtcVideoEncodeType) {
    RTC_VIDEO_ENCODE_TYPE_H264 = 0,
    RTC_VIDEO_ENCODE_TYPE_VP8,
    RTC_VIDEO_ENCODE_TYPE_VP9,
    RTC_VIDEO_ENCODE_TYPE_H263,
    RTC_VIDEO_ENCODE_TYPE_HEVC,
    RTC_VIDEO_ENCODE_TYPE_JPEG,
    RTC_VIDEO_ENCODE_TYPE_OTHER
};
```

RtcVideoEncodeType 定义

```
typedef NS_ENUM(NSInteger, RtcVideoEncodeType) {
    RTC_VIDEO_ENCODE_TYPE_H264 = 0,
    RTC_VIDEO_ENCODE_TYPE_VP8,
    RTC_VIDEO_ENCODE_TYPE_VP9,
    RTC_VIDEO_ENCODE_TYPE_H263,
    RTC_VIDEO_ENCODE_TYPE_HEVC,
    RTC_VIDEO_ENCODE_TYPE_JPEG,
    RTC_VIDEO_ENCODE_TYPE_OTHER
};
```

RtcAudioEncodeType 定义

```
typedef NS_ENUM(NSInteger, RtcAudioEncodeType) {
    RTC_AUDIO_ENCODE_TYPE_OPUS = 0,  // 默认opus编码器
    RTC_AUDIO_ENCODE_TYPE_AAC,
    RTC_AUDIO_ENCODE_TYPE_EAAC,
    RTC_AUDIO_ENCODE_TYPE_SPEEX,
    RTC_AUDIO_ENCODE_TYPE_AMRWB,
    RTC_AUDIO_ENCODE_TYPE_PCMU,  // PCMU 对比 OPUS CPU占用更小
    RTC_AUDIO_ENCODE_TYPE_PCMA,
    RTC_AUDIO_ENCODE_TYPE_G722,
    RTC_AUDIO_ENCODE_TYPE_OTHER
};
```

返回

无

获取媒体参数

```
- (RtcParameterSettings *)getParamSettings;
```

获取音视频相关参数。

获取当前设置的音视频相关参数，如分辨率、帧率、码率、视频方向等。

参数

无

返回

RtcParameterSettings 音视频参数信息

开启/关闭引擎统计信息

```
- (void)setEngineStateStatistics:(BOOL)bOnStatistics;
```

RTC统计信息开关。

当打开开关时 onEngineStatisticsInfo 函数每隔1秒返回一次统计信息，并且可通过queryEngineStatisticsInfo 函数主动查询RTC统计信息。统计信息包括 CPU, FPS，video codec等。

参数

参数	类型	描述
bOnStatistics	BOOL	bOnStatistics = true, 打开开关；bOnStatistics = false, 关闭引擎统计信息

返回

无

房间相关接口

登录房间（1/2）

```
- (BOOL)loginRtcRoomWithRoomName:(NSString *)roomName
                          userID:(NSInteger)userId
                     displayName:(NSString *)displayName;
```

登录房间。

登录房间成功，在同一个房间的人能进行相互音视频聊天，如果失败，会通过onErrorInfoUpdate call back 返回错误信息。

参数

参数	类型	描述
roomName	NSString*	房间名，长度不可超过 255 byte
userId	NSInteger	用户 id,每个房间的用户 ID 必须唯一
displayName	NSString*	用户昵称

返回

true 成功，false 失败

登录房间（2/2）

```
 /** 登录参数 */
@interface LoginOptions : NSObject
@property (nonatomic, assign) BOOL isCompulsive;
@property (nonatomic, assign) BRTCRoleType roleType;
@property (nonatomic, strong) NSString *tokenStr;
@end

 /**
 * @param roomName 房间名，长度不可超过 255 byte
 * @param userId 用户id,每个房间的用户ID必须唯一
 * @param displayName 用户昵称
 * @param options 其他登录参数
 **/
   - (BOOL)loginRtcRoomWithRoomName:(NSString *)roomName
                          userID:(NSInteger)userId
                     displayName:(NSString *)displayName
                         options:(LoginOptions *)options;
```

options.isCompulsive = YES; 强制登录房间。

登录房间成功，在同一个房间的人能进行相互音视频聊天，如果失败，会通过onErrorInfoUpdate call back 返回错误信息。此接口会踢出当前房间内同一userID用户，建议在断网重连或者初次登录失败时调用。

参数

参数	类型	描述
roomName	NSString*	房间名，长度不可超过 255 byte
userId	NSInteger	用户 id,每个房间的用户 ID 必须唯一
displayName	NSString*	用户昵称
options.isCompulsive	BOOL	是否开启强制登录

返回

true 成功，false 失败

登出房间

```
- (BOOL)logoutRtcRoom;
```

退出房间。

执行 logoutRtcRoom 后，会停止音视频采集，断开与房间服务器的连接，取消音视频的传输，销毁音视频传输通道以及释放其他资源。

参数

无

返回

true 成功，false 失败

启动跨房间通信

```
- (void)startRoomMediaRelay:(NSString *)destRoomName
                     userId:(NSInteger)userId
                      token:(NSString *)token;
```

启动跨房间通信。

参数

参数	类型	描述
destRoomName	NSString*	目标房间
userId	NSInteger	加入目标房间时使用的userId，必须在加入房间中是唯一的
token	NSString*	派发的token字符串

返回

无

停止跨房间通信

```
- (void)stopRoomMediaRelay:(NSString *)destRoomName
                    userId:(NSInteger)userId;
```

停止跨房间通信。

参数

参数	类型	描述
destRoomName	NSString*	目标房间
userId	NSInteger	加入目标房间时使用的userId，必须在加入房间中是唯一的

返回

无

停止所有跨房间通信

```
- (void)stopRoomMediaRelayAll;
```

停止所有跨房间通信。

停止所有启动的跨房间通信。

参数

无

返回

无

踢出某成员

```
- (void)kickOffUserWithId:(NSInteger)userId;
```

踢除聊天。

房管／主播／会议主持 把某人踢出聊天室。

参数

参数	类型	描述
userId	NSInteger	在房间中的用户的 用户 ID

返回

无

禁言某成员

```
- (void)shutUpUserWithId:(NSInteger)userId isDisable:(BOOL)isDisable;
```

禁言。

房管/主播／会议主持 禁止某人发言。

参数

参数	类型	描述
userId	NSInteger	在房间中的用户的 用户 ID
isDisable	BOOL	true:禁言 false:取消禁言

返回

无

解散房间

```
- (void)disbandRoom;
```

解散房间。

房间管理员有权利解散整个房间，解散后，房间中的每个人都退出房间。

参数

无

返回

无

切换角色

```
- (void)switchRole:(BRTCRoleType)roleType;
```

切换角色。

参数 type

BRTC_ROLE_ANCHOR 主播角色 支持推流和拉流 BRTC_ROLE_AUDIENCE 观众角色，仅仅支持拉流

返回 无

获取房间成员ID列表

```
- (NSArray *)queryUserListOfRoom;
```

查询媒体用户。

查询房间用户信息，获取房间中所有媒体用户列表。

参数

无

返回

NSArray* 用户信息列表

获取房间成员详细列表

```
- (NSArray *)queryMessageUserListOfRoom;
```

查询消息用户。

查询房间用户信息，获取房间中 所有用户列表及用户流状态信息，是否可订阅，是否禁言，禁画中

参数

无

返回

NSArray* 用户信息列表

发布/订阅流相关接口 开始直播推流

```
- (void)startLiveServerStreaming:(NSString *)url
                          isMix:(BOOL)isMix
                    isRecording:(BOOL)isRecording
                     mixTemplate:(NSString *)mixTmplt
                   transferMode:(RtcLiveTransferMode)mode;
```

动态配置server端推流参数。

该接口用于动态配置server推流的参数。configLiveServerWithUrl在执行loginRtcRoom之前调用。本接口在loginRtcRoom成功后调用。两个接口不要混用。

参数

参数	类型	描述
url	NSString*	rtmp推流地址
isMix	BOOL	是否做混流处理
isRecording	BOOL	是否录制
mixTmplt	NSString*	混流模版
mode	RtcLiveTransferMode	转推模式，有两种： 聊天室模式， 主播转推模式

返回

无 **停止直播推流**

```
- (void)stopLiveServerStreaming:(RtcLiveTransferMode)transferMode;
```

停止server推流。

若需要通话过程中，停止转推，调用该接口。

参数

参数	类型	描述
transferMode	RtcLiveTransferMode	转推模式，有两种： 聊天室模式， 主播转推模式

返回

无 **发布媒体流**

```
- (void)publishStreaming;
```

发布流。

流发布在 roomId 指定的房间，在同一房间 joined 的用户可以相互订阅流，默认在发布流的同时， listening/subscriber 在该房间其他用户的流。

参数

无

返回

无

停止发布媒体流

```
- (void)stopPublishStreaming;
```

停止发布流。

stop 通过 publishStreaming 发布的流。

参数

无

返回

无

订阅某成员媒体流

```
- (void)subsribeStreaming:(NSArray<NSNumber *> *)streamingIds;;
```

订阅流。

用于订阅同一房间的其他用户的流.

参数

参数	类型	描述
streamingIds	NSArray<NSNumber*>*	用户要订阅的其他用户的流id列表(即其他用户id列表)

返回

无

订阅某成员音频流

```
- (void)subscribeAudioStreaming:(NSNumber *)feedId;
```

订阅指定远端用户音频流。

参数

参数	类型	描述
feedId	NSNumber *	指定远端用户ID

返回

无

订阅某成员视频流

```
- (void)subscribeVideoStreaming:(NSNumber *)feedId;
```

订阅指定远端用户视频流。

参数

参数	类型	描述
feedId	NSNumber *	指定远端用户ID

返回

无

订阅所有远端用户音频流

```
- (void)subscribeAllRemoteAudioStreams;
```

订阅同一房间所有远端用户音频流

参数

无

返回

无

订阅所有远端用户视频流

```
- (void)subscribeAllRemoteVideoStreams;
```

订阅同一房间所有远端用户视频流

参数

无

返回

无

停止订阅某成员媒体流

```
- (void)stopSubscribeStreaming:(NSArray<NSNumber *> *)streamingIds;;
```

停止订阅流。

该接口停止已经订阅的流.

参数

参数	类型	描述
streamingIds	NSArray<NSNumber*>*	用户要停止订阅的流id列表(即其他用户id列表)

返回

无

停止订阅某成员音频流

```
- (void)stopSubscribeAudioStreaming:(NSNumber *)feedId;
```

停止订阅指定远端用户音频流。

参数

参数	类型	描述
feedId	NSNumber *	指定远端用户ID

返回

无

停止订阅某成员视频流

```
- (void)stopSubscribeVideoStreaming:(NSNumber *)feedId;
```

停止订阅指定远端用户视频流。

参数

参数	类型	描述
feedId	NSNumber *	指定远端用户ID

返回

无

停止订阅所有远端用户音频流

```
- (void)stopSubscribeAllRemoteAudioStreams;
```

停止订阅同一房间所有远端用户音频流

参数

无

返回

无

停止订阅所有远端用户视频流

```
- (void)stopSubscribeAllRemoteVideoStreams;
```

停止订阅同一房间所有远端用户视频流

参数

无

返回

无

消息相关接口

广播消息（数据通道）

```
- (int)sendMessage:(NSString *)message;
```

广播消息。

当enable数据通道后，可通过该接口发送文本消息给在同一房间的其他用户。

参数

参数	类型	描述
message	NSString*	需要发送的文本消息

返回

-1 发送失败，0 发送成功 **广播消息（信令通道）**

```
- (int)sendMessage2:(NSString *)message;
```

广播消息。

通过信令信道 该接口发送文本消息给在同一房间的其他用户。

参数

参数	类型	描述
message	NSString*	需要发送的文本消息

返回

-1 发送失败，0 发送成功

发送SEI消息

```
- (void)sendSEIMsg:(NSData *)message repeatCount:(int)repeatCount;
```

message 二进制消息 消息长度不超过1K。

repeatCount 重复次数

发送SEI消息，房间中其他用户通过 onRecvSEIMsg 回调收到消息。

参数

参数	类型	描述
message	NSData*	二进制消息
repeatCount	int	重复次数

向某成员发送消息

```
- (int)sendMessage2WithUserId:(NSString *)message userId:(NSNumber *)userId;
```

指定用户发送消息。

发送文本消息给在同一房间的指定用户。

参数

参数	类型	描述
message	NSString*	需要发送的文本消息
userId	NSNumber*	指定接收消息的用户 ID

返回

-1 发送失败，0 发送成功

设置用户属性

```
- (int)setUserAttribute:(NSString *)attribute;
```

attribute 属性设置。

设置本用户 attribute 属性，房间其他用户由 onTextMessageAttribute 通知收到。

参数

参数	类型	描述
attribute	NSString*	attribute 值

返回

-1 设置失败，0 设置成功

获取用户属性

```
- (int)getUserAttribute:(NSNumber *)userID;
```

attribute属性获取。

获取指定用户 attribute 属性，属性值由 onTextMessageAttribute 通知得到。

参数

参数	类型	描述
userID	NSNumber*	用户ID

返回

-1 获取失败，0 获取成功

视频相关接口

设置自定义视频采集器

```
- (void)setVideoCapturer:(id<BaiduVideoCapturer>)capturer;
```

设置自定义主视频源。主视频流一般为相机流，当接入方通过调用该方法设置自定义源，内部相机采集将不会启动

必须在登录房间前调用，并且不能置空。

参数

参数	类型	描述
capturer	id<BaiduVideoCapturer>	遵循 BaiduVideoCapturer 协议的对象

返回

无

设置本地视频采集器

> v2.3 开始已废弃, 请使用 setVideoCaoturer: 自定义视频采集器接口

```
- (void)setVideoCaptureFactory:(id<BaiduVideoCaptureFactory>)factory BRTC_DEPRECATED("Use setVideoCapturer: instead.");
```

外部采集设置。若开启视频外部采集，则sdk内部不打开拍摄头采集视频。

设置外部采集模块。必须在登录房间前调用，并且不能置空。

参数

参数	类型	描述
factory	id<BaiduVideoCaptureFactory>	工厂对象，遵循 BaiduVideoCaptureFactory 协议的对象

返回

无

设置外部渲染回调对象

```
- (void)setRenderDelegate:(id<BaiduRtcApiRenderDelegate>)renderDelegate;
```

外部渲染设置。

设置外部渲染回调对象。使用外部渲染功能，需要设置代理对象。未设置代理对象，或对象设置错误，可能导致无法正常收到相关回调

> 注意：请使用 BaiduExternalVideoRender

参数

参数	类型	描述
renderDelegate	id<BaiduRtcApiRenderDelegate>	遵循 BaiduRtcApiRenderDelegate 协议的代理对象

返回

无

开启本地预览

```
- (void)startPreview;
```

本地预览。

打开camera，开始预览。

参数

无

返回

无

停止本地预览

```
- (void)stopPreview;
```

停止预览。

关闭camera, 停止本地预览。

参数

无

返回

无

设置本地渲染窗口

```
- (void)setLocalDisplay:(RTCLocalVideoView *)localVideoView;
```

本地显示view设置。

设置本地显示view。在 loginRtcRoomWithRoomName 之前调用，loginRtcRoomWithRoomName 之后，本地视频数据会显示到 localVideoView, localVideoView 的位置大小要与采集的视频大小成比例。

参数

参数	类型	描述
localVideoView	RTCLocalVideoView*	本地显示view,用于显示camera采集的视频数据

返回

无

设置本地指定类型流渲染窗口

```
- (void)setLocalDisplay:(RTCLocalVideoView *)localVideoView mediaTarget:(RtcMediaTarget)mediaTarget;
```

设置本地关联指定媒体目标类型的显示view。

设置本地显示view。在 loginRtcRoomWithRoomName 之前调用，loginRtcRoomWithRoomName 之后，本地视频数据会显示到 localVideoView, localVideoView 的位置大小要与采集的视频大小成比例。

参数

参数	类型	描述
localVideoView	RTCLocalVideoView*	本地显示view,用于显示camera采集的视频数据
mediaTarget	RtcMediaTarget	视图需要绑定到的媒体目标，具体信息参考 BaiduRtcRoomApiDefines.h

返回

无

设置指定远端用户渲染窗口

```
- (void)setRemoteDisplay:(RTCRemoteVideoView *)remoteVideoView userId:(NSInteger)userId;
```

多人模式设置指定用户远端 view。

在 loginRtcRoomWithRoomName 之后调用，远端视频数据会显示到指定view

参数

参数	类型	描述
remoteVideoView	RTCRemoteVideoView*	远端画面显示 view, 用于显示远端用户传输过来的视频数据
userId	NSInteger	远端用户ID

返回

无

设置指定远端用户指定类型流渲染窗口

```
- (void)setRemoteDisplay:(RTCRemoteVideoView *)remoteVideoView userId:(NSInteger)userId mediaTarget:
(RtcMediaTarget)mediaTarget;
```

两人通话模式，设置远端指定用户关联指定媒体目标类型的显示 view。

设置远端显示view。在 loginRtcRoomWithRoomName 之前调用，loginRtcRoomWithRoomName 之后，远端视频数据会显示到 remoteVideoView，remoteVideoView 的大小要与采集的视频大小成比例。

参数

参数	类型	描述
remoteVideoView	RTCRemoteVideoView*	远端画面显示view, 用于显示远端用户传输过来的视频数据
userId	NSInteger	远端用户ID
mediaTarget	RtcMediaTarget	视图需要绑定到的媒体目标，具体信息参考 BaiduRtcRoomApiDefines.h

返回

无

更新远端用户渲染窗口

```
- (void)updateDisplay:(RTCRemoteVideoView *)remoteVideoView userId:(NSInteger)userId;
```

更新指定用户的远端显示view。

在 loginRtcRoomWithRoomName 之后调用，远端视频数据会显示到指定view

参数

参数	类型	描述
remoteVideoView	RTCRemoteVideoView*	远端画面显示 view, 用于显示远端用户传输过来的视频数据
userId	NSInteger	远端用户ID

返回

无

设置是否拉取某成员视频流

```
-(void)setRemoteVideoPlayState:(BOOL)stats userId:(NSInteger)userId;
```

指定远端画面暂停/恢复播放。

通过控制远端视频流是否拉取，指定远端画面暂停/恢复播放。

参数

参数	类型	描述
stats	BOOL	true：拉取；false：停止拉取 默认拉取
userId	NSInteger	远端用户ID

返回

无

设置本地视频后台推流模式

```
 typedef NS_ENUM(NSInteger, RtcVideoBgPushMode) {
    BRTC_VIDEO_BG_PUSH_NOT = 0,
    BRTC_VIDEO_BG_PUSH_BLACKFRAME,
    BRTC_VIDEO_BG_PUSH_LASTFRAME
};

typedef NS_ENUM(NSInteger, RtcParamSettingType) {
    RTC_PARAM_SETTINGS_VIDEO_BG_PUSH_MODE
};
```

通过 setParamSettings:paramType: 接口设置 videoBgPushMode 选择后台推流模式。需要在登录房间之前设置，默认为不后台推流

视频画面截图

```
- (void)takeVideoSnapshot:(NSInteger)userId
              mediaTarget:(RtcMediaTarget)mediaTarget
          completionBlock:(void (^)(UIImage *image, RtcErrorCodes errCode))completionBlock;
```

对本地或远端用户指定类型的的视频流进行截图，通过回调返回截图结果

注：对本地用户仅支持默认视频流。

参数

参数	类型	描述
userId	NSInteger	视频流用户 ID，0 表示截取本地视频画面
mediaTarget	RtcMediaTarget	流类型，如果传入 nil 则指定为默认视频流
completionBlock	block	截图完成回调

回调

completionBlock

参数	类型	描述
image	UIImage *	截图结果图像
errCode	RtcErrorCodes	0: 截图成功; 601: 视频视图不存在; 602: 视频视图未实现截图能力; 603: 视频视图截图结果为空;

设置视频上行降级策略

```
typedef NSString *RtcVideoDegradationPreference NS_TYPED_ENUM;

extern RtcVideoDegradationPreference const RTC_VIDEO_DEGRADATION_PREFERENCE_MAINTAIN_FRAMERATE;
extern RtcVideoDegradationPreference const RTC_VIDEO_DEGRADATION_PREFERENCE_MAINTAIN_RESOLUTION;
extern RtcVideoDegradationPreference const RTC_VIDEO_DEGRADATION_PREFERENCE_BALANCED;
```

通过 setParamSettings:paramType: 接口设置 degradationPreference 选择视频上行降级策略。需要在登录房间之前设置，默认为 MAINTAIN_FRAMERATE

注：当该配置为 MAINTAIN_FRAMERATE 或 BALANCED 时，若同时开启了 isEnableFixedResolution，分辨率动态调整策略会失效

摄像头相关接口

切换摄像头

```
- (void)switchCamera;
```

摄像头切换。

切换摄像头，前后摄像头切换。

参数

无

返回

无

关闭/开启本地视频采集

```
- (void)muteCamera:(BOOL)isMute;
```

关闭/打开摄像头。

在关闭摄像头后，不传输本地视频数据给对方，默认打开。

参数

参数	类型	描述
isMute	BOOL	true：关闭视频数据传输；fasle：打开

返回

无

设置前后摄像头

```
-(void)setCameraFace:(BOOL)front;
```

设置前后摄像头。

设置前后摄像头。

参数

参数	类型	描述
front	BOOL	true:前置； false:后置

返回

无

设置摄像头对焦

```
- (void)cameraFocusWithPoint:(CGPoint)point andPlaneSize:(CGSize)size;
```

摄像头对焦。

摄像头对焦功能，支持手动和自动对焦。

参数

参数	类型	描述
point	CGPoint	对焦点坐标
size	CGSize	camera预览view尺寸

返回

无

音频相关接口

设置音频录制回调对象

```
- (void)setAudioRecordDelegate:(id<BaiduRtcApiAudioRecordDelegate>)audioRecordDelegate;
```

音频录制回调对象设置。

设置音频录制回调代理对象。开启音频录制功能，需要设置代理对象。未设置代理对象，或对象设置错误，可能导致无法正常收到相关回调。

参数

参数	类型	描述
audioRecordDelegate	id<BaiduRtcApiAudioRecordDelegate>	遵循 BaiduRtcApiAudioRecordDelegate 协议的代理对象

返回

无

设置音频会话代理对象

```
- (void)setAudioSessionDelegate:(id<BaiduRtcRoomApiAudioSessionDelegate>)audioSessionDelegate;
```

设置音频会话代理对象。

接收音频设备状态变化的通知，详情请看BaiduRtcRoomApiAuidoSession.h。

参数

参数	类型	描述
audioSessionDelegate	id<BaiduRtcRoomApiAudioSessionDelegate>	详细定义参考 BaiduRtcRoomApiAuidoSession.h

返回

无 **设置音频外部采集代理对象**

```
- (void)setAudioExternalDeviceDelegate:(id<BaiduRtcRoomApiAudioExternalDeviceDelegate>)audioExternalDelegate;
```

设置音频外部采集代理对象。

接收音频外部采集的回调功能。

参数

参数	类型	描述
audioExternalDelegate	id<BaiduRtcRoomApiAudioExternalDeviceDelegate>	详细定义参考 BaiduRtcRoomApiAudioExternalDevice.h

返回

无

关闭音频采集

```
- (void)muteMicphone;
```

关闭音频采集。

关闭音频采集，停止音频的采集

参数

无

返回

无 **关闭/开启本地音频采集**

```
- (int)enableLocalAudio:(BOOL)enabled;
```

关闭/开启本地音频采集。

关闭音频采集，停止音频的采集。

参数

参数	类型	描述
enabled	BOOL	true：开启本地音频采集；false：关闭本地音频采集

返回

-1 关闭/开启本地音频采集失败，0 关闭/开启本地音频采集成功 **关闭/开启扬声器**

```
- (void)muteSpeaker:(BOOL) isMute;
```

关闭/开启扬声器。

关闭扬声器，停止音频播放。

参数

参数	类型	描述
isMute	BOOL	true：关闭扬声器；false：开启扬声器

返回

无 **关闭/打开音频采集**

```
- (void)muteMicphone:(BOOL)isMute;
```

关闭/打开音频采集。

关闭音频采集，停止音频的采集，默认开启。

参数

参数	类型	描述
isMute	BOOL	true：停止音频发送；false：开启音频发送

返回

无

切换扬声器/听筒（1/2）

```
- (void)switchLoundSpeaker;
```

开关扬声器。

开关扬声器，听筒与扬声器切换(免提功能)。

参数

无

返回

无 **切换扬声器/听筒（2/2）**

```
- (void)switchAudioCategaryWithSpeaker:(BOOL)isSpeaker;
```

扬声器和听筒切换

参数

参数	类型	描述
isSpeaker	BOOL	true：开启扬声器；false：开启听筒模式

返回

无

预置听筒/扬声器

```
- (void)presetLoudSpeaker:(BOOL)isPresetLoudSpeaker;
```

预置听筒扬声器。

在 initSDKWithAppID 之后，loginRtcRoomWithRoomName 之前调用，预置听筒/扬声器播放语音。

参数

参数	类型	描述
isPresetLoudSpeaker	BOOL	true: 扬声器 false: 听筒 默认听筒

返回

无

获取扬声器状态

```
- (BOOL)isSpeakerOn;
```

检查音频路由是否是系统扬声器

参数

无

返回

BOOL 系统扬声器状态，YES 表示当前通话音频从系统扬声器输出，NO 表示从其他路由输出,如听筒、耳机、蓝牙设备等

获取房间成员语音激励列表

```
-(NSArray*)getRemoteAudioLevels;
```

语音激励接口。

获取用户语音激励列表。

参数

无

返回

NSArray <RtcRoomAudioLevel *> *

```
 /** 用户语音激励 */
@interface RtcRoomAudioLevel : NSObject
/** 用户 ID */
@property (nonatomic, copy) NSNumber *userID;
/** 用户名 */
@property (nonatomic, copy) NSString *userName;
/** 音量 */
@property (nonatomic, assign) NSInteger volumeLevel;
@end
```

启停用户音量提示

```
- (void)enableAudioVolumeIndication:(BOOL)enable interval:(int)interval;
```

启停用户音量提示，开启后按设置间隔时间，通过 onAudioVolumeIndication 方法回调

参数

参数	类型	描述
enable	BOOL	控制启用或停止音量提示
interval	int	提示回调间隔，单位毫秒。若启用，不能低于 100ms，建议 300ms

返回

无

设置音频会话模式

```
- (void)setAudioSessionMode:(AVAudioSessionMode)audioSessionMode;
```

设置音频会话模式。

参数

参数	类型	描述
audioSessionMode	AVAudioSessionMode	音频会话模式

返回

无 **开启默认音频自动增益**

```
- (void)enableAgc:(BOOL)enable;
```

自动增益开关。

是否开启音频自动增益。

参数

参数	类型	描述
enable	BOOL	true: 开启 false: 关闭 默认开启

返回

无

开启默认音频自动噪声抑制

```
- (void)enableAns:(BOOL)enable;
```

噪声抑制开关。

是否开启音频噪声抑制。

参数

参数	类型	描述
enable	BOOL	true: 开启 false: 关闭 默认开启

返回

无 **设置音频输出设备**

```
- (void)setSoundMode:(RtcSoundMode)mode;
```

设置声音输出方式。

设置声音输出方式，可选值：speaker（扬声器）或 ear（听筒)。

参数

参数	类型	描述
mode	RtcSoundMode	可选值：speaker（扬声器）或 ear（听筒)

返回

无

设置本地采集音频音量

```
- (void)setLocalAudioCaptureVolume:(float)volume;
```

设置本地采集音频音量, 范围[0, 4]。

参数

参数	类型	描述
volume	float	音量。范围[0, 4].

返回

无

设置是否拉取某成员音频流

```
-(void)setRemoteAudioPlayState:(BOOL)stats userId:(NSInteger)userId;
```

指定远端音频暂停/恢复播放。

通过控制是否拉取远端音频流，控制指定远端用户声音暂停/恢复播放。

参数

参数	类型	描述
stats	BOOL	true: 拉取 false: 停止拉取 默认拉取
userId	NSInteger	用户ID

返回

无 **设置远端用户音频播放音量**

```
-(void)setRemoteAudioPlayVolume:(float)volume userId:(NSInteger)userId;
```

指定用户音量。

指定用户音量. 音量设置只在本端有效，其他订阅者无影响。

参数

参数	类型	描述
volume	float	音量，范围[0,1]。
userId	NSInteger	用户ID

返回

无

设置所有远端用户混音后音量

```
- (void)setRemoteAudioPlayVolume:(float)volume;
```

设置所有远端用户混音后音量，范围[0, 4]。

参数

参数	类型	描述
volume	float	音量。范围[0, 4].

返回

无

转推配置相关接口

配置媒体转推参数

```
- (void)configLiveServerWithUrl::(NSString *)url
                          isMix:(BOOL)isMix
                    isRecording:(BOOL)isRecording
                    mixTemplate:(NSString *)mixTmplt
                   transferMode:(RtcLiveTransferMode)mode
                        avParam:(RtcLiveAuidoVideoParameters*)avParam;
```

该接口用于配置server推流的参数，聊天室模式：在同一个rtc房间的所有参与者在混流后，直接转推到一个指定的直播房间；主播转推模式：主播推向不同的直播房间。

参数

参数	类型	描述
url	NSString*	转推url地址
isMix	BOOL	是否混流
isRecording	BOOL	是否录制
mixTemplate	NSString*	混流模板
transferMode	RtcLiveTransferMode	转推模式
avParam	RtcLiveAuidoVideoParameters*	转推参数

混流模板名称	说明
side_by_side_primary_360p_4_3	主次平铺, 分辨率 480x360 (固定码率、帧率)
side_by_side_primary_360p_4_3-xx-xx	主次平铺, 分辨率 480x360 (自定义码率、帧率) 下面模板也可以设置
side_by_side_primary_360p_16_9	主次平铺, 分辨率640x360
side_by_side_primary_480p_4_3	主次平铺, 分辨率640x480
side_by_side_primary_480p_16_9	主次平铺, 分辨率854x480
side_by_side_primary_540p_16_9	主次平铺, 分辨率960x540
side_by_side_primary_720p_4_3	主次平铺, 分辨率960x720
side_by_side_primary_720p_16_9	主次平铺, 分辨率1280x720
side_by_side_equal_360p_4_3	平铺模式(大小相等), 分辨率480x360
side_by_side_equal_360p_16_9	平铺模式(大小相等), 分辨率640x360
side_by_side_equal_480p_4_3	平铺模式(大小相等), 分辨率640x480
side_by_side_equal_480p_16_9	平铺模式(大小相等), 分辨率854x480
side_by_side_equal_540p_16_9	平铺模式(大小相等), 分辨率960x540
side_by_side_equal_720p_4_3	平铺模式(大小相等), 分辨率960x720
side_by_side_equal_720p_16_9	平铺模式(大小相等), 分辨率1280x720
picture_in_picture_bottom_360p_4_3	画中画模式, 分辨率480x360
picture_in_picture_bottom_360p_16_9	画中画模式, 分辨率640x360
picture_in_picture_bottom_480p_4_3	画中画模式, 分辨率640x480
picture_in_picture_bottom_480p_16_9	画中画模式, 分辨率854x480
picture_in_picture_bottom_540p_16_9	画中画模式, 分辨率960x540
picture_in_picture_bottom_720p_4_3	画中画模式, 分辨率960x720
picture_in_picture_bottom_720p_16_9	画中画模式, 分辨率1280x720
side_by_side_primary_480p_9_16	主次平铺, 分辨率480x854（竖屏）
side_by_side_primary_540p_9_16	主次平铺, 分辨率540*960（竖屏）
side_by_side_primary_544p_9_16	主次平铺, 分辨率544*960（竖屏）
side_by_side_primary_720p_9_16	主次平铺, 分辨率720*1280（竖屏）
side_by_side_primary_1080p_9_16	主次平铺, 分辨率1080*1920（竖屏）
side_by_side_equal_480p_9_16	平铺模式, 分辨率480x854（竖屏）
side_by_side_equal_540p_9_16	平铺模式, 分辨率540*960（竖屏）
side_by_side_equal_544p_9_16	平铺模式，分辨率544*960（竖屏）
side_by_side_equal_720p_9_16	平铺模式，分辨率720*1280（竖屏）
side_by_side_equal_1080p_9_16	平铺模式，分辨率1080*1920（竖屏）
picture_in_picture_bottom_480p_9_16	画中画模式, 分辨率480x854（竖屏）
picture_in_picture_bottom_540p_9_16	画中画模式，分辨率540*960（竖屏）
picture_in_picture_bottom_544p_9_16	画中画模式，分辨率544*960（竖屏）
picture_in_picture_bottom_720p_9_16	画中画模式，分辨率720*1280（竖屏）
picture_in_picture_bottom_1080p_9_16	画中画模式，分辨率1080*1920（竖屏）

返回

无

事件回调

登录成功事件

```
/**
 * 登录成功回调
 * @param roomId 房间id
 * @param uid 用户uid
 * @param elapsed 登录耗时
*/
- (void)onLoinSuccess:(NSNumber *)roomId uid:(NSInteger)uid elapsed:(int)elapsed;
```

登录成功回调

参数

参数	类型	描述
roomId	NSNumber	房间id
uid	NSInteger	房间id
elapsed	int	登录耗时

错误信息回调

```
/**
* 错误信息回调
* @param errCode 错误号码
* @param errMsg 错误消息提示
* @param extInfo 附加信息
*/
- (void)onError:(NSInteger)errCode errMsg:(NSString *)errMsg extInfo:(nullable NSDictionary*)extInfo;
```

错误信息回调

参数

参数	类型	描述
errCode	NSInteger	错误码 RtcErrorCodes
errMsg	NSString	错误信息
extInfo	NSDictionary	附加信息

回调事件

```
typedef NS_ENUM(NSInteger, RtcErrorCodes) {
    NO_ERR = 0,
    ERR_INTERNAL = 500,             // 内部错误
    ERR_LOGIN_ROOM_PARAMS,          // 登录参数错误
    ERR_CHANNEL_IO_EXCEPTION,       // 信令错误
    ERR_CHANNEL_IO_DISCARD_BY_REMOTE, // 信令被远端关闭
    ERR_AUDIO_DEVICE_CAPTURE,       // 音频采集错误
    ERR_AUDIO_DEVICE_CONFIGURE,     // 音频设备配置错误
    ERR_RECONNECT_OVER_MAX_COUNT,   // 重连接超限
    ERR_SESSION_DESCRIPTION,        // 内部SDP错误
    ERR_VIDEO_VIEW_NOT_EXIST = 510, // 视频视图不存在
    ERR_VIDEO_VIEW_CANNOT_SNAPSHOT, // 视频视图未实现截图能力
    ERR_VIDEO_VIEW_SNAPSHOT_EMPTY,  // 视频视图截图结果为空
};
```

警告信息回调

```
/**
 * 警告信息回调
 * @param warningCode 警告号码
 * @param warnMsg 警告消息提示
 * @param extInfo 附加信息
 */
- (void)onWarning:(NSInteger)warningCode warningMsg:(NSString *)warnMsg extInfo:(nullable NSDictionary*)extInfo;
```

警告信息回调

链接状态变更

```
/**
* 链接状态发生变化
* @param state 链接状态值  取值: RtcConnectionState
* @param reason 状态变化原因 取值: RtcConnectionChange
*/
- (void)onConnectionStateChanged:(NSInteger)state reason:(NSInteger)reason;
```

链接状态变更回调

参数

参数	类型	描述
state	NSInteger	链接状态值 RtcConnectionState
reason	NSInteger	变更原因， 0默认行为

视频首帧

```
/**
* 视频首帧回调
* @param uid 远端用户uid
* @param width 视频宽
* @param height 视频高
*/
- (void)onFirstVideoFrame:(NSInteger)uid width:(int)width height:(int)height;
```

视频渲染首帧回调

参数

参数	类型	描述
uid	NSInteger	用户uid
width	int	用户视频宽
height	int	用户视频高

远端用户加入

```
/**
* 远端用户加入
* @param uid 用户uid
* @param userName 用户昵称
*/
- (void)onRemoteUserJoinRoom:(NSInteger)uid userName:(NSString *)userName;
```

远端用户加入事件

参数

参数	类型	描述
uid	NSInteger	用户uid
userName	NSString	用户昵称

远端用户离开

```
/**
* 远端用户离开
* @param uid 用户uid
* @param userName 用户昵称
*/
- (void)onRemoteUserLeaveRoom:(NSInteger)uid userName:(NSString *)userName;
```

远端用户离开事件

参数

参数	类型	描述
uid	NSInteger	用户uid
userName	NSString	用户昵称

远端视频流变更

```
/**
* 远端视频流状态变化
* @param uid 用户uid
* @param userName 用户昵称
* @param desc 流描述
* @param state 远端流到来 RTC_STREAM_ADD (1) ， 远端流离开 RTC_STREAM_REMOVE (2)，RtcStreamChange
*/
- (void)onVideoStreamChangedState:(NSInteger)uid userName:(NSString *)userName desc:(NSString *)desc state:
(NSInteger)state;
```

远端视频流到达/离开

参数

参数	类型	描述
uid	NSInteger	用户uid
userName	NSString	用户昵称
desc	NSString	描述 默认 "video"
state	NSInteger	状态 RtcStreamChange

远端音频流变更

```
/**
* 远端音频流状态变化
* @param uid 用户uid
* @param userName 用户昵称
* @param desc 流描述
* @param state 远端流到来 RTC_STREAM_ADD (1) ， 远端流离开 RTC_STREAM_REMOVE (2)，RtcStreamChange
*/
- (void)onAudioStreamChangedState:(NSInteger)uid userName:(NSString *)userName desc:(NSString *)desc state:
(NSInteger)state;
```

远端音频频流到达/离开

参数

参数	类型	描述
uid	NSInteger	用户uid
userName	NSString	用户昵称
desc	NSString	描述 默认 "video"
state	NSInteger	状态 RtcStreamChange

媒体通路状态变更

```
/**
* 媒体通路状态变化
* @param uplink 上行 YES， 下行 NO
* @param state 状态 RtcMediaStreamState
*/
- (void)onMediaStreamStateChange:(BOOL)uplink state:(NSInteger)state;
```

媒体链路变更

参数

参数	类型	描述
uplink	BOOL	上行 YES， 下行 NO
state	NSInteger	状态 RtcMediaStreamState

用户被禁言

```
/**
 * 用户被禁言
 */
- (void)onUserShutUp:(NSInteger)uid;
```

用户被禁言

参数

参数	类型	描述
uid	NSInteger	用户uid

用户被解除禁言

```
/**
 * 解除用户被禁言
 */
- (void)onUserDisShutUp:(NSInteger)uid;
```

用户被解除禁言

参数

参数	类型	描述
uid	NSInteger	用户uid

用户房间被解散

```
/**
 * 用户房间解散
 */
- (void)onRoomDisbanded;
```

用户房间被解散

参数

参数	类型	描述
uid	NSInteger	用户uid

用户被踢出房间

```
/**
 * 用户被踢出房间
 */
- (void)onUserKickOff:(NSInteger)uid;
```

用户被踢出房间

参数

参数	类型	描述
uid	NSInteger	用户uid

发送音视频状态

```
 /**
* 发送音视频状态
* @param type 媒体类型
* @param state 0 推流失败, 1推流成功
*/
- (void)onSendMediaState:(BRTCMediaType)type state:(int)state;
```

发送音视频状态

参数

参数	类型	描述
type	BRTCMediaType	BRTCMediaType 媒体类型
state	BRTCMediaType	0 推流失败，1推流成功

视频拉流降级

```
 /** 视频下行降级
 * @param uid 用户uid
 * @param netState 网络质量状态
 * @param demoteState 降级状态 RtcDemoteType
 */
 - (void)onPullRemoteState:(NSInteger)uid netState:(int)netState demoteState:(int)demoteState;
```

视频下行降级

参数

参数	类型	描述
uid	NSInteger	用户uid
netState	int	网络状态
demoteState	int	RtcDemoteType 降级类型

转推音视频

```
 /** 转推音视频状态
 * @param mode 直播转推模式  RtcLiveTransferMode
 * @param url  推流地址
 * @param state  状态 RtcLiveTransferState
 */
 - (void)onLivePublishState:(RtcLiveTransferMode)mode url:(NSString *)url state:(RtcLiveTransferState)state;
```

转推音视频

参数

参数	类型	描述
mode	RtcLiveTransferMode	转推类型
url	NSString	转推地址
state	RtcLiveTransferState	转推状态 RtcLiveTransferState

relay状态

```
/** MediaRealy状态更新
* @param type 类型  RtcMediaRealyType   START_MEDIA_RELAY:1  STOP_MEDIA_RELAY:2
* @param state  状态 RtcMediaRealyState
*/
- (void)onMediaRelayState:(RtcMediaRealyType)type state:(RtcMediaRealyState)state;
```

relay状态变化

参数

参数	类型	描述
type	RtcMediaRealyType	媒体类型
state	RtcMediaRealyState	状态

开启/关闭引擎统计信息

```
- (void)onEngineStatisticsInfo:(NSArray *)statistics;
```

RTC引擎状态信息统计。该callback返回当前rtc engine的一些参数和性能信息，如传输fps,码率，网络状况,cpu等信息给app

参数

参数	类型	描述
statistics	NSArray	rtc引擎状态信息统计数组

返回

无

用户音量提示回调

```
- (void)onAudioVolumeIndication:(NSArray<RtcRoomAudioLevel *> *)audioLevels;
```

当通过 enableAudioVolumeIndication 接口开启音量提示后，SDK 会通过该方法按设置间隔回调音量信息。

该方法默认不进行回调。

参数

参数	类型	描述
audioLevels	NSArray<RtcRoomAudioLevel >	音量提示信息，包括自己和远端用户

返回

无

接收到房间成员的消息回调（1/2）

```
- (void)onTextMessageArrival:(RtcMessageInfo *)msg;
```

当房间的其他用户发消息时，会收到该通知（数据通道）

参数

参数	类型	描述
msg	RtcMessageInfo	消息结构体，具体消息在该结构体中

返回

无

接收到房间成员的消息回调（2/2）

```
- (void)onTextMessageArrival2:(RtcMessageInfo *)msg;
```

当房间的其他用户发消息时，会收到该通知（信令通道）

参数

参数	类型	描述
msg	RtcMessageInfo	消息结构体，具体消息在该结构体中

返回

无

接收到房间成员SEI消息

```
- (void)onRecvSEIMsg:(NSInteger)uid message:(NSData *)message;
```

当房间的其他用户发SEI消息时，会收到该通知

参数

参数	类型	描述
uid	NSInteger	用户uid
message	NSData *	消息

返回

无

接收到获取用户属性回调

```
- (void)onTextMessageAttribute:(NSNumber *)userID attribute:(NSString *)attribute;
```

房间其他用户调用接口setUserAttribute，或本用户调用getUserAttribute获取用户Attribute属性时，该通知被触发，返回用户attribute属性

参数

参数	类型	描述
userID	NSNumber	用户ID
attribute	NSString*	属性值

返回

无

回调视频卡顿事件(1/2)

```
- (void)onVideoRenderStuckStart:(NSInteger)userId;
```

视频超过600ms未渲染回调该事件，用户通知出现视频渲染卡顿 开始

参数

参数	类型	描述
userId	NSInteger	用户ID

返回

无

回调视频卡顿事件（1/2）

```
- (void)onVideoRenderStuckEnd:(NSInteger)userId duration:(NSInteger)duration;
```

视频超过600ms未渲染回调该事件，用户通知出现视频渲染卡顿结束

参数

参数	类型	描述
userId	NSInteger	用户ID
duration	NSInteger	卡顿时长

返回

无

屏幕分享相关接口

> 使用 BRTC 屏幕分享能力，需要同时集成 BaiduRtcReplayKit.framework 扩展模块

开始应用内屏幕分享

```
- (void)startShareAppScreen API_AVAILABLE(ios(11.0));
```

开始应用内屏幕分享。仅抓取宿主应用的屏幕内容，仅支持 iOS 11 及以上系统可用

参数

无

返回

无

开始系统屏幕分享

```
- (void)startShareSystemScreenWithAppGroup:(NSString *)appGroup API_AVAILABLE(ios(11.0));
```

开始系统屏幕分享。支持抓取整个 iOS 系统的屏幕，详细接入流程及注意事项可参考 SDK 集成文档

参数

参数	类型	描述
appGroup	NSString *	用于宿主应用与录屏进程共享的 Application Group Identifier, 目前可不填

返回

无

停止屏幕分享

```
- (void)stopShareScreen;
```

停止屏幕分享

参数

无

返回

无

更新屏幕分享配置参数

```
- (int)updateScreenShareParams:(BRTCScreenShareParams *)params;
```

更新屏幕分享配置参数，包括是否开启音视频流、音视频流配置等。该接口须在 startShareAppScreen 或 startShareSystemScreenWithAppGroup: 之后调用

参数

参数	类型	描述
params	BRTCScreenShareParams *	屏幕流音视频参数

```
 @interface BRTCScreenShareAudioParams : NSObject<NSCopying>
/// 系统音频流采集音量，[0, 100]，默认 100
@property (nonatomic, assign) NSInteger volume;
@end

@interface BRTCScreenShareVideoParams : RtcVideoEncodeParams
@end

/// 屏幕分享配置参数
@interface BRTCScreenShareParams : NSObject<NSCopying>

/// 是否分享系统音频流分享
@property (nonatomic, assign) BOOL enableAudioCapture;

/// 音频流配置参数
@property (nonatomic, copy) BRTCScreenShareAudioParams *audioCaptureParams;

/// 是否分享系统屏幕视频流
@property (nonatomic, assign) BOOL enableVideoCapture;

/// 屏幕视频流配置参数
@property (nonatomic, copy) BRTCScreenShareVideoParams *videoCaptureParams;

@end
```

返回

int 参数更新结果。0 成功; -1 当前没有在进行屏幕分享

屏幕分享开始回调

```
- (void)onShareScreenStartComplete:(int)errCode;
```

开启屏幕分享的结果回调。当您通过 startShareAppScreen, startShareSystemScreen 等方法启动屏幕分享能力后，会回调该事件

参数

参数	类型	描述
errCode	int	开启屏幕分享的结果，0: 开启成功; 1: rtc 当前不可用; 2: 系统录屏功能当前不可用; 3: 系统版本不支持;

返回

无

屏幕分享结束回调

```
- (void)onShareScreenStopped:(int)reason;
```

本地屏幕分享已停止。在屏幕分享过程中，当您通过 stopShareScreen, logoutRtcRoom 等方法停止屏幕分享后，会回调该事件

参数

参数	类型	描述
reason	int	停止原因。 0: 用户调用接口主动停止

返回

无

系统屏幕分享扩展

系统屏幕分享需要基于 iOS 系统的 App Extension 机制实现，主 App 通过集成 Broadcast Setup Extension，使系统屏幕录制音视频数据可以回调到和主 App 关联的 扩展进程 中。

在扩展进程中，需要通过以下类和接口进行对接

屏幕分享扩展类

BaiduRtcRoomReplayKitExt

```
 + (instancetype)sharedInstance;

/// 以下方法和 Broadcast Setup Extension 的 RPBroadcastSampleHandler 方法一一对应，需要在其中调用这些方法
- (void)broadcastStartedWithAppGroup:(NSString *)appGroup delegate:(id<BaiduRtcRoomReplayKitExtDelegate>)delegate;
- (void)broadcastPaused;
- (void)broadcastResumed;
- (void)broadcastFinished;
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer bufferType:(RPSampleBufferType)buferType;
```

屏幕分享扩展停止回调

```
 - (void)replayKitScreenShareStopped:(BaiduRtcRoomReplayKitExt *)replayKit reason:
(BaiduRtcReplayKitStopReason)reason;
```

屏幕分享扩展进程停止回调。在系统屏幕分享过程中，当通过系统控制、主 App 停止分享、主 App 退出房间等行为停止分享，会在扩展中回调该方法

参数

参数	类型	描述
replayKit	BaiduRtcRoomReplayKitExt *	系统屏幕分享扩展类对象
reason	BaiduRtcReplayKitStopReason	系统屏幕分享停止原因，具体定义参考 BaiduRtcRoomReplayKitExt.h

返回

无

屏幕分享扩展鉴权代理

```
- (void)replayKitGetRtcTokenWithAppId:(NSString *)appId
                             roomName:(NSString *)roomName
                               userId:(NSInteger)userId
                           completion:(void (^)(NSString *tokenStr))completin;
```

如果接入 BRTC 的业务开启了业务鉴权，那么在登录房间的过程中，需要使用业务生成的 token 值。该扩展回调为系统屏幕分享进程的用户提供了业务鉴权对接能力，

在对接系统屏幕分享时，如果开启了业务鉴权，则需要实现该方法，并在 completion 中回调 token 值；如果没有开启业务鉴权，需要在 completion 中回调一个任意内容的非空字符串作为 token。

参数

参数	类型	描述
appId	NSString *	用于生成鉴权 token 的业务 appid
roomName	NSString *	用于生成鉴权 token 的房间号
userId	NSInteger	用于生成鉴权 token 的用户名
completin	void (^)(NSString *tokenStr)	用于给扩展类返回生成 token 值的回调

返回

无

美颜相关接口

> 使用 BRTC 美颜能力，需要同时集成 BaiduRtcBeauty.framework 扩展模块

获取美颜管理类实例

```
- (id<BaiduRtcBeautyManager>)getRtcBeautyManager;
```

获取 BRTC 内部音频处理管理类，用于控制 变声音效、背景音效 等音频处理效果。

参数

无

返回

当前 BRTC 实例对应的美颜管理类实例。

在调用该接口之前，需要通过 setParamSettings:paramType: 接口设置 enableBeauty = YES 以开启对应能力，否则该接口返回 nil。

开启采集视频美颜效果

```
- (void)setBeautyEnabled:(BOOL)enabled;
```

开关整体美颜效果。该接口用于临时开关美颜效果，关闭前对美颜的设置不会丢失

该接口定义在 BaiduRtcBeautyManager 扩展协议中。

参数

参数	类型	描述
enabled	BOOL	enabled 是否开启

返回

无

设置采集视频美颜效果参数

```
- (void)setBeautyFeature:(RtcBeautyFeatureType)featureType value:(NSObject *)value;
```

美颜塑形等能力通用调参接口。具体效果类型 RtcBeautyFeatureType 参考 BaiduRtcAudioProcessManager.h 中的定义。

该接口定义在 BaiduRtcBeautyManager 扩展协议中。

参数

参数	类型	描述
type	RtcBeautyFeatureType	美颜效果类型
value	NSObject *	美颜效果的设置值

返回

无

美颜特效参数模型

BRTCEffectParams

参数

参数	类型	描述
resourcePath	NSString *	特效资源地址。传入资源 case 文件地址，内部加载对应 case bundle 文件实现如 背景分割 等特效
resourceId	NSString *	特效资源 Id

设置虚拟背景

```
- (void)setHumanSegEnabled:(BOOL)enabled effectParams:(BRTCEffectParams * _Nullable)params;
```

开关虚拟背景效果，设置虚拟背景效果参数。

该接口定义在 BaiduRtcBeautyManager 扩展协议中。

参数

参数	类型	描述
enabled	BOOL	是否开启
params	BRTCEffectParams *	虚拟背景对应特效参数，传入虚拟背景资源地址

返回

无

设置外部自定义视频前处理器

```
- (void)setVideoProcessor:(BaiduRtcVideoProcessor *)processor;
```

设置外部自定义相机流编码前处理节点。

若使用 BRTC 自带的相机采集器，则可以通过该方法进行自定义前处理行为；

若已经通过 setMediaCapturer: 设置了 自定义采集器，既可以通过该方法进行自定义前处理，也可以在 自定义采集器 内部执行完前处理，再将数据输出到回调

参数

参数	类型	描述
processor	BaiduRtcVideoProcessor *	外部自定义视频前处理器

返回

无

说明

关于如何实现一个 BaiduRtcVideoProcessor，可参考 RTCVideoBufferNode.h 中的定义进行。

音频处理相关接口

> 使用 BRTC 音频处理相关接口，需要同时集成 BaiduRtcAudioProcess.framework 扩展模块

获取音频处理管理类实例

```
- (id<BaiduRtcAudioProcessManager>)getRtcAudioProcessManager;
```

获取 BRTC 内部音频处理管理类，用于控制 变声音效、背景音效 等音频处理效果。

参数

无

返回

当前 BRTC 实例对应的音频处理管理类实例。

在调用该接口之前，需要通过 setParamSettings:paramType: 接口设置 enableAudioProcess = YES 以开启对应能力，否则该接口返回 nil。

开启采集音频变声效果

```
- (void)setVoiceChangeEnabled:(BOOL)enabled;
```

开关整体变声音效。该接口用于临时开关变声音效，关闭前设置的变声效果类型不会丢失

该接口定义在 BaiduRtcAudioProcessManager 扩展协议中。

参数

参数	类型	描述
enabled	BOOL	enabled 是否开启

返回

无

设置采集音频变声效果

```
- (void)setVoiceChangeType:(RtcVoiceChangeType)type;
```

设置变声效果类型。具体类型枚举参考 BaiduRtcAudioProcessManager.h 中的定义。

该接口定义在 BaiduRtcAudioProcessManager 扩展协议中。

参数

参数	类型	描述
type	RtcVoiceChangeType	变声效果类型

返回

无

视频处理相关接口

> 使用 BRTC 视频处理相关接口，需要同时集成 BaiduRtcVideoProcess.framework 扩展模块

获取视频处理管理类实例

```
- (id<BaiduRtcVideoProcessManager>)getRtcVideoProcessManager;
```

获取 BRTC 的视频处理管理类，用于处理水印功能。 该接口定义在BaiduRtcRoomApi.h中定义。

参数

无

返回

当前 BRTC 实例对应的视频处理管理类实例。

在调用该接口之前，需要通过 setParamSettings:paramType: 接口设置 enableVideoProcess = YES 以开启对应能力，否则该接口返回 nil。

开启水印

```
- (void)enableWatermark:(BOOL)enable watermarkParams:(BRTCWatermarkParams *)watermarkParams;
```

设置是否开启水印，水印参数配置。 该接口定义在BaiduRtcVideoProcessManager.h中定义。

参数

参数	类型	描述
enable	BOOL	enable 是否开启
watermarkParams	BRTCWatermarkParams *	水印参数

返回

无

音视频录制相关接口

获取视音视频录制管理类实例

```
- (BRTCMediaRecorder *)getRtcMediaRecorder;
```

获取 BRTC 内部音视频录制管理类，用于录制本音视频或进行本地与远端音视频的混流录制。

该接口定义在BaiduRtcRoomApi.h中定义。

参数

无

返回

当前 BRTC 实例对应的音视频录制实例。

设置录制回调代理

```
- (void)setMediaRecorderDelegate:(id<BRTCMediaRecorderDelegaate>)delegate;
```

设置录制回调代理，可通过代理方法查看录制状态，录制信息等。 该接口定义在BRTCMediaRecorder.h中定义。

参数

参数	类型	描述
delegate	id	录制回调对象

返回

无

音视频录制参数模型

BRTCMediaRecordParam

参数

参数	类型	描述
type	BRTCMediaRecorderType	录制内容：本地仅音频、本地仅视频、本地音视频、混流仅音频、混流仅视频、混流音视频
mixType	BRTCMediaMixType	混流录制布局，详细说明
audioParam	BRTCAudioRecordParam *	音频参数配置
videoParam	BRTCVideoRecordParam *	视频参数配置
savePath	NSString *	录制存储路径
format	BRTCMediaRecorderContainerFormat	存储形式：aac:音频 mp4：视频/音视频
maxDurationMs	NSInteger	最大录制时长，单位为毫秒，默认值为 120000。
recorderInfoUpdateInterval	NSInteger	录制信息更新间隔，单位为毫秒，取值范围为 [1000,10000]。SDK 会根据设置值触发 informationDidUpdated 回调，报告更新的录制信息。

BRTCMediaRecorderType

枚举

```
typedef NS_ENUM (NSInteger, BRTCMediaRecorderType) {

    BRTCMediaRecorderTypeAudio,     ///< 本地录制 - 仅音频
    BRTCMediaRecorderTypeVideo,     ///< 本地录制 - 仅视频
    BRTCMediaRecorderTypeBoth,      ///< 本地录制 - 音视频

    BRTCMediaRecorderTypeMixAudio，///< 混流录制 - 仅音频
    BRTCMediaRecorderTypeMixVideo，///< 混流录制 - 仅视频
    BRTCMediaRecorderTypeMixBoth，  ///< 混流录制 - 音视频
};
```

BRTCMediaMixType

枚举

```
typedef NS_ENUM(NSInteger, BRTCMediaMixType) {
    /// 画中画：远端第一路视频是大画面，其余视频流是子画面，悬浮于画布底部从右到左依次平铺，高宽为画布 1/5，子画面内容适配填充。
    BRTCMediaMixTypePip,
    /// 主次平铺：远端第一路视频是大画面，高和画布相同，宽为画布 4/5。其余视频流是子画面，垂直排列于画布右侧从上到下依次平铺，高宽为画布 1/5，子画面内容适配填充。
    BRTCMediaMixTypePas,
    /// 平铺：本地视频和远端视频按从左到右从上到下按宫格形式依次平铺，高宽均分。
    BRTCMediaMixTypeTile,
};
```

BRTCAudioRecordParam

参数

参数	类型	描述
numberOfChannel	int	录制文件的音频声道数
sampleRate	int	录制文件的音频采样率
audioBitrate	int	录制文件的音频码率，kbps

BRTCVideoRecordParam

参数

参数	类型	描述
videoFps	int	录制文件的视频帧率
videoWidth	int	录制文件的视频宽度
videoHeight	int	录制文件的视频高度
videoBitrate	int	录制文件的视频码率，kbps

开始录制

```
- (int)startRecordWithParam:(BRTCMediaRecordParam *)param;
```

开始录制音视频，可录制媒体内容为本地或混流、类型为仅音频、仅视频 或 音视频。

该接口定义在BRTCMediaRecorder.h中定义。

> 使用混流录制，需要同时集成 BaiduRtcVideoProcess.framework 扩展模块，

> 并通过 setParamSettings:paramType: 接口设置 enableVideoProcess = YES 启用扩展模块。
>
> 混流最多展示6路媒体流。

参数

参数	类型	描述
param	BRTCMediaRecordParam *	录制参数配置

返回

开始录制的状态结果，0表示开始录制成功，其他表示失败。 具体可查看BRTCMediaRecorderErrorCode，也可通过代理回调查看。

停止录制

```
- (int)stopRecord;
```

停止录制，停止录制成功，可通过配置的路径查看录制的内容。

参数

无

返回

停止录制的状态结果，0表示停止录制成功，其他表示失败。 具体可查看BRTCMediaRecorderErrorCode，也可通过代理回调查看。

音乐文件播放相关接口

> 使用 BRTC 音乐文件播放相关接口，需要同时集成 BaiduRtcAudioProcess.framework 扩展模块

获取音乐文件播放管理对象

```
- (id<BaiduRtcAudioProcessManager>)getRtcAudioProcessManager;
```

获取 BRTC 内部音频处理管理类，用于控制 变声音效、背景音效 等音频处理效果。此接口在BaiduRtcRoomApi.h中定义。

参数

无

返回

当前 BRTC 实例对应的音频处理管理类实例。

在调用该接口之前，需要通过 setParamSettings:paramType: 接口设置 enableAudioProcess = YES 以开启对应能力，否则该接口返回 nil。

设置音乐文件播放状态代理 - (void)setAudioProcessDelegate:(id)delegate;

设置音频处理回调代理。此接口在BaiduRtcAudioProcessManager.h中定义。

参数

参数	类型	描述
delegate	id<BaiduRtcAudioProcessDelegate>	遵循 BaiduRtcAudioProcessDelegate 协议的代理对象

返回

无

开始音乐文件播放混音

```
- (int)startAudioMixing:(BRTCAudioMixingParams *)params;
```

开始音乐文件播放混音。

参数

参数	类型	描述
params	BRTCAudioMixingParams *	音乐文件混音参数

BRTCAudioMixingParams对象参数

参数

参数	类型	描述
filePath	NSString *	本地音乐文件路径
loopback	BOOL	是否只在本地播放音乐文件，YES 表示仅本地播放，NO 表示播放同时将音乐文件混音推流，默认 YES。
cycle	NSInteger	音乐文件播放次数，≥ 0 表示次数，-1 表示不限制，默认 -1。

返回

调用方法返回的结果，0表示调用开始播放方法成功，其他表示失败。

暂停音乐文件播放混音

```
- (int)pauseAudioMixing;
```

暂停音乐文件播放混音。

参数

无

返回

调用方法返回的结果，0表示调用暂停播放方法成功，其他表示失败。

恢复音乐文件播放混音

```
- (int)resumeAudioMixing;
```

恢复音乐文件播放混音。

参数

无

返回

调用方法返回的结果，0表示调用恢复播放方法成功，其他表示失败。

停止音乐文件播放混音

```
- (int)stopAudioMixing;
```

停止音乐文件播放混音。

参数

无

返回

调用方法返回的结果，0表示调用停止播放方法成功，其他表示失败。

其他接口

获取SDK版本号

```
+ (NSString *)version;
```

获取 iOS SDK 版本号。

参数

无

返回

返回当前版本号

开启/关闭调试信息输出

```
+ (void)setVerbose:(BOOL)bOnVerbose;
```

是否打开调试信息。

建议在初始化 SDK 前调用。建议在调试阶段打开此开关，方便调试。

参数

参数	类型	描述
bOnVerbose	BOOL	是否打开调试信息，true 打开，false 不打开。默认为 false

返回

无

查询RTC统计信息

```
- (NSArray *)queryEngineStatisticsInfo;
```

查询RTC统计信息。 如CPU, FPS，video codec等

参数

无

返回

NSArray* rtc 状态统计信息

开启/关闭RTC质量监控数据上报

```
- (void)enableStatsToServer:(BOOL)isEnable qualityMonitorEnv:(NSString *)qualityMonitorEnv;
```

RTC质量监控数据上报。

预置接口，监控信息上报开关 当打开开关时，上报帧率、码率、分辨率、丢包率等监控信息到服务端，console可查。

参数

参数	类型	描述
isEnable	BOOL	是否打开rtc质量监控数据上报，true 打开，false不打开。默认为 false
qualityMonitorEnv	NSString*	线上环境："online" 沙盒："qa"。 默认值为 "online"

返回

无 **开启/关闭RTC异常信息上报**

```
- (void)enableErrorInfoReprot:(BOOL)enableErrorInfoReport;
```

RTC异常信息上报开关。

上报异常信息，方便问题排查

参数

参数	类型	描述
enableErrorInfoReport	BOOL	enableErrorInfoReport 是否打开上报开关 true 开启上报, false 关闭上报。默认公网域名开启上报

返回

无

白板

创建对象

```
 // 获取白板对象
+ (instancetype)sharedManager;

// 设置回调
- (void)setBaiduRtcBoardDelegate:(id<BaiduRtcBoardDelegate>)delegate;
```

白板登录

```
 @interface BoardLoginParam : NSObject

/// 房间名
@property (nonatomic, copy) NSString *channelName;

/// app key
@property (nonatomic, copy) NSString *appId;

/// app secret
@property (nonatomic, copy) NSString *token;

/// uid
@property (nonatomic, assign) NSUInteger uid;

/// 是否开启 web 调试日志
@property (nonatomic, assign) BOOL debug;

/// 是否开启观众模式
@property (nonatomic, assign) BOOL audience;

@end

// 登录白板
- (void)callWebLoginWithParam:(BoardLoginParam *)param;
```

登录白板房间

设置画笔

```
 // 设置画笔类型 自由画笔、直线、剪头、椭圆、菱形、矩形、文本、激光笔等
- (void)setToolType:(RtcBoardToolType)type;

// 设置画笔颜色
- (void)setBrushColor:(UIColor *)color;
```

清除白板

```
 // 清除白板
- (void)clear;
```

通话前测速相关

Lastmile初始化

```
- (instancetype)initProbeTest:(NSString *)appId
                     tokenStr:(NSString *)tokenStr
                uplinkBitrate:(int)uplinkBitrate
              downlinkBitrate:(int)downlinkBitrate
                     delegate:(id<BaiduRtcLastmileDelegate>)delegate;
```

Lastmile初始化。

初始化Lastmile并创建网络测速实例 初始化网络测速时调用，初始化网络测速失败会导致网络测速功能异常。

参数

参数	类型	描述
appId	NSString	百度为用户App签发的App ID, 用于识别应用, 全局唯一，详见创建应用。
tokenStr	NSString	百度智能云RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。详见应用鉴权。
uplinkBitrate	int	用户期望的最高发送码率，单位为bps，范围为100000～5000000。
downlinkBitrate	int	用户期望的最高接收码率，单位为bps，范围为100000～5000000。
delegate	BaiduRtcLastmileDelegate	遵循BaiduRtcLastmileDelegate协议的代理对象 。

返回

返回Lastmile实例，nil 表示初始化失败

Lastmile结束

```
- (void)stopProbeTest;
```

结束网络探测, 可以在网络探测回调函数lastmileProbeTestResult中调用, 也可以手动触发结束。

BaiduRtcLastmileDelegate代理对象

```
- (void)rtcLastmile:(BaiduRtcLastmile *)lastmile lastmileProbeTestResult:(RtcLastmileResult *)result;
```

参数

参数	类型	描述
lastmile	BaiduRtcLastmile	Lastmile实例
result	lastmileProbeTestResult	开始通话前网络质量探测，向用户反馈上下行网络的带宽，丢包，网络抖动和往返时延数据。

RtcLastmileResult对象参数

参数

参数	类型	描述
state	RtcLastmileResultState	网络探测状态, 参考RtcLastmileResultState。
rtt	NSUInteger	网络探测往返平均延时, 单位(ms)。
quality	NSUInteger	探测网络质量, 参考RtcLastmileQuality。
uplinkReport	RtcLastmileOneWayResult	上行网络探测质量。
downlinkReport	RtcLastmileOneWayResult	下行网络探测质量。

RtcLastmileOneWayResult对象参数

参数

参数	类型	描述
lossRate	NSUInteger	探测的网络丢包率。
jitter	NSUInteger	探测的网络抖动，单位(ms)。
bandwidth	NSUInteger	探测的网络带宽，单位(Kbps)。

RtcLastmileResultState 状态定义

状态	值	含义
RtcLastmileResultComplete	1	本次质量探测是完整的
RtcLastmileResultIncompleteNoBwe	2	本次质量探测未进行带宽预测，因此结果不完整
RtcLastmileResultUnavailable	3	未进行质量探测。一个可能的原因是网络连接中断

RtcLastmileQuality 质量定义

状态	值	含义
RtcLastmileQualityUnknown	0	质量未知
RtcLastmileQualityExcellent	1	质量极好
RtcLastmileQualityGood	2	用户主观感觉和极好差不多，但码率可能略低于极好
RtcLastmileQualityPoor	3	用户主观感受有瑕疵但不影响沟通
RtcLastmileQualityBad	4	勉强能沟通但不顺畅
RtcLastmileQualityVBad	5	网络质量非常差，基本不能沟通
RtcLastmileQualityDown	6	完全无法沟通
RtcLastmileQualityDetecting	8	SDK正在探测网络质量

Web SDK

概述

BRTC Web端SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

API

启动BRTC SDK

```
BRTC_Start()
```

介绍

启动SDK时使用。

参数

参数	类型	描述
server	string	百度的RTC 服务器， 使用默认值即可。
appid	string	百度派发的AppID，开发者的唯一标识
token	string	app server 派发的token字符串，用来校验通信的合法性

该接口参数数量较多，请参考下面的参数详解进行了解。

返回

无

参数详解

参数	类型	描述	默认值
server	string	百度的RTC 服务器， 使用默认值即可。	"wss://rtc.exp.bcelive.com/janus"
appid	string	百度派发的AppID, 开发者的唯一标识	
token	string	app server 派发的token字符串, 用来校验对应通信的合法性	
roomname	string	房间名称	
userid	string	用户ID, 小于2^53整数的字符串	
displayname	string	显示的用户名	
remotevideoviewid	string	显示远端视频，来自html的DOM对象的ID名称	
localvideoviewid	string	显示本地摄像头视频，来自html的DOM对象的ID名称	
showvideobps	bool	是否显示视频的带宽值	默认true，显示。
shownovideo	bool	不存在视频时的显示提示	默认true，显示。
showspinner	bool	是否显示加载过程	默认true
aspublisher	bool	是否是发布者	默认true，是发布者
usingdatachannel	bool	是否使用数据通道	默认true， 开启数据通道
usingvideo	bool	是否使用本地视频设备	默认true
usingaudio	bool	是否使用本地音频设备	默认true
sharescreen	bool	是否是屏幕共享	默认false
videodeviceid	string	视频设备ID	默认使用摄像头， 设备的值可以通过函数BRTC_GetVideoDevices获得
rtmpserver	string	直播转推流的地址	格式是："rtmp://server/stream"
rtmpmixtemplate	string	直播转推模版	模板名称，默认值为默认大小窗口模版
rtmpmix	bool	直播转推是否混流	默认false
rtmpmixlayoutindex	string	转推混流ID标识	配合RTMP转推混流函数BRTC_StartLiveServerStreamingEx使用
autosubscribe	bool	是否自动订阅流	默认true
autopublish	bool	是否自动发布流	默认true
waitpermissiontimeoutms	int	等待权限超时大小，单位是毫秒	默认值是180000毫秒

candidateip	string	代理服务器的IP	默认值null
mediaserverip	string	代理指向的媒体服务器IP	默认值null
videocodec	string	视频编码类型配置	可选参数, 默认值是'h264'; 支持'h264','vp8','h263','vp9'视频编码格式, 需要房间中的每个用户配置一致.
videoprofile	string 或 Object	本地摄像头视频配置参数	'lowres','stdres','hires','fhdres','4kres'或者配置视频的宽高值 { 'height': {'ideal': 240}, 'width': {'ideal': 240}},
bitrate	int	设置摄像头编码的码率	默认值浏览器根据带宽决定， 单位kbps， 对于720p视频建议设置为1500
screenbitrate	int	设置屏幕共享的码率	默认值1500，单位kbps
remotevideoon	function(idx)	远端视频流到达的回调	
remotevideooff	function(idx)	远端视频流离开的回调	
remotevideocoming	function(id,display,attribute)	远端用户流上线的回调	
remotevideoleaving	function(id)	远端用户流离开的回调	
onlocalstream	function(stream,name)	本地视频流自动， name是流名称，相机是'camera'或屏幕共享'screen'	
onlocalstream_end	function(name)	本地视频流关闭， name是流名称，相机是'camera'或屏幕共享'screen'	
localvideopublishing	function	本地视频开始发布的回调	
localvideopublished_ok	function	本地视频成功发布到回调	
onmessage	function(msg)	消息事件回调{msg.id,msg.data}	
onattribute	function(a)	属性事件回调{a.id,a.attribute}	
userevent	bool	是否启用用户级事件	true表示启用用户级事件， 当用户一旦加入房间就会发出事件， 无需用户推流。 false 表示不启用
userevent_joinedroom	function(id,display,attribute)	用户加入房间的事件，此时用户还没有发布流	
userevent_leavingroom	function(id,display)	用户离开房间，用户已经关闭了流或者没有发布过流	
success	function()	BRTC_Start()成功	
error	function(error)	BRTC_Start()失败，或运行过程中出现了错误	
destroyed	function(error)	运行过程中出现错误被销毁的回调	
debuglevel	bool/array	是否打印调试信息	默认值为true， 可取值为: true, false, 'all', ['debug','log','error']

停止BRTC

```
BRTC_Stop()
```

介绍

停止BRTC

参数

无

返回

无

获得SDK的版本号

```
BRTC_Version();
```

介绍

获得SDK的版本号信息。

参数

无

返回

返回BRTC SDK的版本号

获得视频设备列表

```
BRTC_GetVideoDevices()
```

介绍

获得视频设备列表

返回

通过回调返回视频设备列表。

参数详解

```
 var callback = {
success: function (devices) {
   const str = JSON.stringify(devices);
   }
}
BRTC_GetVideoDevices(callback);
```

切换视频设备

```
BRTC_ReplaceVideo(DeviceID)
```

介绍

用于多个摄像头设备上进行摄像头切换。

参数

参数	类型	描述
DeviceID	string	视频设备的ID号，可以通过BRTC_GetVideoDevices获得视频设备列表

返回

无

替换视频源

```
BRTC_ReplaceStream(stream);
```

介绍

在已经推流的实例上使用，用于把video或canvas标签抓取的内容作为源发送出去。

参数

参数	类型	描述
stream	MediaStream	抓取canvas的视频流: var stream = canvas.captureStream(25);

返回

无

订阅指定的视频流

```
BRTC_SubscribeStreaming(manualvideo, feedid);
```

介绍

订阅指定的视频流，并显示到指定的视频标签上。 前置条件: 需要在调用BRTC_Start时设置 autosubscribe: false

参数

参数	类型	描述
manualvideo	string	用于显示视频的DOM标签ID值
feedid	string	要订阅的视频流ID，可以在BRTC_Start方法的回调函数remotevideocoming中获得，或者在BRTC_GetRemoteUsers中取得。

返回

无

停止订阅视频流

```
BRTC_StopSubscribeStreaming(feedid);
```

介绍

停止订阅的视频流。

参数

参数	类型	描述
feedid	string	要停止订阅的视频流的ID，标识一个特定视频流

返回

无

获得房间中的远端用户列表

```
BRTC_GetRemoteUsers();
```

介绍

获得房间中的远端用户列表

参数

无

返回

房间中的远端用户列表数组。

返回结果示例：

```
[
  {"id":100219207,"display":"Tom","attribute":""},
  {"id":100241823,"display":"brtc webclient","attribute":""}
]
```

发布本地音视频流

```
BRTC_StartPublish();
```

介绍

在BRTC_Start()中把autopublish 设置为false时， 可以使用本函数发布音视频流

参数

无

返回

无

停止发布本地音视频流

```
BRTC_StopPublish();
```

介绍

可以使用本函数停止发布音视频流

参数

无

返回

无

重新设置发布流的参数

```
BRTC_SetParamSettings({
        usingdatachannel: true,
        usingaudio: true,
        usingvideo: true ，
        videoprofile: "hires"});
```

介绍

重新设置发布流的参数， 用于在BRTC_StopPublish后改变参数， 然后调用BRTC_StartPublish进行重新发布流。

参数

无

返回

无

麦克风静音

```
BRTC_MuteMicphone(muted);
```

参数

参数	类型	描述
muted	bool	静音标识， true表示要静音， false 是取消静音

返回

无

静默视频

```
BRTC_MuteCamera(muted);
```

参数

参数	类型	描述
muted	bool	静默标识， true表示要静默， false 是取消静默

返回

无

屏幕分享切换

```
BRTC_SwitchScreen();
```

介绍

用于屏幕分享和摄像头之间切换视频源

参数

无

返回

无

返回远端音频等级

```
BRTC_GetRemoteAudioLevels();
```

参数

无

返回

返回结果为一个数组，数组中每个元素表示一个远端流的audioLevel信息；audioLevel为0到1之间的浮点数，0表示静音，取值越大声音越大。

返回结果示例：

```
[
  {
    "feedId":100123,              // 被listen的用户ID
    "display":"brtc webclient",   // 用户展示名
    "ssrcs":[                     // SSRC,一个audioReceiver中会存在多个ssrc，故使用数组表示
      {
        "audioLevel":0.00008912509381337459,     // audioLevel值
        "rtpTimestamp":1665600,
        "source":1526252044,                     // source ID，一个流里面可能有多个source
        "timestamp":1574051117355
      }
    ]
  }
]
```

返回远端视频信息

```
BRTC_GetRemoteVideoInfo(feedid);
```

参数

参数	类型	描述
feedid	int	远端视频流的ID

返回

返回结果为远端视频信息的对象， 包含视频宽高，码率， 帧率信息。

返回结果示例：

```
{
  id: 78757951,
  bitrate: "1020 kbits/sec",
  fps: 15,
  width: 480,
  height: 640
}
```

设置用户属性

```
BRTC_SetUserAttribute(attribute);
```

介绍

属性事件回调会发生在BRTC_Start的onattribute 函数上。 属性值保存在BRTC_Start回调函数 remotevideocoming返回的参数里面。

参数

attribute, 表示用户属性的字符串，比如: "{name:'aaa',tel:'12345'}";

返回

无

获取用户属性值

```
BRTC_GetUserAttribute();
```

介绍

根据用户ID获得特定用户的属性值。

参数 {onattribute: function (a){} ,feedid: id} onattribute, 设置的回调函数。 feedid， 要获取的用户ID号。

返回

返回的属性值在onattribute回调中。

发送用户消息

```
BRTC_SendMessageToUser(msg,id);
```

介绍

本函数用来给特定ID用户发送消息或者向房间内发送广播消息。 消息在接收端的onmessage回调函数中接收。发送用户消息的频率应小于10次/秒，超出的话用户消息可能会被丢弃。

前置条件: 在调用BRTC_Start登录成功后才能调用。

参数

msg, 需要发送的消息内容， 为一个字符串， 比如: "{name:'aaa',tel:'12345'}"

id, 需要发送消息给对端用户的ID值。 **注意: 当id为0或没有id参数时表示在房间内发送广播消息。**

返回

无

开启RTMP直播流转推

```
BRTC_StartLiveServerStreaming(url, mix, mixtemplate, level);
```

介绍

本函数用来向RTMP接流服务器转推数据流。

参数

参数	类型	描述
url	string	RTMP接流地址URL
mix	bool	是否混流， true表示要多路混流， false表示不要混流
mixtemplate	string	混流模板， 取值详见下表
level	string	转推模式， 取值有: "room" 表示聊天室模式，"anchor" 表示主播转推模式

混流模板名称	说明
side_by_side_primary_360p_4_3	主次平铺, 分辨率 480x360 (固定码率、帧率)
side_by_side_primary_360p_4_3-xx-xx	主次平铺, 分辨率 480x360 (自定义码率、帧率) 下面模板也可以设置
side_by_side_primary_360p_16_9	主次平铺, 分辨率640x360
side_by_side_primary_480p_4_3	主次平铺, 分辨率640x480
side_by_side_primary_480p_16_9	主次平铺, 分辨率854x480
side_by_side_primary_540p_16_9	主次平铺, 分辨率960x540
side_by_side_primary_720p_4_3	主次平铺, 分辨率960x720
side_by_side_primary_720p_16_9	主次平铺, 分辨率1280x720
side_by_side_equal_360p_4_3	平铺模式(大小相等), 分辨率480x360
side_by_side_equal_360p_16_9	平铺模式(大小相等), 分辨率640x360
side_by_side_equal_480p_4_3	平铺模式(大小相等), 分辨率640x480
side_by_side_equal_480p_16_9	平铺模式(大小相等), 分辨率854x480
side_by_side_equal_540p_16_9	平铺模式(大小相等), 分辨率960x540
side_by_side_equal_720p_4_3	平铺模式(大小相等), 分辨率960x720
side_by_side_equal_720p_16_9	平铺模式(大小相等), 分辨率1280x720
picture_in_picture_bottom_360p_4_3	画中画模式, 分辨率480x360
picture_in_picture_bottom_360p_16_9	画中画模式, 分辨率640x360
picture_in_picture_bottom_480p_4_3	画中画模式, 分辨率640x480
picture_in_picture_bottom_480p_16_9	画中画模式, 分辨率854x480
picture_in_picture_bottom_540p_16_9	画中画模式, 分辨率960x540
picture_in_picture_bottom_720p_4_3	画中画模式, 分辨率960x720
picture_in_picture_bottom_720p_16_9	画中画模式, 分辨率1280x720
side_by_side_primary_480p_9_16	主次平铺, 分辨率480x854（竖屏）
side_by_side_primary_540p_9_16	主次平铺, 分辨率540x960（竖屏）
side_by_side_primary_544p_9_16	主次平铺, 分辨率544x960（竖屏）
side_by_side_primary_720p_9_16	主次平铺, 分辨率720x1280（竖屏）
side_by_side_primary_1080p_9_16	主次平铺, 分辨率1080x1920（竖屏）
side_by_side_equal_480p_9_16	平铺模式, 分辨率480x854（竖屏）
side_by_side_equal_540p_9_16	平铺模式, 分辨率540x960（竖屏）
side_by_side_equal_544p_9_16	平铺模式，分辨率544x960（竖屏）
side_by_side_equal_720p_9_16	平铺模式，分辨率720x1280（竖屏）
side_by_side_equal_1080p_9_16	平铺模式，分辨率1080x1920（竖屏）
picture_in_picture_bottom_480p_9_16	画中画模式, 分辨率480x854（竖屏）
picture_in_picture_bottom_540p_9_16	画中画模式，分辨率540x960（竖屏）
picture_in_picture_bottom_544p_9_16	画中画模式，分辨率544x960（竖屏）
picture_in_picture_bottom_720p_9_16	画中画模式，分辨率720x1280（竖屏）
picture_in_picture_bottom_1080p_9_16	画中画模式，分辨率1080x1920（竖屏）

返回

无

开启RTMP直播流转推增强接口

```
BRTC_StartLiveServerStreamingEx(url, level, mixcfg);
```

介绍

本函数用来向RTMP接流服务器转推数据流， 支持配置混流背景水印等。

参数

参数	类型	描述	参考值
url	string	RTMP接流地址URL	-
level	string	转推模式	取值 "room" 表示聊天室模式，"anchor" 表示主播转推模式
mixcfg	object	混流转码配置对象	-

mixcfg详解

成员	类型	描述	参考值
mode	string	混流模式	必选成员，单路音视频转推使用"single"；多路音视频转推使用 "mix"；纯音频混流使用 "pure_audio"，无视频输出
bgpUrl	string	背景图片	背景图片的URL地址
encodeCfg	object	混流编码配置	-
encodeCfg.audioChannel	int	混流音频编码通道数	音频通道数取值1表示单声道，2表示双声道
encodeCfg.audioClockRate	int	混流音频编码采样率	推荐值48000，支持取值 8000，12000，16000，22050，24000，32000，44100，48000，88200，96000
encodeCfg.width	int	混流视频宽度	默认值1280， 取值范围0~1920，单位为像素值
encodeCfg.height	int	混流视频高度	默认值720， 取值范围0~1920，单位为像素值
encodeCfg.videoBitrate	int	混流视频编码码率	默认值1200kbps
encodeCfg.fps	int	混流视频帧率	默认值20fps
layoutCfg	object	布局配置	-
layoutCfg.layout	string	布局模板	默认值为 "side_by_side_primary"，可取值 "side_by_side_primary"，"side_by_side_equal"，"picture_in_picture_bottom"，"custom"; 取值 "custom" 时表示采用自定义窗口布局
layoutCfg.windowCfgs	array	自定义窗口布局	为每一路输入流配置一个窗口
layoutCfg.windowCfgs[0].id	string	窗口ID	与用户加入房间时BRTC_Start()中的 rtmpmixlayoutindex一致

layoutCfg.windowCfgs[0].renderMode	int	窗口绘制模式	可取值0和1，0表示缩放后裁剪，1表示缩放并显示黑底，默认值为0
layoutCfg.windowCfgs[0].xpos	int	窗口X坐标	该窗口在输出时矩形左上角的X坐标，单位为像素值，窗口的 xpos 与 width之和不能超过混流输出的总宽度（encodeCfg 中 width），默认为0
layoutCfg.windowCfgs[0].ypos	int	窗口Y坐标	该窗口在输出时矩形左上角的Y坐标，单位为像素值，窗口的 ypos 与 height之和不能超过混流输出的总高度（encodeCfg 中 height），默认为0
layoutCfg.windowCfgs[0].zorder	int	窗口在输出时的Z层级	默认为0，zorder越大处于画面越上层，取值为0时窗口位于最底层
layoutCfg.windowCfgs[0].width	int	窗口宽度	-
layoutCfg.windowCfgs[0].height	int	窗口高度	-
watermarkCfgs	array	水印配置数组	-
watermarkCfgs[0].type	string	水印类型	"image" 表示图片水印，"text" 表示文字水印，"clock" 表示时间戳水印
watermarkCfgs[0].xpos	int	水印X坐标	-
watermarkCfgs[0].ypos	int	水印Y坐标	-
watermarkCfgs[0].width	int	水印宽度	-
watermarkCfgs[0].height	int	水印高度	-
watermarkCfgs[0].imageUrl	string	图片水印地址	水印type 为"image" 时必须携带，图片水印URL，水印图片只支持缩放
watermarkCfgs[0].text	string	水印文本内容	用于文字水印和时间戳水印
watermarkCfgs[0].font	string	水印字体	默认值"Normal"，用于文字水印和时间戳水印
watermarkCfgs[0].size	int	水印字体大小	默认值22， 用于文字水印和时间戳水印
watermarkCfgs[0].color	string	水印字体颜色	用于文字水印和时间戳水印， 采用big-endian ARGB的方式，并使用16进制表示。背景颜色取值共10位，前2位固定为0x，接下来2位表示透明度，后6位表示颜色值，例如"0xFF000000"
watermarkCfgs[0].timeFormat	string	时间戳水印的时间格式	默认 "%Y-%m-%d %H:%M:%S"，时间戳格式参考：https://man7.org/linux/man-pages/man3/strftime.3.html

mixcfg对象示例：

```
{
  "mode":"mix",
```

```
    "bgpUrl":"https://*.jpg",
    "encodeCfg":{
        "audioChannel":2,
        "audioClockRate":48000,
        "width":1280,
        "height":720,
        "videoBitrate":1200,
        "fps":20
    },
    "layoutCfg":{
        "layout":"custom",
        "windowCfgs":[
            {
                "id":"1",
                "renderMode":0,
                "xpos":200,
                "ypos":200,
                "zorder":1,
                "width":100,
                "height":100
            },
            {
                "id":"2",
                "renderMode":1,
                "xpos":400,
                "ypos":400,
                "zorder":2,
                "width":200,
                "height":200
            }
        ]
    },
    "watermarkCfgs":[
        {
            "type":"image",
            "imageUrl":"https://*.jpg",
            "xpos":0,
            "ypos":0,
            "width":200,
            "height":200
        },
        {
            "type":"text",
            "text":"Hello world!",
            "xpos":100,
            "ypos":200,
            "font":"Normal",
            "size":22,
            "color":"0xFF000000"
        },
        {
            "type":"clock",
            "timeFormat":"%Y-%m-%d %H:%M:%S",
            "text":"北京时间：",
            "xpos":300,
            "ypos":400,
            "font":"Normal",
            "size":22,
            "color":"0xFF000000"
        }
    ]
}
```

停止RTMP直播流转推

```
BRTC_StopLiveServerStreaming(level);
```

介绍

本函数用来停止向RTMP接流服务器转推。

参数

参数	类型	描述
level	string	需要停止的转推模式， 取值有: "anchor", "room"

返回

无

开启录制

```
BRTC_StartRecording(level, mixcfg, filecfg, storagecfg);
```

介绍

本函数用于开启服务端录制， 支持配置混流背景水印，文件名称配置等。

注意: 此录制函数将废弃，可以使用服务端API进行录制-云端录制相关接口。

参数

参数	类型	描述	参考值
level	string	录制级别	取值 "room" 表示房间录制，"user" 表示用户录制模式
mixcfg	object	混流转码配置对象	必选参数， 混流转码配置
filecfg	object	文件配置对象	默认值采用百度智能云控制台配置
storagecfg	object	存储配置对象	默认值采用百度智能云控制台配置

mixcfg详解

成员	类型	描述	参考值
mode	string	混流模式	必选成员，单路音视频转推使用"single"；多路音视频转推使用 'mix'；纯音频混流使用"pure_audio"，无视频输出
encodeCfg	object	混流编码配置	-
encodeCfg.audioCodec	string	混流音频编码	音频编码，默认值为"aac", 该配置项只有在 mixcfg.mode="pure_audio" 时生效，支持取值 "aac"、"mp3"; 其它模式均为默认值 'aac'
encodeCfg.width	int	混流视频宽度	默认值1280， 取值范围0～1920，单位为像素值
encodeCfg.height	int	混流视频高度	默认值720， 取值范围0～1920，单位为像素值
encodeCfg.videoBitrate	int	混流视频编码码率	默认值1200Kbps
layoutCfg	object	布局配置	-
layoutCfg.layout	string	布局模板	默认值为 "side_by_side_primary"，可取值 "side_by_side_primary"，"side_by_side_equal"，"picture_in_picture_bottom"；
watermarkCfgs	array	水印配置数组	-
watermarkCfgs[0].type	string	水印类型	"image' 表示图片水印，"text" 表示文字水印，"clock" 表示时间戳水印
watermarkCfgs[0].xpos	int	水印X坐标	-
watermarkCfgs[0].ypos	int	水印Y坐标	-
watermarkCfgs[0].width	int	水印宽度	-
watermarkCfgs[0].height	int	水印高度	-
watermarkCfgs[0].imageUrl	string	图片水印地址	水印type 为"image" 时必须携带，图片水印URL，水印图片只支持缩放
watermarkCfgs[0].text	string	水印文本内容	用于文字水印和时间戳水印
watermarkCfgs[0].font	string	水印字体	默认值"Normal"，用于文字水印和时间戳水印
watermarkCfgs[0].size	int	水印字体大小	默认值22， 用于文字水印和时间戳水印
watermarkCfgs[0].color	string	水印字体颜色	用于文字水印和时间戳水印， 采用big-endian ARGB的方式，并使用16进制表示。背景颜色取值共10位，前2位固定为0x，接下来2位表示透明度，后6位表示颜色值，例如"0xFF000000"
watermarkCfgs[0].timeFormat	string	时间戳水印的时间格式	默认 "%Y-%m-%d %H:%M:%S"，时间戳格式参考：https://man7.org/linux/man-pages/man3/strftime.3.html

mixcfg对象示例：

```
{
  "mode":"mix",
  "encodeCfg":{
     "audioCodec": "aac",
     "width":1280,
     "height":720,
     "videoBitrate":1200,
  },
  "layoutCfg":{
     "layout":"side_by_side_primary",
  },
  "watermarkCfgs":[
     {
        "type":"image",
        "imageUrl":"https://*.jpg",
        "xpos":0,
        "ypos":0,
        "width":200,
        "height":200
     },
     {
        "type":"text",
        "text":"Hello world!",
        "xpos":100,
        "ypos":200,
        "font":"Normal",
        "size":22,
        "color":"0xFF000000"
     },
     {
        "type":"clock",
        "timeFormat":"%Y-%m-%d %H:%M:%S",
        "text":"北京时间：",
        "xpos":300,
        "ypos":400,
        "font":"Normal",
        "size":22,
        "color":"0xFF000000"
     }
  ]
}
```

filecfg详解

成员	类型	描述	参考值
type	string	录制文件格式	该配置项在 mixcfg.mode="mix" 时支持取值 "flv"、"mp4"； mixcfg.mode="pure_audio"时，支持取值为 "aac"、"mp3"，并且此时 type取值需要和 audioCodec 取值相同； mixcfg.mode="single" 时不可配置，只支持取值 "flv"
maxDuration	int	单文件录制时长	单位分钟，达到该阈值时将自动切割文件，默认值为60

filecfg对象示例：

```
{
  "type": "mp4",
  "maxDuration": 60
}
```

storagecfg详解

成员	类型	描述	参考值
vendor	string	存储平台名	目前只支持取值"0"，表示百度智能云 BOS存储
filenameFormat	string	录制文件命名模式	对应BOS平台上文件的路径和文件名。 mixcfg.mode="mix" 或者 mixcfg.mode="pure_audio" 时，默认值为 "%a/%r/%d/recording_%t.%f"，并且要求取值必须含有%t，必须以.%f结尾 mixcfg.mode='single'，默认值为 "%a/%r/%d/recording_%t_%u.%f"，并且要求取值必须含有%t_%u，必须以.%f结尾。

storagecfg对象示例：

```
{
  "vendor": "0",
  "filenameFormat": "%a/%r/%d/recording_%t.%f"
}
```

停止录制

```
BRTC_StopRecording(level);
```

介绍

本函数用来停止服务器录制。

注意: 此函数将废弃，可以使用服务端API进行录制停止-云端停止录制接口。

参数

参数	类型	描述
level	string	需要停止的录制模式， 取值有: "room", "user"

返回

无

摄像头设备的回显测试启动 BRTC_DeviceTest_Start({localvideoviewid:”testvideo”,videoprofile:"hires"});

介绍

测试摄像头设备的回显。

参数

localvideoviewid, 要进行视频回显的DOM ID名称。 videoprofile, 回显视频的分辨率设置。可以取值为：
"lowres","stdres","hires","fhdres","4kres","480x480@15"

返回

无

关闭摄像头设备的回显 BRTC_DeviceTest_Stop();

介绍

关闭摄像头设备的回显。

参数

无

返回

无

静默摄像头设备测试的回显 BRTC_DeviceTest_MuteCamera(muted);

介绍

静默摄像头设备测试的回显。

参数

muted, 是否静默摄像头， true表示静默摄像，false表示打开摄像头。

返回

无

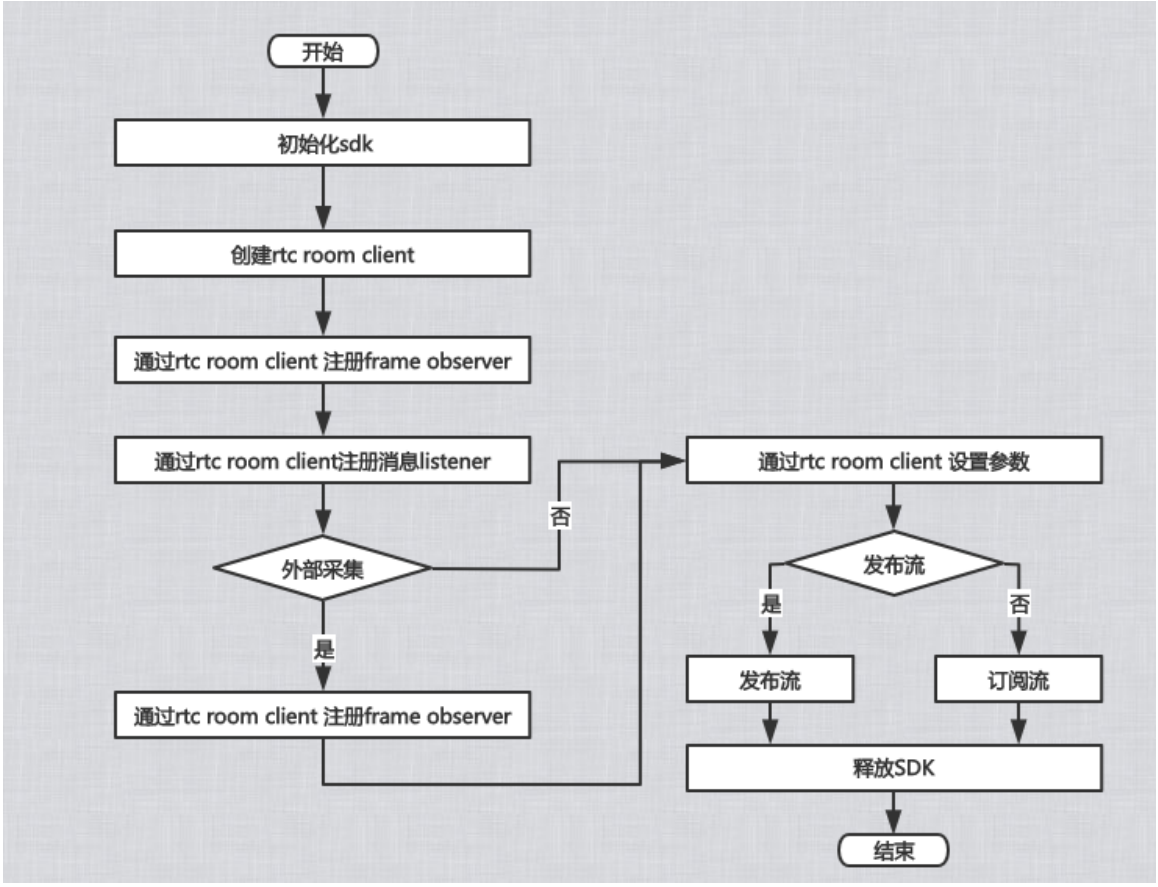
Windows SDK

概述

RTC Windows端SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

API

API调用流程



初始化相关接口 initializeBaiduRtcSdk void initializeBaiduRtcSdk(char *appId, char* token, char* cerPath)

介绍

初始化SDK。

初始化 Windows RTC SDK, 需在使用之前调用进行初始化。

参数

参数	类型	描述
appId	char*	RTC 基础业务单元的唯一标识
token	char*	RTC Server 端鉴权使用的字符串
cerPath	char*	SSL 证书存放路径，证书由百度提供，在SDK文件中

返回

无 **deInitializeBaiduRtcSdk** void deInitializeBaiduRtcSdk()

介绍

释放SDK。

SDK 使用完，可调用该 API 进行释放，销毁操作。

参数

无

返回

无

createBaiduRtcRoomClient std::unique_ptr createBaiduRtcRoomClient()

介绍

创建 BaiduRtcRoomClient。

用户使用该 SDK 开发的入口，首先需要创建 BaiduRtcRoomClient, 通过该 API 初始化 SDK，发布流，订阅流等其他可提供的 API 操作。

参数

无

返回

BaiduRtcRoomClient 指针

createBaiduRtcRoomClient std::unique_ptr createBaiduRtcRoomClient(RtcConfig config)

介绍

创建 BaiduRtcRoomClient。

用户使用该 SDK 开发的入口，首先需要创建 BaiduRtcRoomClient, 通过该 API 初始化 SDK，发布流，订阅流等其他可提供的 API 操作。

参数

参数	类型	描述
config	RtcConfig	配置参数，用于初始化SDK相关配置

返回

BaiduRtcRoomClient 指针

createBaiduRtcRoomClientRef BaiduRtcRoomClient* createBaiduRtcRoomClientRef()

介绍

创建 BaiduRtcRoomClient。

用户使用该 SDK 开发的入口，首先需要创建 BaiduRtcRoomClient, 通过该 API 初始化 SDK，发布流，订阅流等其他可提供的 API 操作。

参数

无

返回

BaiduRtcRoomClient* ，销毁SDK时需调用freeBaiduRtcRoomClient

createBaiduRtcRoomClientRef BaiduRtcRoomClient* createBaiduRtcRoomClientRef(RtcConfig config)

介绍

创建 BaiduRtcRoomClient。

用户使用该 SDK 开发的入口，首先需要创建 BaiduRtcRoomClient, 通过该 API 初始化 SDK，发布流，订阅流等其他可提供的 API 操作。

参数

参数	类型	描述
config	RtcConfig	配置参数，用于初始化SDK相关配置

返回

BaiduRtcRoomClient* ，销毁SDK时需调用freeBaiduRtcRoomClient

freeBaiduRtcRoomClient BaiduRtcRoomClient* createBaiduRtcRoomClientRef(RtcConfig config)

介绍

释放 BaiduRtcRoomClient。与createBaiduRtcRoomClientRef接口对应，在销毁SDK时进行释放。

参数

参数	类型	描述
config	RtcConfig	配置参数，用于初始化SDK相关配置

返回

空

RtcConfig说明 介绍

设置SDK初始化配置

内容

参数	类型	默认值	描述
isOnlyPullStream	bool	false	仅拉视频流模式，专用于视频监控、无人车等场景
h264GopSize	int	3	设置 H264 Gop 大小，用于控制关键帧间隔
isFrameDynamicBind	bool	false	设置是视图绑定模式，为true时允许视图动态添加
isEnableMultistream	bool	true	设置是否使能多流模式，单流模式用于与旧版服务器兼容

房间相关接口 BaiduRtcRoomClient::loginRoom

```
bool BaiduRtcRoomClient::loginRoom(char* roomId, char* userId)
```

介绍

房间登录。

登录房间成功后，用户可以发布流，或订阅流，并能通过IRtcMessageListener接口获取房间状态信息。

默认登录房间后不自动推流，需要调用publishStreaming进行推流。

当设置 setAutoPublish(true) 时会自动推流，可以通过setAutoSubscribe控制是否自动订阅流。

参数

参数	类型	描述
roomId	char*	房间名，长度不超过255字节
userId	char*	用户id, 每个房间的用户id必须唯一

返回

true 成功， false 失败

BaiduRtcRoomClient::logoutRoom

```
bool BaiduRtcRoomClient::loginRoom(char* roomId, char* userId)
```

介绍

房间登出。

关闭媒体通道，关闭信令通道，释放内存资源，及销毁其他申请的资源。

参数

无

返回

true 成功， false 失败

BaiduRtcRoomClient::setForceLogin

```
void BaiduRtcRoomClient::setForceLogin(bool force)
```

介绍

设置是否强制登录，为true时userId相同情况情况下踢出对端用户，为false时userId相同情况登录失败，通过 IRtcErrorListener

回调 RTC_ROOM_USERID_ALREADY_EXIST_ERROR 事件，需业务侧进行提示处理

参数

参数	类型	描述
force	bool	是否使能强制登录

返回

空

发布/订阅流相关接口 BaiduRtcRoomClient::publishStreaming

```
void BaiduRtcRoomClient::publishStreaming(bool isSubsrcibe)
```

介绍

发布流。

流发布在roomId指定的房间，在同一房间加入的用户可以相互订阅流。

参数

参数	类型	描述
isSubscribe	bool	标识在发布流的同时，是否订阅流，默认是订阅流的

返回

无

BaiduRtcRoomClient::publishStreaming

```
void BaiduRtcRoomClient::publishStreaming(char* streamingIds[],int streamingNum)
```

介绍

发布流。

流发布在 roomId 指定的房间，在同一房间加入 的用户可以相互订阅流，该 API 可以指定要订阅的流。

用于提前获取房间内其他用户userId情况下，指定订阅流用户；

参数

参数	类型	描述
streamingIds	char*	用户要订阅的流 id 列表
streamingNum	int	欲订阅的流数目，即streamingIds的大小

返回

无 **BaiduRtcRoomClient::subscribeStreaming**

```
void BaiduRtcRoomClient::subscribeStreaming(char* streamingIds[], int streamingNum)
```

介绍

订阅流。

用于订阅同一房间的其他用户的流。

参数

参数	类型	描述
streamingIds	char*	用户要订阅的其他用户的流id列表(即其他用户id列表)
streamingNum	int	欲订阅的流数目，即streamingIds的大小

返回

无 **BaiduRtcRoomClient::stopPublishStreaming** void BaiduRtcRoomClient::stopPublishStreaming()

介绍

停止发布流。

publishing 的流都可通过该接口停止发布或订阅。

参数

无

返回

无
BaiduRtcRoomClient::stopSubscribeStreaming void BaiduRtcRoomClient::stopSubscribeStreaming(char* streamingIds[], int streamingNum)

介绍

停止订阅流。

订阅的流都可通过该接口停止发布或订阅。

参数

参数	类型	描述
streamingIds	char*	用户要停止订阅的流id列表(即其他用户id列表)
streamingNum	int	欲停止订阅的流数目，即streamingIds的大小

返回

无

BaiduRtcRoomClient::setOnlyPublishVideo void BaiduRtcRoomClient::setOnlyPublishVideo()

介绍

设置仅推视频流，需要在 loginRoom 前调用

参数

无

返回

无

BaiduRtcRoomClient::setOnlyPublishAudio void BaiduRtcRoomClient::setOnlyPublishAudio()

介绍

设置仅推音频流，需要在 loginRoom 前调用

参数

无

返回

无

BaiduRtcRoomClient::setRoomMode void BaiduRtcRoomClient::setRoomMode(RtcRoomMode mode)

介绍

设置房间模式

参数

参数	类型	描述
mode	RtcRoomMode	房间模式

参数类型RtcRoomMode说明：

- RTC_NORMAL_ROOM：普通房间模式，默认为该模式
- RTC_LARGE_ROOM_MODE：大房间模式，SDK不进行房间管理

返回

无

BaiduRtcRoomClient::setRoomMode void BaiduRtcRoomClient::setRoomMode(RtcRoomMode mode)

介绍

设置房间模式

参数

参数	类型	描述
mode	RtcRoomMode	房间模式

参数类型RtcRoomMode说明：

- RTC_NORMAL_ROOM：普通房间模式，默认为该模式
- RTC_LARGE_ROOM_MODE：大房间模式，SDK不进行房间管理

返回

无

BaiduRtcRoomClient::setAutoPublish void BaiduRtcRoomClient::setAutoPublish(bool enable)

介绍

设置是否自动推流

该接口主要用于设置设置是否自动推流，默认为false，为true时login后不需要调用 publishStreaming

参数

参数	类型	描述
enable	bool	是否自动推流

返回

无

BaiduRtcRoomClient::setAutoSubscribe void BaiduRtcRoomClient::setAutoSubscribe(bool enable)

介绍

设置是否自动订阅流

该接口主要用于设置设置是否自动订阅流，默认为true，为false时需手动调用subscribeStreaming 订阅用户流

参数

参数	类型	描述
enable	bool	是否自动订阅流

返回

无

音视频参数设置 BaiduRtcRoomClient::setBitRate void BaiduRtcRoomClient::setBitRate(int bitRate)

介绍

设置视频编码码率。

该接口用于设置视频编码码率，SDK 会根据网络状况做自适应码率调整，但不会超过通过该接口设置的码率即以该值为最大码率。

参数

参数	类型	描述
bitRate	int	需要设置视频码率值，如设置300kbps, 则bitRate = 300*1024

返回

无 **BaiduRtcRoomClient::setResolution** void BaiduRtcRoomClient::setResolution(int width, int height)

介绍

设置视频编码码率。

用于设置视频采集分辨率，若使用外部视频采集，就无需设置。

参数

参数	类型	描述
width	int	视频采集宽
height	int	视频采集高

返回

无 **BaiduRtcRoomClient::setFps** void BaiduRtcRoomClient::setFps(int fps)

介绍

设置视频采集帧率。

用于设置视频采集分辨率，若使用外部视频采集，就无需设置。

参数

参数	类型	描述
fps	int	视频采集的帧率，如15fps, 24fps, 29fps等

返回

无 **BaiduRtcRoomClient::setVideoFromat** void BaiduRtcRoomClient::setVideoFromat(VideoColorFormat vcf)

介绍

设置视频颜色空间格式。

用于设置视频颜色空间格式。

参数

参数	类型	描述
vcf	VideoColorFormat	支持的视频颜色空间格式ARGB/I420

返回

无

摄像头相关接口 BaiduRtcRoomClient::openCamera void BaiduRtcRoomClient::openCamera(char* deviceName)

介绍

打开摄像头。

打开摄像头后，可开始画面预览。

参数

参数	类型	描述
deviceName	char*	设备名称，若为空，开启默认的设备，否则开启指定的设备

返回

无 **BaiduRtcRoomClient::closeCamera** void BaiduRtcRoomClient::closeCamera()

介绍

关闭摄像头。

关闭并释放摄像头资源。

参数

无

返回

无 **BaiduRtcRoomClient::muteCamera** void BaiduRtcRoomClient::muteCamera(bool isMuted)

介绍

开关摄像头。

静默摄像头，停止画面预览及传输。

参数

参数	类型	描述
isMuted	bool	是否静默摄像头

返回

无

音频相关接口 BaiduRtcRoomClient::muteMicphone void BaiduRtcRoomClient::muteMicphone(bool isMuted)

介绍

开关麦克风。

静默麦克风，停止声音传输。

参数

参数	类型	描述
isMuted	bool	是否静默麦克风

返回

无

BaiduRtcRoomClient::muteSpeaker void BaiduRtcRoomClient::muteSpeaker(bool isMuted)

介绍

开关扬声器。

静默扬声器，停止声音播放。

参数

参数	类型	描述
isMuted	bool	是否静默扬声器

返回

无

BaiduRtcRoomClient::setVolume void BaiduRtcRoomClient::setVolume(uint32_t volume)

介绍

音量控制。

用于控制扬声器音量的大小。

参数

参数	类型	描述
volume	uint32_t	音量值，范围 0~100

返回

无

BaiduRtcRoomClient::setUserPlaybackVolume void BaiduRtcRoomClient::setUserPlaybackVolume(long long userId, int volume)

介绍

设置远端用户音量，在订阅远端用户流完成后

参数

参数	类型	描述
userId	long long	远端用户userId
volume	int	远端用户音量，范围 0~100

返回

无

视频相关接口 BaiduRtcRoomClient::startPreview void BaiduRtcRoomClient::startPreview()

介绍

开始预览。

打开摄像头后，可开始画面预览。

参数

无

返回

无 **BaiduRtcRoomClient::stopPreview** void BaiduRtcRoomClient::stopPreview()

介绍

停止预览。

在开始预览后，可通过该接口停止预览。

参数

无

返回

无

BaiduRtcRoomClient::setVideoMirror void BaiduRtcRoomClient::setVideoFromat(bool isOpenMirror)

介绍

设置镜像。

用于设置是否打开镜像功能。

参数

参数	类型	描述
isOpenMirror	bool	设置是否打开镜像功能

返回

无

BaiduRtcRoomClient::registerVideoFrameObserver void BaiduRtcRoomClient::registerVideoFrameObserver(IVideoFrameObserver* iVfo[], int iVfoNum)

介绍

注册视频数据帧到达Observer。

当使用外部渲染时，需要注册视频数据监测接口，有视频数据到来时，可通知用户去做渲染或其他处理。接口详细信息可查看头文件：BaiduExternalVideoRendererInterface.h

参数

参数	类型	描述
iVfo	IVideoFrameObserver*	数据帧到达Observer接口数组，多个接口可多路流，一个接口listening一路流
iVfoNum	int	Observer 接口数

返回

无

屏幕共享相关接口 BaiduRtcRoomClient::startShareScreen void BaiduRtcRoomClient::startShareScreen()

介绍

屏幕共享。

可通过该接口开启屏幕分享功能, 让其他用户看到你的屏幕操作。

参数

无

返回

无 **BaiduRtcRoomClient::stopShareScreen** void BaiduRtcRoomClient::stopShareScreen()

介绍

停止屏幕共享。

关闭屏幕分享功能。

参数

无

返回

无 **BaiduRtcRoomClient::startShareScreenWithSource** void BaiduRtcRoomClient::startShareScreenWithSource(long long sourceId)

介绍

可通过该接口开启屏幕分享功能, 让其他用户看到你的屏幕操作，通过sourceId指定分享的屏幕。

参数

参数	类型	描述
sourceId	long long	屏幕ID，可通过getWindowSourceList获取，或者外部传入

返回

无

BaiduRtcRoomClient::getWindowSourceList void BaiduRtcRoomClient::getWindowSourceList(CaptureType type, int& count)

介绍

可通过该接口获取可共享屏幕/窗口列表

参数

参数	类型	描述
type	CaptureType	类型，桌面或者窗口
count	int&	列表长度

返回

无 **BaiduRtcRoomClient::freeWindowSourceList** void BaiduRtcRoomClient::freeWindowSourceList(WindowSource* sourceList, int length)

介绍

释放共享屏幕/窗口列表

参数

参数	类型	描述
sourceList	WindowSource*	列表指针
length	int	列表长度

返回

无 **BaiduRtcRoomClient::startShareWindow** void BaiduRtcRoomClient::startShareWindow(long long sourceId)

介绍

可通过该接口开启窗口分享功能, 让其他用户看到你的窗口操作，通过sourceId指定分享的窗口。

参数

参数	类型	描述
sourceId	long long	窗口Wid,通过getWindowSourceList获取,或者外部获取传入

返回

无

BaiduRtcRoomClient::setShareScreenMode void BaiduRtcRoomClient::setShareScreenMode(ShareScreenMode type)

介绍

设置屏幕共享模式

可通过该接口设置屏幕共享模式, 可以同时回调屏幕数据与摄像头采集数据

参数

参数	类型	描述
type	ShareScreenMode	屏幕共享模式

参数类型ShareScreenMode说明：

- kOnlyScreen：仅回调屏幕数据
- kWithPreviewVideo：回调屏幕数据与摄像头数据

返回

无

BaiduRtcRoomClient::stopShareWindow void BaiduRtcRoomClient::stopShareWindow()

介绍

关闭窗口分享功能

参数

无

返回

无

消息接口 BaiduRtcRoomClient::sendMessage void BaiduRtcRoomClient::sendMessage(const char* message)

介绍

发送消息。

可通过该接口发送文本消息到聊天室，房间的其他人能接收到所发消息。

参数

参数	类型	描述
message	const char*	准备发的消息

返回

无

BaiduRtcRoomClient::sendMessageToUser void BaiduRtcRoomClient::sendMessageToUser(const char* message, long userId)

介绍

发送消息。

通过该接口发送消息给其他人，userId=0 时房间内广播消息，其他指定用户发送消息

参数

参数	类型	描述
message	const char*	准备发的消息
userId	long	用户UserId，为0时广播消息

返回

无

BaiduRtcRoomClient::registerRtcMessageListener void BaiduRtcRoomClient::registerRtcMessageListener(IRtcMessageListener* msgListener)

介绍

注册消息监听接口。

用户可以实现参数定义的接口，并通过该接口注册，当SDK启动后，可以收到来自SDK的消息，并做相应的处理。接口详细信息可查看头文件：BaiduRtcCommonDefine.h

参数

参数	类型	描述
msgListener	IRtcMessageListener*	rtc sdk 消息上报接口

返回

无

状态获取接口 BaiduRtcRoomClient::getRoomStates void BaiduRtcRoomClient::getRoomStates(RtcRoomUserInfo** userInfoList, int& userNumber)

介绍

查询房间信息。

可通过该接口查询房间用户信息。

参数

参数	类型	描述
userInfoList	RtcRoomUserInfo**	准备发的消息
userNumber	int&	用户id

返回

无 **BaiduRtcRoomClient::getConnectionStats** ConnectionStats BaiduRtcRoomClient::getConnectionStats(ConnectionType connectionType)

介绍

获取当前网络状态。

根据连接类型，返回用户当前的网络连接状态值。

参数

参数	类型	描述
connectionType	ConnectionType	网络连接类型，信令通道连接/媒体通道连接

返回

ConnectionStats 网络连接状态值

转推相关接口 可将当前用户RTP 流转推为 RTMP 流，用于互娱、教育等场景 **BaiduRtcRoomClient::configLiveStreamWithUrl** void BaiduRtcRoomClient::getRoomStates(char *url, bool isMix, bool isRecording, char* mixTemplate, RtcLiveTransferMode transferMode)

介绍

配置Server端推流参数

接口用于配置Server推流的参数，在登录前调用。

聊天室模式：在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间。

主播转推模式：主播推向不同的直播房间，进入房间前调用。

参数

参数	类型	描述
url	char*	转推RTMP地址
isMix	bool	是否混流
isRecording	bool	是否录制
mixTemplate	char*	转推RTMP模板
transferMode	RtcLiveTransferMode	转推模式

返回

无

BaiduRtcRoomClient::startLiveServerStreaming void BaiduRtcRoomClient::startLiveServerStreaming(char *url, bool isMix, bool isRecording, char* mixTemplate, RtcLiveTransferMode transferMode)

介绍

开启server端推流

接口开启server推流的参数，在推流成功后调用。

聊天室模式：在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间。

主播转推模式：主播推向不同的直播房间，进入房间前调用。

参数

参数	类型	描述
url	char*	转推RTMP地址
isMix	bool	是否混流
isRecording	bool	是否录制
mixTemplate	char*	转推RTMP模板
transferMode	RtcLiveTransferMode	转推模式

返回

无

BaiduRtcRoomClient::stopLiveServerStreaming void BaiduRtcRoomClient::stopLiveServerStreaming(RtcLiveTransferMode transferMode)

介绍

停止Server端推流

聊天室模式：在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间。

主播转推模式：主播推向不同的直播房间，进入房间前调用。

参数

参数	类型	描述
transferMode	RtcLiveTransferMode	转推模式

返回

无

转推相关接口 声卡采集相关接口，可用于麦克风与系统音频混音输出

BaiduRtcRoomClient::enableLoopbackRecording void BaiduRtcRoomClient::enableLoopbackRecording(bool enabled)

介绍

控制声卡采集开启，在推流成功后调用

参数

参数	类型	描述
enabled	bool	为true时开启，为false时关闭

返回

无

BaiduRtcRoomClient::setLoopbackRecordingVolume void BaiduRtcRoomClient::setLoopbackRecordingVolume(int volume)

介绍

设置声卡采集音量

参数

参数	类型	描述
volume	int	声卡采集音量，范围 0~100

返回

无

BaiduRtcRoomClient::setLoopbackRecordingMute void BaiduRtcRoomClient::setLoopbackRecordingMute(bool mute)

介绍

设置声卡采集静音

参数

参数	类型	描述
mute	bool	是否静音

返回

无

其他接口 BaiduRtcRoomClient::enableRtcStatsInfoDot void BaiduRtcRoomClient::enableRtcStatsInfoDot(bool isEnable)

介绍

RTC状态信息打点。

是否开启 RTC 状态信息打点功能，默认开启。当开启打点时，日志信息会上传到百度日志统计Server,并通过图表的形式展示，可实时查询。

参数

参数	类型	描述
isEnable	bool	是否开启打点

返回

ConnectionStats 网络连接状态值

getBaiduRtcSdkVersion void getBaiduRtcSdkVersion(char* version, int len)

介绍

获取SDK版本号。

获取百度 RTC SDK 版本号。 **参数**

参数	类型	描述
version	char*	版本号字符串
int	len	字符串长度

返回

无 **enableBaiduRtcLog** void enableBaiduRtcLog(bool isEnable)

介绍

日志功能。

日志功能开关, 谨记在销毁 SDK 时需关闭。

参数

参数	类型	描述
isEnable	bool	若为 true, 开启日志功能， 开启日志功能后，会把日志保存到本地文件中；若为false 关闭日志功能。

返回

无 **configStreamingAddr** void configStreamingAddr(const char* url)

介绍

切换媒接入Server。

当链路不稳定，或视频画面卡顿时，可使用该接口，切换媒体接入Server. 切换后，需要重新进入房间，并发布流或拉流。

参数

参数	类型	描述
url	const char*	媒体服务器地址

返回

无

setVideoDevice void setVideoDevice(const char* deviceId)

介绍

设置视频设备。

用户通过getVideoDeviceList 获取的设备信息列表，设置准备开启的视频设备。

参数

参数	类型	描述
deviceId	const char*	视频设备 ID 或 NAME

返回

无

getVideoDeviceList RtcVideoDeviceInfo* getVideoDeviceList(int & deviceCount)

介绍

获取视频设备列表。

获取当前可用的视频设备列表，用户可根据获取的列表，选择要开启的视频设备。

参数

参数	类型	描述
deviceCount	int&	可用设备数目

返回

RtcVideoDeviceInfo *指向当前可用信息列表* ***freeVideoDeviceList*** *void freeVideoDeviceList(RtcVideoDeviceInfo* deviceInfoList)

介绍

释放视频设备列表。

当用户调用 getVideoDeviceList 获取视频设备列表后，须使用改接口是否释放设备列表资源。

参数

参数	类型	描述
deviceInfoList	RtcVideoDeviceInfo*	当前可用信息列表

返回

无

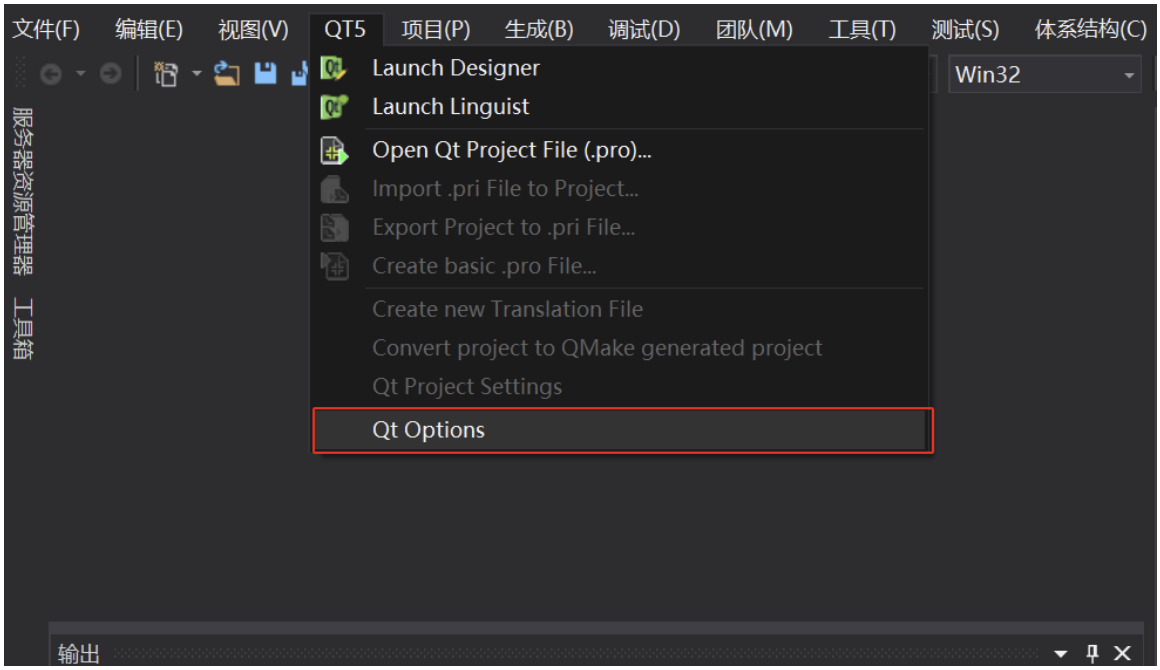
开发环境搭建

本demo使用QT设计开发UI交互，QT+VS环境搭建如下：

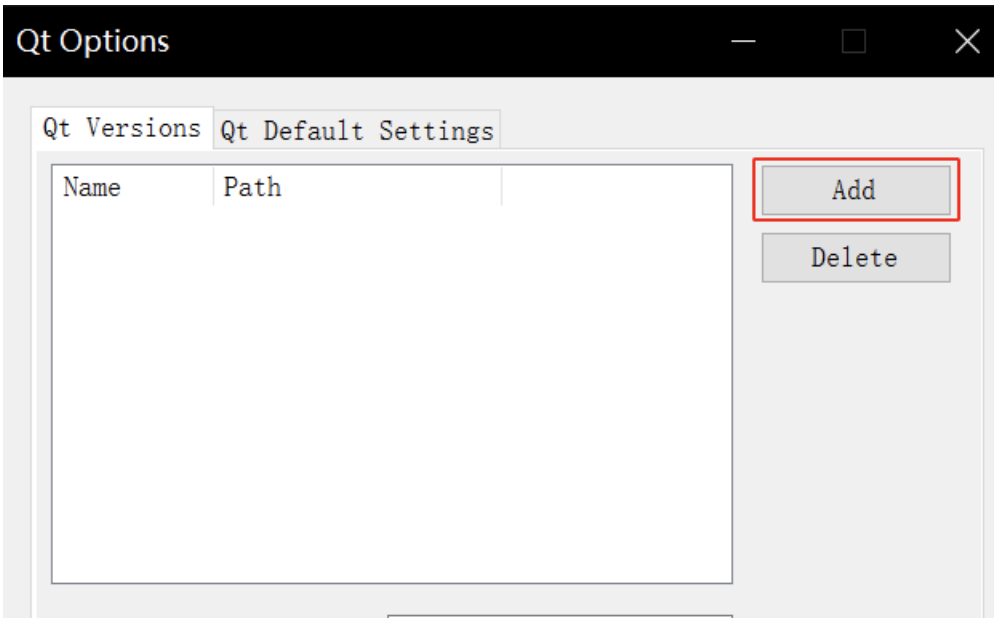
1. 下载QT开发QT5.9

2. 下载qt-vs-addin-1.2.5.exe， 该插件会将Qt开发工具集成到vs上；

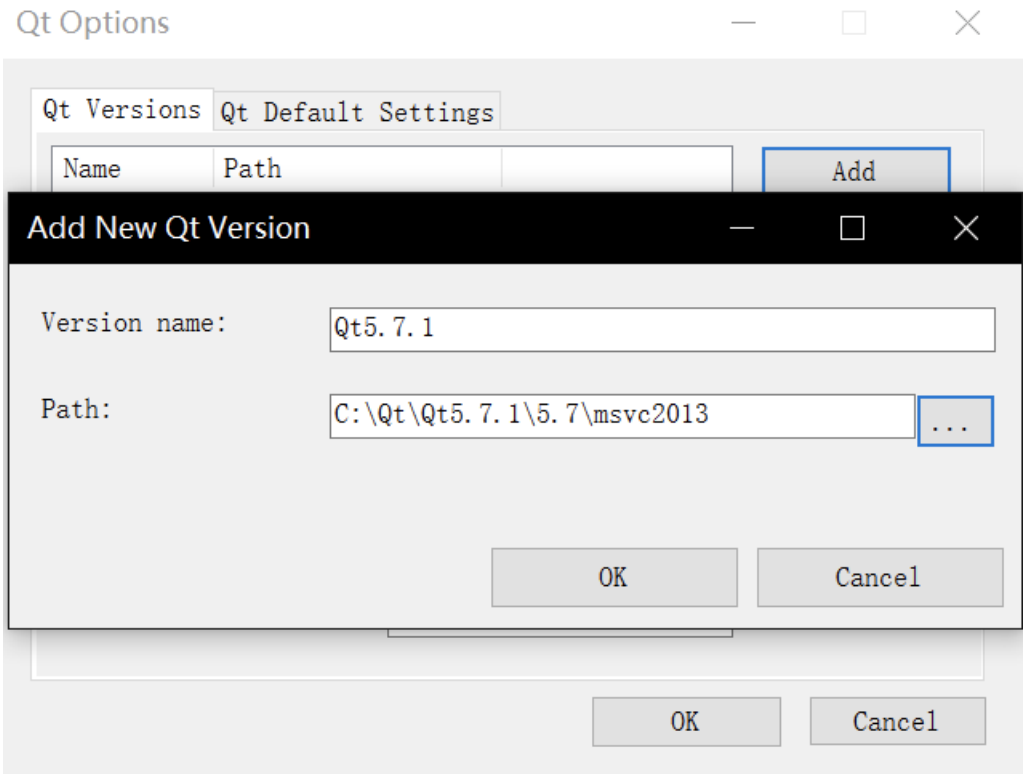
3. 进入vs工程，点击菜单栏"QT5" 或 “Qt Vs Tools”，选择"Qt Options"：



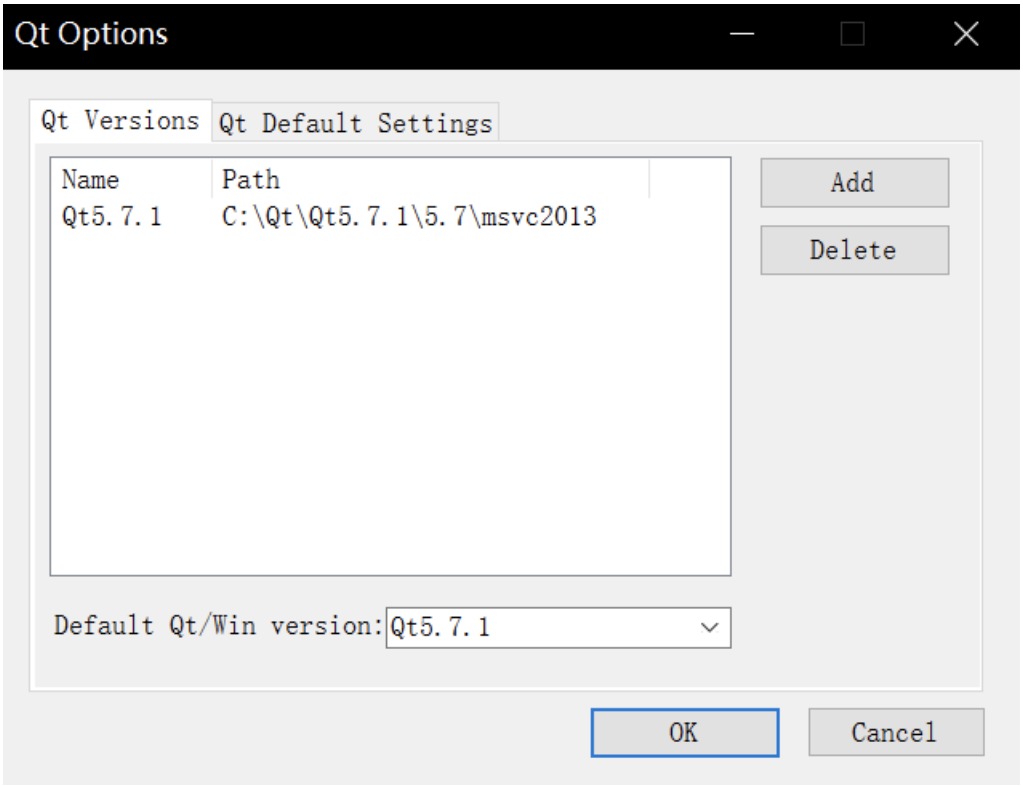
4. 点击Add，添加本地Qt存放目录路径：



5. 根据自己的QT目录情况，按照下图示例设置正确路径：



6. 点击OK，Qt环境配置完毕：



API（c#）参考

初始化相关接口

brtc_create_client void* brtc_create_client()

介绍

通过该接口创建 RTC client handle，开发者可通过该接口返回的handle管理使用RTC SDK

参数

无

返回

返回 RTC client handle

brtc_set_rtc_config void brtc_set_rtc_config(void *rtc_client, RtcConfig config)

介绍

开发者可通过该接口配置RTC SDK

参数

参数	类型	说明
rtc_client	void*	RTC handle
config	RtcConfig	具体配置

参数类型RtcConfig定义：

类型	属性	默认值	取值范围	说明
bool	isOnlyPullStream	false	可取值true，false	是否仅拉视频流，用于监控等场景
int	h264GopSize	3	取值大于0	h264编码gop值设置
bool	isFrameDynamicBind	false	可取值true，false	是否动态创建视图
bool	isEnableMultistream	true	可取值true，false	是否使能多流模式

返回

空

brtc_destroy_client void brtc_destroy_client(void *rtc_client)

介绍

开发者可通过该接口删除 brtc_create_client 创建的 RTC client hanle

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_init_sdk void brtc_init_sdk(void *rtc_client, char* app_id, char *token, char* cer_path)

介绍

初始化SDK

参数

参数	类型	说明
rtc_client	void*	rtc handle
app_id	char*	百度为App签发的 App ID, 用于识别应用, 全局唯一，详见创建应用
token	char*	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。 详见应用鉴权
cer_path	char*	证书路径，证书文件在SDK文件中放置

返回

空

brtc_delnit_sdk void brtc_delnit_sdk(void *rtc_client)

介绍

在SDK使用完后，可调用该函数进行释放

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

房间相关接口

brtc_set_server_url void brtc_set_server_url(void *rtc_client, char* server_url)

介绍

设置server地址

参数

参数	类型	说明
rtc_client	void*	RTC handle
server_url	char*	server地址

返回

空

brtc_login_room void brtc_login_room(void *rtc_client, char* room_name, char *user_id, char* token)

介绍

登录房间

登录房间成功后，用户可以发布流，或订阅流，并能通过brtc_set_message_arrival_callback接口获取房间内状态事件

参数

参数	类型	取值范围	说明
rtc_client	void*	无	RTC handle
room_name	char*	任意字符串，长度不超过255字节	房间名
user_id	char*	大于0的数字类型字符串	用户ID, 每个房间的用户ID必须唯一
token	char*	字符串	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。 详见应用鉴权

返回

空

brtc_logout_room void brtc_logout_room(void *rtc_client)

介绍

退出房间

关闭媒体通道，关闭信令通道，释放内存资源，及销毁其他申请的资源

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_publish_streaming void brtc_publish_streaming(void *rtcClient, bool is_auto_subscribe_stream)

介绍

发布流

流发布在指定的房间，在同一房间加入的用户可以相互订阅流，默认在发布流的同时，订阅在该房间其他用户的流；

参数

类型	参数	默认值	取值范围
void*	rtc_client	无	无
bool	is_auto_subscribe_stream	true	取值true，false

返回

空

brtc_publish_streaming_with_stream_id void brtc_publish_streaming_with_stream_id(void *rtcClient, char* streamingIds[], int streamingNum)

介绍

发布流

发布在指定的房间，在同一房间加入的用户可以相互订阅流；该API可以指定要订阅的具体多条流或多个用户的流。

用于在预先知道远端用户UserId情况下指定订阅流，不订阅指定UserId外的其他流。

参数

参数	类型	说明
rtc_client	void*	RTC handle
streamingIds	char*[]	远端用户UserId列表
streamingNum	int	订阅用户数目

返回

空

brtc_stop_publish_streaming void brtc_stop_publish_streaming(void *rtc_client)

介绍

停止发布流

可通过该接口停止发布或订阅

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_subscribe_streaming void brtc_subscribe_streaming(void *rtc_client, char* stream_id)

介绍

订阅流

用于订阅同一房间的其他用户的流

参数

参数	类型	取值范围	说明
rtc_client	void*	无	RTC handle
stream_id	char*	大于0的数字类型字符串	远端用户UserId

返回

空

brtc_stop_subscribe_streaming void brtc_stop_subscribe_streaming(void *rtc_client, char* stream_id)

介绍

停止订阅流

订阅的流可通过该接口停止订阅

参数

参数	类型	取值范围
rtc_client	void*	无
stream_id	char*	大于0的数字类型字符串

返回

空

brtc_set_only_publish_audio void brtc_set_only_publish_audio(void *rtc_client)

介绍

设置只发布音频流

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

设备参数相关接口

brtc_mute_camera void brtc_mute_camera(void *rtc_client, bool is_mute)

介绍

关闭摄像头，停止画面预览及传输

参数

参数	类型	说明
rtc_client	void*	RTC handle
is_mute	bool	是否关闭摄像头

返回

空

brtc_mute_micphone void brtc_mute_micphone(void *rtc_client, bool is_mute)

介绍

关闭麦克风，停止声音采集及传输

参数

参数	类型	说明
rtc_client	void*	RTC handle
is_mute	bool	是否关闭麦克风

返回

空

brtc_set_video_bitrate void brtc_set_video_bitrate(void *rtc_client, int bit_rate)

介绍

设置视频码率

设置视频编码码率，该码率是最大码率，会随着网络变化做自适应的变化

参数

参数	类型	取值范围	说明
rtc_client	void*	无	RTC handle
bit_rate	int	大于0	视频码率，如想设置300kbps, 则bit_rate = 300*1024

返回

空

brtc_set_video_fps void brtc_set_video_fps(void *rtc_client, int fps)

介绍

设置视频采集fps

参数

参数	类型	取值范围	说明
rtc_client	void*	无	RTC handle
fps	int	0 < fps <= 60	视频采集fps

返回

空

brtc_set_video_resolution void brtc_set_video_resolution(void *rtc_client, int width, int height)

介绍

设置视频分辨率

视频分辨率会随着网络状态自适应变化

参数

参数	类型	取值范围	说明
rtc_client	void*	RTC handle	
width	int	大于0	视频宽度
height	int	大于0	视频高度

返回

空 **brtc_set_video_mirror** void brtc_set_video_mirror(void *rtc_client, bool is_mirror)

介绍

设置本地视频镜像显示

参数

参数	类型	说明
rtc_client	void*	RTC handle
is_mirror	bool	是否开启镜像显示

返回

空

brtc_start_preview void brtc_start_preview(void *rtc_client)

介绍

打开摄像头后，可开始画面预览

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_stop_preview void brtc_stop_preview(void *rtc_client)

介绍

在开始预览后，可通过该接口停止预览

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_start_mic_recording void brtc_start_mic_recording(void *rtc_client)

介绍

开始录制音频，提供于插拔麦克风后，启动采集，默认会自动开始采集

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

设备管理相关接口

brtc_get_audio_record_device_num int brtc_get_audio_record_device_num(void *rtc_client)

介绍

通过该接口获取音频采集设备数目

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_get_audio_playout_device_num int brtc_get_audio_playout_device_num(void *rtc_client)

介绍

通过该接口获取音频输出设备数目

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_get_audio_record_devices void brtc_get_audio_record_devices(void *rtc_client, char* devNameList, int listLen, int maxNameLen)

介绍

通过该接口获取音频采集设备列表，列表中存储的是设备名

参数

参数	类型	说明
rtc_client	void*	RTC handle
devNameList	char *	存储具体的采集设备名
listLen	int	设备列表长度
maxNameLen	int	每个设备名的最大长度

返回

空

brtc_get_audio_playout_devices void brtc_get_audio_playout_devices(void *rtc_client, char* devNameList, int listLen, int maxNameLen)

介绍

通过该接口获取音频输出设备列表，列表中存储的是设备名

参数

参数	类型	说明
rtc_client	void*	RTC handle
devNameList	char*	存储具体的音频输出设备名
listLen	int	设备列表长度
maxNameLen	int	每个设备名的最大长度

返回

空

brtc_set_auido_record_device void brtc_set_auido_record_device(void *rtc_client, char* devName)

介绍

通过设备名，设置选取的音频采集设备

参数

参数	类型	说明
rtc_client	void*	RTC handle
devName	char *	欲设置的设备名

返回

空

brtc_set_auido_playout_device void brtc_set_auido_playout_device(void *rtc_client, char* devName)

介绍

通过设备名，设置选取的音频播放输出设备

参数

参数	类型	说明
rtc_client	void*	RTC handle
devName	char*	欲设置的设备名

返回

空

回调相关接口

brtc_set_message_arrival_callback void brtc_set_message_arrival_callback(void *rtc_client, MessageArrivalCallback msg_call_back)

介绍

设置消息到达回调

设置消息接收接口，当有消息到达时，通过msg_call_back上报应用层

参数

参数	类型	说明
rtc_client	void*	RTC handle
msg_call_back	MessageArrivalCallback	需要应用层设置的callback

返回

空

brtc_set_event_arrival_callback void brtc_set_event_arrival_callback(void *rtc_client, EventArrivalCallback event_call_back)

介绍

设置事件到达回调

设置事件接收接口，当有事件到达时，通过event_call_back上报应用层

参数

参数	类型	说明
rtc_client	void*	RTC handle
event_call_back	EventArrivalCallback	需要应用层设置的callback

返回

空 **brtc_set_video_arrival_callback** void brtc_set_video_arrival_callback(void *rtc_client, VideoArrivalCallback videoArrivalCb, int observerId)

介绍

设置视频帧到达回调

设置视频到达C#接口，当有视频帧到达时，通过videoArrivalCb C#上报应用层， 应用层可对视频进行处理渲染等

参数

参数	类型	说明
rtc_client	void*	RTC handle
videoArrivalCb	VideoArrivalCallback	需要应用层设置的callback
observerId	int	视频帧id，应用可以自己设定，便于底层SDK识别observer

返回

空

brtc_set_preview_video_arrival_callback void brtc_set_preview_video_arrival_callback(void *rtc_client, VideoArrivalCallback videoArrivalCb, int observerId)

介绍

设置预览视频帧到达回调

设置视频到达C#接口，当有视频帧到达时，通过videoArrivalCb C#上报应用层， 应用层可对视频进行处理渲染等

参数

参数	类型	说明
rtc_client	void*	RTC handle
videoArrivalCb	VideoArrivalCallback	需要应用层设置的callback
observerId	int	视频帧观察着id，应用可以自己设定，便于底层SDK识别observer

返回

空

其他接口

brtc_send_message void brtc_send_message(void *rtc_client, const char* message)

介绍

发送消息

可通过该接口发送文本消息到聊天室，房间的其他人能接收到所发消息

参数

参数	类型	说明
rtc_client	void*	RTC handle
message	const char*	准备发的消息

返回

空

brtc_get_sdk_version char *brtc_get_sdk_version(void* rtc_client)

介绍

获取SDK版本号

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

SDK 版本号

brtc_enable_rtc_stats_info_dot void brtc_enable_rtc_stats_info_dot(void *rtc_client, bool isEnable)

介绍

开启/关闭RTC信息打点

该接口用于控制RTC状态信息统计，并上传到Server

参数

参数	类型	说明
rtc_client	void*	RTC handle
isEnable	bool	为true时开启信息打点统计

返回

空

brtc_set_log_file void brtc_set_log_file(void *rtc_client, char* filePath)

介绍

设置日志输出路径，默认在exe所在目录下

参数

参数	类型	说明
rtc_client	void*	RTC handle
filePath	char*	日志文件路径

返回

空

brtc_enable_rtc_logger void brtc_enable_rtc_logger(void *rtc_client, bool isEnable)

介绍

开启/关闭日志功能

用于打开或关闭日志功能，若打开会在exe目录下保存RTC运行日志文件，若关闭将不进行日志文件保存。

参数

参数	类型	说明
rtc_client	void*	RTC handle
isEnable	bool	是否开启

返回

空

brtc_get_room_id unsigned long long brtc_get_room_id(void *rtc_client)

介绍

获取当前房间号

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

当前房间roomId

屏幕/窗口分享相关接口

brtc_start_share_screen void brtc_start_share_screen(void *rtc_client)

介绍

可通过该接口开启屏幕分享功能, 让其他用户看到你的屏幕操作,默认多屏幕下画面自动合并

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_start_share_screen_with_source void brtc_start_share_screen_with_source(void *rtc_client, long long sourceId)

介绍

可通过该接口开启屏幕分享功能, 让其他用户看到你的屏幕操作;通过sourceId指定分享的屏幕。

可通过brtc_get_window_source_list 接口获取sourceId。

参数

参数	类型	说明
rtc_client	void*	RTC handle
sourceId	long long	屏幕序号，如0，1，2

返回

空

brtc_stop_share_screen void brtc_stop_share_screen(void *rtc_client)

介绍

关闭屏幕分享功能

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_get_window_source_list void *brtc_get_window_source_list(void* rtc_client, CaptureType type, int& length)

介绍

通过该接口获取可共享屏幕/窗口列表

参数

参数	类型	说明
rtc_client	void*	RTC handle
type	CaptureType	获取类型
length	int&	返回列表长度

CaptureType类型说明：

- kScreen：屏幕采集
- kWindow：窗口采集

返回

返回窗口列表，可参考c# Demo工程解析

brtc_set_share_screen_mode void brtc_set_share_screen_mode(void *rtc_client, ShareScreenMode type)

介绍

设置屏幕共享模式

参数

参数	类型	说明
rtc_client	void*	RTC handle
type	ShareScreenMode	共享类型

ShareScreenMode说明：

- kOnlyScreen：仅屏幕采集
- kWithPreviewVideo：屏幕采集同时，进行摄像头采集，可通过brtc_set_preview_video_arrival_callback接口可获取摄像头数据

返回

空

brtc_free_window_source_list void brtc_free_window_source_list(void *rtc_client, void* sourceList, int length)

介绍

释放共享窗口列表

参数

参数	类型	说明
rtc_client	void*	RTC handle
sourceList	void*	要释放窗口列表
length	int	列表长度

返回

空

brtc_start_share_window void brtc_start_share_window(void *rtc_client, long long sourceId)

介绍

可通过该接口开启窗口分享功能，让其他用户看到你的窗口操作，通过sourceId指定分享的窗口。

参数

参数	类型	说明
rtc_client	void*	RTC handle
sourceId	long long	窗口WinId，通过getWindowSourceList获取，或者外部获取后传入

返回

空

brtc_stop_share_window void brtc_stop_share_window(void *rtc_client)

介绍

关闭窗口分享功能

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

声卡采集相关接口

brtc_enable_loopback_recording void brtc_enable_loopback_recording(void *rtc_client, bool enabled)

介绍

开启声卡采集

参数

参数	类型	说明
rtc_client	void*	RTC handle

返回

空

brtc_set_loopback_recording_volume void brtc_set_loopback_recording_volume(void *rtc_client, int volume)

介绍

设置声卡采集音量

参数

参数	类型	取值范围	说明
rtc_client	void*	无	RTC handle
volume	int	0-100	声卡采集音量

返回

空

brtc_set_loopback_recording_mute void brtc_set_loopback_recording_mute(void *rtc_client, bool mute)

介绍

设置声卡采集静音

参数

参数	类型	说明
rtc_client	void*	RTC handle
mute	bool	是否设置采集静音

返回

空

转推相关接口 可将当前用户RTP 流转推为 RTMP 流，用于互娱、教育等场景

brtc_config_live_stream_with_url void brtc_config_live_stream_with_url(void *rtc_client, char* url, bool is_mix, bool is_recording, char* mix_template, RtcLiveTransferMode transfer_mode)

介绍

该接口用于配置server推流的参数；

参数

参数	类型	说明
rtc_client	void*	RTC handle
url	char*	转推RTMP地址
is_mix	bool	是否混流
is_recording	bool	是否录制
mix_template	char*	转推RTMP模板
transfer_mode	RtcLiveTransferMode	转推模式

RtcLiveTransferMode说明：

- RTC_LIVE_TRANSFER_MODE_ROOM_TRANSMISSION：聊天室模式，在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间
- RTC_LIVE_TRANSFER_MODE_ANCHOR_TRANSMISSION：主播转推模式，主播推向不同的直播房间;进入房间前调用;

返回

空

brtc_start_live_server_streaming void brtc_start_live_server_streaming(void *rtc_client, char* url, bool is_mix, bool is_recording, char* mix_template, RtcLiveTransferMode transfer_mode)

介绍

启动动态转推功能，进入房间成功后调用。

参数

参数	类型	说明
rtc_client	void*	RTC handle
url	char*	转推RTMP地址
is_mix	bool	是否混流
is_recording	bool	是否录制
mix_template	char*	转推RTMP模板
transfer_mode	RtcLiveTransferMode	转推模式

RtcLiveTransferMode说明：

- RTC_LIVE_TRANSFER_MODE_ROOM_TRANSMISSION：聊天室模式，在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间
- RTC_LIVE_TRANSFER_MODE_ANCHOR_TRANSMISSION：主播转推模式，主播推向不同的直播房间，进入房间前调用;

返回

空

brtc_stop_live_server_streaming void brtc_stop_live_server_streaming(void *rtc_client, RtcLiveTransferMode transfer_mode)

介绍

停止动态转推功能

参数

参数	类型	说明
rtc_client	void*	RTC handle
transfer_mode	RtcLiveTransferMode	转推模式

RtcLiveTransferMode说明：

- RTC_LIVE_TRANSFER_MODE_ROOM_TRANSMISSION：聊天室模式，在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间
- RTC_LIVE_TRANSFER_MODE_ANCHOR_TRANSMISSION：主播转推模式，主播推向不同的直播房间;进入房间前调用;

返回

空

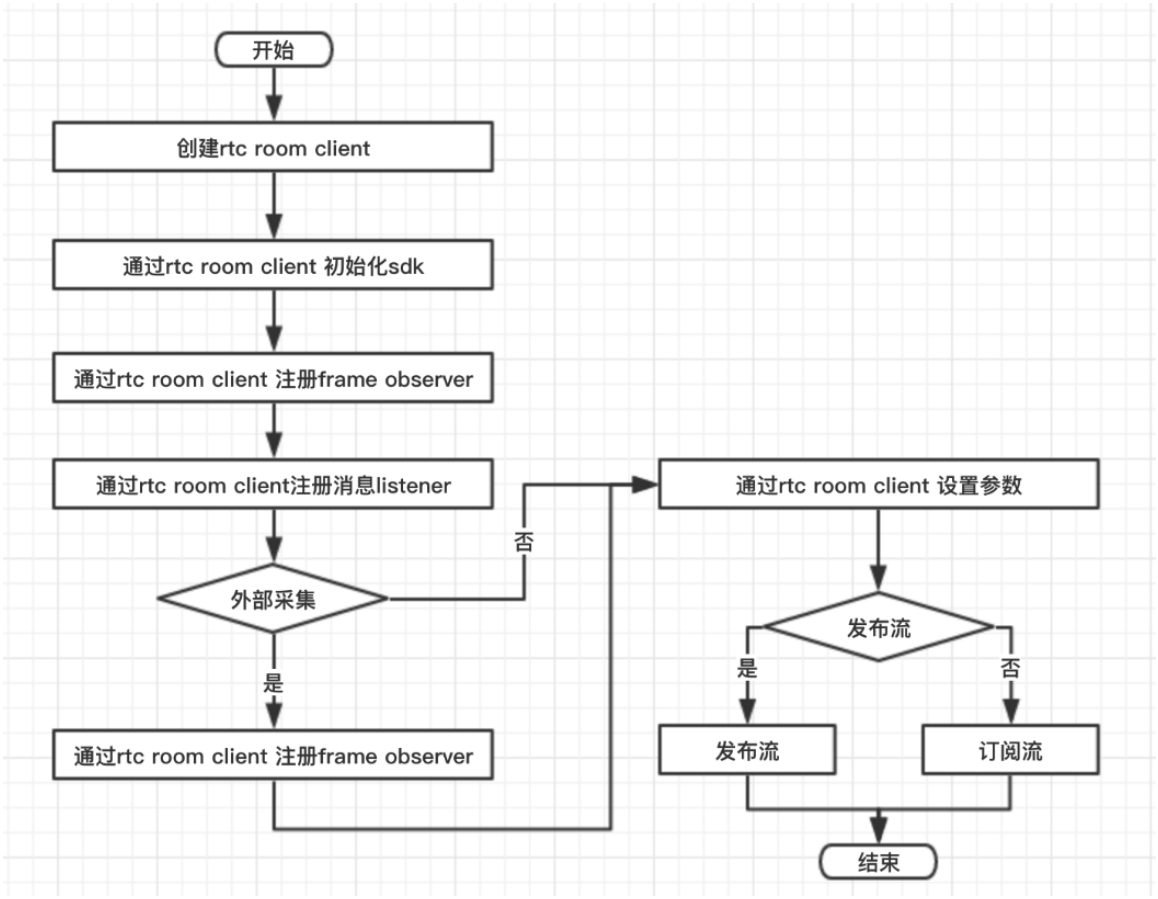
Linux SDK

概述

RTC Linux端SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

API

API调用流程



初始化相关接口

createBaiduRtcRoomClient BaiduRtcRoomClient* createBaiduRtcRoomClient();

介绍

创建 BaiduRtcRoomClient。

用户使用该 sdk 开发的入口，首先需要创建 BaiduRtcRoomClient, 通过该 api 初始化 sdk，发布流，订阅流等其他可提供的 api 操作。

参数

无

返回

BaiduRtcRoomClient 对象

BaiduRtcRoomClient::setAppID

```
void setAppID(const char *AppID);
```

介绍

设置AppID。

设置应用的AppID， 需要从百度智能云申请得到。

参数

参数	类型	描述
AppID	char*	应用的AppID

返回

无

BaiduRtcRoomClient::setMediaServerURL

```
void setMediaServerURL(const char * MediaServerURL);
```

介绍

设置BRTC服务器的URL。

设置BRTC服务器的URL， 默认值是wss://rtc.exp.bcelive.com:8989/janus。

参数

参数	类型	描述
MediaServerURL	char*	BRTC服务器连接地址

返回

无

BaiduRtcRoomClient::setCER

```
void setCER(const char *cerFile);
```

介绍

设置用于BRTC连接的根证书。

ssl证书文件，用于RTC信令服务器的根证书, 已经随SDK提供。

参数

参数	类型	描述
cerFile	char*	证书文件路径

返回

无

BaiduRtcRoomClient::Destory

```
void Destory() ;
```

介绍

销毁本对象。

销毁对象，释放对象的内存，断开和服务器连接，释放网络资源。

参数

无

返回

无

房间相关接口 BaiduRtcRoomClient::loginRoom

```
bool BaiduRtcRoomClient::loginRoom(const char* roomName, const char* userId, const char* displayName, const char* token);
```

介绍

房间登录。

登录房间成功后，用户可以发布流，或订阅流，并能通过IRtcMessageListener接口获取房间状态信息。

参数

参数	类型	描述
roomName	char*	房间名，长度不超过255字节
userId	char*	用户id, 数字字符串，每个房间的用户id必须唯一
displayName	char*	用户显示名
token	char*	用于鉴权的token串

返回

true 成功， false 失败

BaiduRtcRoomClient::logoutRoom

```
bool BaiduRtcRoomClient::logoutRoom()
```

介绍

房间登出。

关闭媒体通道，关闭信令通道，释放内存资源，及销毁其他申请的资源。

参数

无

返回

true 成功， false

发布/订阅流相关接口 BaiduRtcRoomClient::startPublish

```
void BaiduRtcRoomClient::startPublish()()
```

介绍

发布流。

用户发布流。

参数

无

返回

无

BaiduRtcRoomClient::stopPublish

```
void BaiduRtcRoomClient::stopPublish()
```

介绍

停止发布流。

停止手动发布流。

参数

无

返回

无 **BaiduRtcRoomClient::subscribeStreaming**

```
void BaiduRtcRoomClient::subscribeStreaming(const char * feedId, IAudioFrameObserver *afo, IVideoFrameObserver *vfo, IDataFrameObserver *dfo) = 0;
```

介绍

订阅流。

用于订阅同一房间的其他用户的流。

参数

参数	类型	描述
feedId	const char *	用户要订阅的其他用户的流id列表
afo	IAudioFrameObserver	音频回调接口， 也可以填写NULL，表示使用Client对象的音频回调接口
vfo	IVideoFrameObserver	视频回调接口， 也可以填写NULL，表示使用Client对象的音频回调接口
dfo	IDataFrameObserver	Data回调接口， 也可以填写NULL，表示使用Client对象的音频回调接口

返回

无

BaiduRtcRoomClient::stopSubscribeStreaming void BaiduRtcRoomClient::stopSubscribeStreaming(const char * feedId)

介绍

停止订阅流。

停止订阅feedId的码流。

参数

参数	类型	描述
feedId	const char *	用户要停止订阅的流id

返回

无

音视频参数设置 BaiduRtcRoomClient::setParamSettings void setParamSettings(RtcParameterSettings* paramSettings,RtcParameterSettings::RtcParamSettingType paramType)

介绍

设置视频编码等参数。

该接口用于设置音视频编码的参数。

参数

参数	类型	描述
paramSettings	RtcParameterSettings	音视频参数集
paramType	RtcParamSettingType	设置的参数类型

RtcParameterSettings成员	类型	描述
HasMultiNetwork	bool	是否启用多卡， 默认false
HasVideo	bool	是否启用视频， 默认true
HasAudio	bool	是否启用音频， 默认true
HasData	bool	是否启用数据通道， 默认false
AudioINFrequency	int	音频输入采样率
AudioINChannel	int	音频输入通道数
AudioOUTFrequency	int	音频输出采样率
AudioOUTChannel	int	音频输出通道数
VideoWidth	int	输入视频的宽度
VideoHeight	int	输入视频的高度
VideoFps	int	输入视频的帧率
VideoMaxkbps	int	视频发送的最大带宽值，默认1500Kbps
VideoMinkbps	int	视频发送的最小带宽值
ImageINType	RtcImageType	视频输入的数据类型， 默认自动识别
ImageOUTType	RtcImageType	视频输出的数据类型， 目前仅支持H264
ConnectionTimeoutMs	int	连接超时时长， 默认5秒
ReadTimeoutMs	int	读数据超时， 默认5秒
AutoPublish	bool	是否自动发布流， 默认true
AutoSubscribe	bool	是否自动拉流， 默认true
AsPublisher	bool	是否作为发布者， 默认true
AsListener	bool	是否作为接收者， 默认true

RtcParamSettingType枚举值	类型	描述
RTC_PARAM_SETTINGS_ALL	enum	更新全部参数
RTC_VIDEO_PARAM_SETTINGS_BITRATE	enum	只更新视频最大编码码率

RtcImageType枚举值	类型	描述
RTC_IMAGE_TYPE_NONE	enum	未指定流类型
RTC_IMAGE_TYPE_JPEG	enum	JPEG图片流
RTC_IMAGE_TYPE_H264	enum	H264的NAL视频格式
RTC_IMAGE_TYPE_I420P	enum	I420P视频流
RTC_IMAGE_TYPE_RGB	enum	RGB视频流

返回

无

BaiduRtcRoomClient::setFeedId void setFeedId(const char * feedId)

介绍

设置拉流的种子id。

设置拉取的音视频流的种子id。

参数

参数	类型	描述
feedId	char*	用户要订阅的音视频流id

返回

无

BaiduRtcRoomClient::registerVideoFrameObserver void
BaiduRtcRoomClient::registerVideoFrameObserver(IVideoFrameObserver* iVfo[], int iVfoNum)

介绍

注册视频数据帧到达Observer。

当使用外部渲染时，需要注册视频数据监测接口，有视频数据到来时，可通知用户去做渲染或其他处理。接口详细信息可查看头文件：BaiduExternalVideoRendererInterface.h

参数

参数	类型	描述
iVfo	IVideoFrameObserver*	数据帧到达Observer接口数组，多个接口可多路流，一个接口listening一路流
iVfoNum	int	Observer 接口数

返回

无

BaiduRtcRoomClient::registerAudioFrameObserver void
BaiduRtcRoomClient::registerAudioFrameObserver(IAudioFrameObserver* iAfo[], int iAfoNum)

介绍

注册音频数据帧到达Observer。

用于接收远端的音频数据，PCM格式。接口详细信息可查看头文件：BaiduExternalVideoRendererInterface.h

参数

参数	类型	描述
iAfo	IAudioFrameObserver*	数据帧到达Observer接口数组，多个接口可多路流，一个接口listening一路流
iAfoNum	int	Observer 接口数

返回

无

BaiduRtcRoomClient::sendImage void BaiduRtcRoomClient::sendImage(const char *data, int len)

介绍

发送视频数据。

发送视频数据, 数据类型可以是none,jpeg,h264,yuv，需要通过setParamSettings指定,默认是none即自动识别

参数

参数	类型	描述
data	const char *	视频数据内存地址
len	int	视频数据大小

返回

无

BaiduRtcRoomClient::sendAudio void BaiduRtcRoomClient::sendAudio(const char *data, int len)

介绍

发送音频数据。

发送音频数据, 数据类型为PCM数据，需要通过setParamSettings指定输入音频的采样和通道参数

参数

参数	类型	描述
data	const char *	音频数据内存地址
len	int	音频数据大小

返回

无

消息接口 BaiduRtcRoomClient::sendMessageToUser void BaiduRtcRoomClient::sendMessageToUser(const char *msg, const char* id)

介绍

发送消息。

可通过该接口发送自定义消息给聊天室中的用户， 当id为"0"时， 表示广播消息。

参数

参数	类型	描述
msg	const char*	消息体， 建议使用json串
id	const char*	接收消息的用户id

返回

无

BaiduRtcRoomClient::registerRtcMessageListener void BaiduRtcRoomClient::registerRtcMessageListener(IRtcMessageListener* msgListener)

介绍

消息监听接口。

用户可以实现参数定义的接口，并通过该接口注册，当sdk 启动后，可以收到来自sdk的消息，并做相应的处理。接口详细信息可查看头文件：BaiduRtcCommonDefine.h

参数

参数	类型	描述
msgListener	IRtcMessageListener*	rtc sdk 消息上报接口

RtcMessage成员	类型	描述
msgType	RtcMessageType	消息类型
data	union	联合类型
feedId	int64_t	消息id号
errorCode	int64_t	错误码
extra_info	const char*	消息额外信息

RtcMessageType枚举值	对应值	描述
RTC_MESSAGE_ROOM_EVENT_LOGIN_OK	100	用户登录成功
RTC_MESSAGE_ROOM_EVENT_LOGIN_TIMEOUT	101	用户登录超时
RTC_MESSAGE_ROOM_EVENT_LOGIN_ERROR	102	用户登录失败
RTC_MESSAGE_ROOM_EVENT_CONNECTION_LOST	103	连接丢失
RTC_MESSAGE_ROOM_EVENT_REMOTE_COMING	104	远端视频出现
RTC_MESSAGE_ROOM_EVENT_REMOTE_LEAVING	105	远端视频流离开
RTC_MESSAGE_ROOM_EVENT_REMOTE_RENDERING	106	远端开始渲染
RTC_MESSAGE_ROOM_EVENT_REMOTE_GONE	107	远端结束渲染
RTC_MESSAGE_ROOM_EVENT_SERVER_ERROR	108	服务器错误，错误码为errorCode， 错误详情为extra_info
RTC_ROOM_EVENT_AVAILABLE_SEND_BITRATE	200	可用带宽事件，data 为带宽值
RTC_ROOM_EVENT_ON_USER_JOINED_ROOM	300	用户进入房间事件
RTC_ROOM_EVENT_ON_USER_LEAVING_ROOM	301	用户离开房间事件
RTC_ROOM_EVENT_ON_USER_MESSAGE	302	用户自定义消息事件，feedId 为发送方的id， extra_info为消息内容
RTC_ROOM_EVENT_ON_USER_ATTRIBUTE	303	用户属性变更事件，feedId为变更方的id， extra_info为属性内容
RTC_MESSAGE_STATE_STREAM_UP	2000	音视频流建立事件
RTC_MESSAGE_STATE_SENDING_MEDIA_OK	2001	音视频媒体发送成功事件
RTC_MESSAGE_STATE_SENDING_MEDIA_FAILED	2002	音视频媒体发送失败事件
RTC_MESSAGE_STATE_STREAM_DOWN	2003	音视频流断开事件
RTC_STATE_STREAM_SLOW_LINK_NACKS	2100	网络状态事件
RTC_STATE_SIGNAL_DELAY_ECHO	2101	信令延迟反馈
RTC_MESSAGE_SOURCE_VIDEO_INFO	3000	发送端的网络带宽

返回

无

BaiduRtcRoomClient::setUserAttribute void BaiduRtcRoomClient::setUserAttribute(const char * a)

介绍

设置用户属性。

给用户设置属性，可以把用户状态保存起来， 并通知房间中的其他用户。

参数

参数	类型	描述
a	const char *	设置当前用户的属性值

返回

无 **BaiduRtcRoomClient::getUserAttribute** void BaiduRtcRoomClient::getUserAttribute(const char * id)

介绍

获取用户属性。

获取用户属性，属性值通过事件RTC_ROOM_EVENT_ON_USER_ATTRIBUTE回调。

参数

参数	类型	描述
id	const char *	用户id

返回

返回的属性值通过事件RTC_ROOM_EVENT_ON_USER_ATTRIBUTE回调得到。

BaiduRtcRoomClient::setUE4 void BaiduRtcRoomClient::setUE4(const char *ue4)

介绍

设置UE4直连。

设置UE4的地址进行直接把UE4的渲染数据推到BRTC。

参数

参数	类型	描述
ue4	const char *	UE4的PixelStreaming的ip:port

返回

无

BaiduRtcRoomClient::setCandidateIP void BaiduRtcRoomClient::setCandidateIP(const char *candidateIP)

介绍

设置RTP候选地址。

设置RTP候选地址，和setMediaServerIP配合使用可以通过nginx UDP代理转发RTP数据。

参数

参数	类型	描述
candidateIP	const char *	推流端RTP的候选地址

返回

无

BaiduRtcRoomClient::setMediaServerIP void BaiduRtcRoomClient::setMediaServerIP(const char *mediaserverIP)

介绍

设置固定接收RTP的服务器IP地址。

设置固定接收RTP的服务器IP地址，和setCandidateIP配合使用可以通过nginx UDP代理转发RTP数据。

参数

参数	类型	描述
mediaserverIP	const char *	百度RTC的RTP媒体服务器列表中的IP

返回

无

其他接口

getBaiduRtcSdkVersion const char* getBaiduRtcSdkVersion()

介绍

获取SDK版本号。

获取百度 RTC SDK 版本号。 **参数**

无

返回

获得版本号的字符串

enableBaiduRtcLog void enableBaiduRtcLog(bool isEnable)

介绍

日志功能。

日志功能开关。

参数

参数	类型	描述
isEnable	bool	若为 true, 开启日志功能， 开启日志功能后，会把日志保存到本地文件中；若为false 关闭日志功能。

返回

无

macOS SDK

概述

概述

本文档主要介绍如何将 RTC macOS SDK 集成到您的项目中，并介绍各类功能的使用方法。在使用本文档前，您需要先了解RTC的一些基本知识，并已经开通了RTC服务。若您还不了解RTC，可以参考产品描述。

RTC macOS SDK 能够帮助您实现登录音视频通信房间并开始通信，并可以对发布/订阅、音视频参数、设备参数进行设置。 **接口概览** RTC macOS SDK 提供以下接口 **初始化接口** |API|描述| |-|-| |initSDKWithAppID|初始化SDK| |setParamSettings|音视频相关参数设置| |getParamSettings|获取音视频相关设置参数| |setEngineStateStatistics|RTC统计信息开关|

房间相关接口 |API|描述| |-|-| |loginRtcRoomWithRoomName|登录房间| |logoutRtcRoom|退出房间| |kickOffUserWithId|踢除聊天| |shutUpUserWithId|禁言| |disbandRoom|解散房间| |queryUserListOfRoom|查询媒体用户| |queryMessageUserListOfRoom|查询房间用户| **发布/订阅流相关接口** |API|描述| |-|-| |publishStreaming|发布流| |stopPublishStreaming|停止发布流| |subsribeStreaming|订阅流| |stopSubscribeStreaming|停止订阅流|

消息相关接口 |API|描述| |-|-| |sendMessage2|广播消息| |sendMessage2WithUserId|指定用户发送消息| |setUserAttribute|attribute 属性设置| |getUserAttribute|attribute属性获取|

视频相关接口 |API|描述| |-|-| |startPreview|本地预览| |stopPreview|停止预览| |setLocalDisplay|本地显示view设置| |setRemoteDisplay|远端view设置| |setRemoteDisplay: userId:|多人模式设置指定用户远端view| |updateDisplay: userId:|多人模式更新指定用户远端view| |setVideoCaptureFactory|外部采集代理设置| |setRenderDelegate|外部渲染代理设置| |setRemoteVideoPlayState|设置远端视频流订阅状态|

摄像头相关接口 |API|描述| |-|-| |switchCamera|摄像头切换| |muteCamera|关闭/打开摄像头| |setCameraFace|关闭/打开摄像头| |cameraFocusWithPoint|摄像头对焦|

音频相关接口 |API|描述| |-|-| |muteMicphone|关闭麦克风| |getRemoteAudioLevels|获取远端用户音量| |setRemoteAudioPlayState|指定某个远端用户音频暂停/播放|

美颜相关接口 |API|描述| |-|-| |setWhitenFactor|美白度设置| |setRedenFactor|红润度设置| |setBuffingFactor|磨皮度设置|

转推配置相关接口 |API|描述| |-|-| |configLiveServerWithUrl|转推配置|

通知相关接口 |API|描述| |-|-| |onRoomEventUpdate|用户信息通知| |onPeerConnectStateUpdate|server端连接状态通知| |onStreamInfoUpdate|媒体流通知| |onErrorInfoUpdate|错误通知| |onEngineStatisticsInfo|RTC引擎状态信息统计| |onTextMessageArrival2|消息通知| |onTextMessageAttribute|属性更新通知|

其他接口 |API|描述| |-|-| |version|版本号| |setUseTestEnv|是否启用测试环境| |setVerbose|是否打开调试信息| |upLoadLog|上报日志| |setBaiduRtcAppID|动态设置AppId和TokenStr| |queryEngineStatisticsInfo|查询RTC统计信息| |enableStatsToServer|RTC统计信息上报|

API

初始化接口

initSDKWithAppID

```
- (instancetype)initSDKWithAppID:(NSString *)appId
                       tokenStr:(NSString *)tokenStr
                       delegate:(id<BaiduRtcRoomDelegate>)delegate;
```

介绍

初始化SDK。

初始化SDK时调用。设置 AppID、TokenStr 等。

参数

参数	类型	描述
appId	NSString*	RTC 基础业务单元的唯一标识
tokenStr	NSString*	RTC Server 端鉴权使用的字符串
delegate	id<BaiduRtcRoomDelegate>	遵循 BaiduRtcRoomDelegate 协议的对象

返回

返回 SDK 实例，nil 表示初始化失败

setParamSettings

```
- (void)setParamSettings:(RtcParameterSettings *)paramSettings paramType:(RtcParamSettingType)paramType;
```

介绍

音视频相关参数设置。

该函数在 loginRtcRoomWithRoomName 前调用，用于设置音视频采集，编解码相关的参数。

参数

参数	类型	描述
paramSettings	RtcParameterSettings*	设置参数包括分辨率、帧率、码率、视频方向等
paramType	RtcParamSettingType	参数类型，可指定设置某一项，或设置所有参数

RtcParameterSettings

```
 @property (nonatomic, assign) BOOL hasAudio;    // 是否采集音频数据
@property (nonatomic, assign) BOOL isCreatingAecDump; // 是否开启aecdump
@property (nonatomic, assign) BOOL isUsingLevelControl; // 是否开启等级控制
@property (nonatomic, assign) BOOL isUsingManualConfig; // 是否开启手动配置
@property (nonatomic, assign) BOOL isExportAudioRecord; // 是否开启外部采集
@property (nonatomic, assign) BOOL isExportAudioPlayout; // 是否开启外部播放
@property (nonatomic, assign) BOOL isExportAudioRecordPlayoutMix; // 是否开启外部混音
@property (nonatomic, assign) BOOL hasVideo; // 是否开启视频采集
@property (nonatomic, assign) int videoFps;  // 视频帧率
@property (nonatomic, assign) int videoWidth; // 视频宽
@property (nonatomic, assign) int videoHeight; // 视频高
@property (nonatomic, assign) int videoBitrate; // 视频码率(bps)
@property (nonatomic, copy) NSString* videoCodecType; // 编码类型
@property (nonatomic, assign) BOOL isEnableExternalCapturer; // 是否开启外部采集
@property (nonatomic, assign) BOOL isEnableExternalRender; // 是否开启外部渲染
@property (nonatomic) BOOL isAutoPublish; // 是否自动发布流
@property (nonatomic) BOOL isAutoSubscribe; // 是否自动订阅流
@property (nonatomic, assign) BOOL isMultiPlayerModel; // 是否是多人模式
@property (nonatomic, assign) BOOL isEnableDataChannel; // 是否开启数据通道
@property (nonatomic, assign) LiveServerInfo *roomTransInfo; // 房间转推配置
@property (nonatomic, assign) LiveServerInfo *anchorTransInfo; // 主播转推配置
@property (nonatomic, assign) BOOL isMixing; // 是否混流
@property (nonatomic, copy) NSString *rtmpUrl;  // 转推地址
@property (nonatomic, assign) RtcLiveTransferMode liveTransferMode; // 转推模式
```

返回

无

getParamSettings

```
- (RtcParameterSettings *)getParamSettings;
```

介绍

获取音视频相关参数。

获取当前设置的音视频相关参数，如分辨率、帧率、码率、视频方向等。

参数

无

返回

RtcParameterSettings 音视频参数信息

setEngineStateStatistics

```
- (void)setEngineStateStatistics:(BOOL)bOnStatistics;
```

介绍

RTC统计信息开关。

当打开开关时 onEngineStatisticsInfo 函数每隔1秒返回一次统计信息，并且可通过queryEngineStatisticsInfo 函数主动查询RTC统计信息。统计信息包括 CPU, FPS，video codec等。

参数

参数	类型	描述
bOnStatistics	boolean	bOnStatistics = YES, 打开开关；bOnStatistics = NO, 关闭引擎统计信息

返回

无

房间相关接口

loginRtcRoomWithRoomName

```
- (BOOL)loginRtcRoomWithRoomName:(NSString *)roomName userID:(int)userId;
```

介绍

登录房间。

登录房间成功，在同一个房间的人能进行相互音视频聊天，如果失败，会通过onErrorInfoUpdate call back 返回错误信息。

参数

参数	类型	描述
roomName	NSString*	房间名，长度不可超过 255 byte
userId	int	用户 id,每个房间的用户 ID 必须唯一

返回

true 成功，false 失败

logoutRtcRoom

```
- (bool)logoutRtcRoom;
```

介绍

退出房间。

执行 logoutRtcRoom 后，会停止音视频采集，断开与房间服务器的连接，取消音视频的传输，销毁音视频传输通道以及释放其他资源。

参数

无

返回

true 成功，false 失败

kickOffUserWithId

```
- (void)kickOffUserWithId:(int)userId;
```

介绍

踢除聊天。

房管／主播／会议主持 把某人踢出聊天室。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户 ID

返回

无

shutUpUserWithId

```
- (void)shutUpUserWithId:(int)userId;
```

介绍

禁言

房管/主播／会议主持 禁止某人发言。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户 ID

返回

无

disbandRoom

```
- (void)disbandRoom;
```

介绍

解散房间

房管/主播／会议主持 解散房间。

参数

无

返回

无

queryUserListOfRoom

```
- (NSArray *)queryUserListOfRoom;
```

介绍

查询媒体用户。

查询房间用户信息，获取房间中所有媒体用户列表。

参数

无

返回

NSArray* 用户信息列表

queryMessageUserListOfRoom

```
- (NSArray *)queryMessageUserListOfRoom;
```

介绍

查询消息用户。

查询房间用户信息，获取房间中 Message 用户列表。

参数

无

返回

NSArray* 用户信息列表

发布/订阅流相关接口 publishStreaming

```
- (void)publishStreaming;
```

介绍

发布流。

流发布在 roomId 指定的房间，在同一房间 joined 的用户可以相互订阅流，默认在发布流的同时， listening/subscriber 在该房间其他用户的流。

参数

无

返回

无

stopPublishStreaming

```
- (void)stopPublishStreaming;
```

介绍

停止发布流。

stop 通过 publishStreaming 发布的流。

参数

无

返回

无

subsribeStreaming

```
- (void)subsribeStreaming:(NSArray<NSNumber *> *)streamingIds;;
```

介绍

订阅流。

用于订阅同一房间的其他用户的流.

参数

参数	类型	描述
streamingIds	NSArray<NSNumber*>*	用户要订阅的其他用户的流id列表(即其他用户id列表)

返回

无

stopSubscribeStreaming

```
- (void)stopSubscribeStreaming:(NSArray<NSNumber *> *)streamingIds;;
```

介绍

停止订阅流。

该接口停止已经订阅的流.

参数

参数	类型	描述
streamingIds	NSArray<NSNumber*>*	用户要停止订阅的流id列表(即其他用户id列表)

返回

无

消息相关接口

sendMessage2

```
- (int)sendMessage2:(NSString *)message;
```

介绍

广播消息。

广播消息给在同一房间的其他用户。

参数

参数	类型	描述
message	NSString*	需要发送的文本消息

返回

-1 发送失败，0 发送成功

sendMessage2WithUserId

```
- (int)sendMessage2WithUserId:(NSString *)message userId:(NSNumber *)userId;
```

介绍

指定用户发送消息。

发送文本消息给在同一房间的指定用户。

参数

参数	类型	描述
message	NSString*	需要发送的文本消息
userId	NSNumber*	指定接收消息的用户 ID

返回

-1 发送失败，0 发送成功

setUserAttribute

```
- (int)setUserAttribute:(NSString *)attribute;
```

介绍

attribute 属性设置。

设置本用户 attribute 属性，房间其他用户由 onTextMessageAttribute 通知收到。

参数

参数	类型	描述
attribute	NSString*	attribute 值

返回

-1 发送失败，0 发送成功

getUserAttribute

```
- (int)getUserAttribute:(NSNumber *)userID;
```

介绍

attribute属性获取。

获取指定用户 attribute 属性，属性值由 onTextMessageAttribute 通知得到。

参数

参数	类型	描述
userID	NSNumber*	用户ID

返回

-1 发送失败，0 发送成功

视频相关接口

setVideoCaptureFactory

```
- (void)setVideoCaptureFactory:(id<BaiduVideoCaptureFactory>)factory;
```

介绍

外部采集设置。

设置外部采集模块。必须在登录房间前调用，并且不能置空。

参数

参数	类型	描述
factory	id<BaiduVideoCaptureFactory>	工厂对象，遵循 BaiduVideoCaptureFactory 协议的对象

返回

无

setRenderDelegate

```
- (void)setRenderDelegate:(id<BaiduRtcApiRenderDelegate>)renderDelegate;
```

介绍

外部渲染设置。

设置外部渲染回调对象。使用外部渲染功能，需要设置代理对象。未设置代理对象，或对象设置错误，可能导致无法正常收到

相关回调

> 注意：请使用 BaiduExternalVideoRender

参数

参数	类型	描述
renderDelegate	id<BaiduRtcApiRenderDelegate>	遵循 BaiduRtcApiRenderDelegate 协议的代理对象

返回

无

startPreview

```
- (void)startPreview;
```

介绍

本地预览。

打开camera，开始预览。

参数

无

返回

无

stopPreview

```
- (void)stopPreview;
```

介绍

停止预览。

关闭camera, 停止本地预览。

参数

无

返回

无

setLocalDisplay

```
- (void)setLocalDisplay:(NSView<RTCVideoRenderer> *)localVideoView;
```

介绍

本地显示view设置。

设置本地显示view。在 loginRtcRoomWithRoomName 之前调用，loginRtcRoomWithRoomName 之后，本地视频数据会显示到

localVideoView, localVideoView 的位置大小要与采集的视频大小成比例。

参数

参数	类型	描述
localVideoView	NSView <RTCVideoRenderer> *	本地显示view,用于显示camera采集的视频数据

返回

无

setRemoteDisplay

```
- (void)setRemoteDisplay:(NSView<RTCVideoRenderer> *)remoteVideoView;
```

介绍

远端view显示。

设置远端显示view。在 loginRtcRoomWithRoomName 之前调用，loginRtcRoomWithRoomName 之后，远端视频数据会显示到 remoteVideoView，remoteVideoView 的大小要与采集的视频大小成比例。

参数

参数	类型	描述
remoteVideoView	NSView <RTCVideoRenderer> *	远端画面显示view, 用于显示远端用户传输过来的视频数据

返回

无

setRemoteDisplay: userId:

```
- (void)setRemoteDisplay:(NSView<RTCVideoRenderer> *)remoteVideoView userId:(NSInteger)userId;
```

介绍

多人模式设置指定用户远端 view。

在远端用户流到达后调用，指定用户视频数据会显示到 remoteVideoView，remoteVideoView 的大小要与采集的视频大小成比例

参数

参数	类型	描述
remoteVideoView	NSView <RTCVideoRenderer> *	远端画面显示 view, 用于显示远端用户传输过来的视频数据
userId	NSInteger	远端用户ID

返回

无

updateDisplay: userId:

```
- (void)updateDisplay:(NSView<RTCVideoRenderer> *)remoteVideoView userId:(NSInteger)userId;
```

介绍

更新指定用户的远端显示 view

多人模式，更新指定用户视频数据显示view，remoteVideoView 的大小要与采集的视频大小成比例

参数

参数	类型	描述
remoteVideoView	NSView <RTCVideoRenderer> *	远端画面显示 view, 用于显示远端用户传输过来的视频数据
userId	NSInteger	远端用户ID

返回

无

setRemoteVideoPlayState

```
-(void)setRemoteVideoPlayState:(BOOL)stats userId:(NSInteger)userId;
```

介绍

指定远端画面暂停/恢复播放

通过控制指定远端用户流是否拉取，控制指定远端用户画面暂停/恢复播放

参数

参数	类型	描述
stats	BOOL	true 订阅； false 停止订阅
userId	NSInteger	远端用户ID

返回

无

摄像头相关接口

switchCamera

```
- (void)switchCamera;
```

介绍

摄像头切换。

切换摄像头，前后摄像头切换。

参数

无

返回

无

muteCamera

```
- (void)muteCamera;
```

介绍

关闭/打开摄像头。

在关闭摄像头后，本地无法预览，且不传输本地视频数据给对方

参数

无

返回

无

cameraFocusWithPoint

```
- (void)cameraFocusWithPoint:(CGPoint)point andPlaneSize:(CGSize)size;
```

介绍

摄像头对焦。

摄像头对焦功能，支持手动和自动对焦。

参数

参数	类型	描述
point	CGPoint	对焦点坐标
size	CGSize	camera预览view尺寸

返回

无

setCameraFace

```
- (void)setCameraFace:(BOOL)front
```

介绍

设置前后置摄像头。

设置前后置摄像头，默认前置摄像头。

参数

参数	类型	描述
front	BOOL	true 前置；false 后置

返回

无

音频相关接口

muteMicphone

```
- (void)muteMicphone;
```

介绍

关闭麦克风。

关闭麦克风，停止音频的采集

参数

无

返回

无

getRemoteAudioLevels

```
- (NSArray *)getRemoteAudioLevels;
```

介绍

获取远端用户音量。

获取远端用户音量，返回数组包含各远端用户userID及对应音量。

参数

无

返回

RtcRoomAudioLevel 数组

RtcRoomAudioLevel

```
 @interface RtcRoomAudioLevel : NSObject  // 用户语音激励
@property (nonatomic, copy) NSNumber *userID; // 用户 ID
@property (nonatomic, copy) NSString *userName;  // 用户名
@property (nonatomic, assign) NSInteger volumeLevel; // 音量
@end
```

setRemoteAudioPlayState

```
- (void)setRemoteAudioPlayState:(BOOL)stats userId:(NSInteger)userId;
```

介绍

指定某个远端用户音频暂停/播放。

控制特定远端用户音频暂停/播放。控制音频流是否拉取

参数

参数	类型	描述
stats	BOOL	true 拉取；false 停止拉取
userId	NSInteger	用户 ID

返回

无

美颜相关接口

setWhitenFactor

```
- (void)setWhitenFactor:(float)factor;
```

介绍

美白度设置。

设置美白度，调节美白系数

参数

参数	类型	描述
factor	float	在美颜中的美白因子

返回

无

setRedenFactor

```
- (void)setRedenFactor:(float)factor;
```

介绍

红润度设置。

设置红润度，调节红润系数。

参数

参数	类型	描述
factor	float	在美颜中的红润因子

返回

无

setBuffingFactor

```
- (void)setBuffingFactor:(float)factor;
```

介绍

磨皮度设置。

设置磨皮度，调节磨皮系数

参数

参数	类型	描述
factor	float	在美颜中的磨皮因子

返回

无

转推配置相关接口

configLiveServerWithUrl

```
- (void)configLiveServerWithUrl::(NSString *)url
                           isMix:(BOOL)isMix
                     isRecording:(BOOL)isRecording
                     mixTemplate:(NSString *)mixTmplt
                    transferMode:(RtcLiveTransferMode)mode
                         avParam:(RtcLiveAuidoVideoParameters*)avParam;
```

介绍

该接口用于配置server推流的参数，聊天室模式：在同一个rtc房间的所有参与者在混流后，直接转推到一个指定的直播房间；主播转推模式：主播推向不同的直播房间。

参数

参数	类型	描述
url	NSString*	转推url地址
isMix	BOOL	是否混流
isRecording	BOOL	是否录制
mixTemplate	NSString*	混流模板
transferMode	RtcLiveTransferMode	转推模式
avParam	RtcLiveAuidoVideoParameters*	转推参数

返回

无

通知相关接口

onRoomEventUpdate

```
- (void)onRoomEventUpdate:(RtcRoomEvents *)roomEvents;
```

介绍

房间中用户加入或离开时，通知房间中其他用户，其他用户可接收到房间中他人离开的信息。

参数

参数	类型	描述
roomEvents	RtcRoomEvents	房间事件信息，具体信息参考 BaiduRtcRoomApiDefines.h

回调事件

```
 RTC_ROOM_EVENTS_ROOM_CREATED = 100,  // 创建房间成功
RTC_ROOM_EVENTS_ROOM_JOINED,          // 加入房间成功
RTC_ROOM_EVENTS_USER_JOINED,         // 有用户加入房间
RTC_ROOM_EVENTS_USER_LEAVING,        // 有用户离开房间
RTC_ROOM_EVENTS_RECEIVED_HANGUP,     // 收到server端挂断通知
RTC_ROOM_EVENTS_RECEIVED_TIMEOUT,    // server端接收信令超时通知, 没有收到client发送到信令
RTC_ROOM_EVENTS_SENT_TIMEOUT,        // client端发送信令超时，没有server responce
RTC_ROOM_EVENTS_SENDING_MEDIA_OK, // 媒体数据发送成功
RTC_ROOM_EVENTS_SENDING_MEDIA_FAILED, // 媒体数据发送失败
RTC_ROOM_EVENTS_PUBLISHE_STREAMING_ERROR, // server端规定时间内未能收到client端推送上去的流
RTC_ROOM_EVENTS_USEREVENTS_USER_JOINED = 110, // userevent事件 用户加入房间
RTC_ROOM_EVENTS_USEREVENTS_USER_LEAVING,  // userevent事件 用户退出房间
RTC_ROOM_EVENTS_DISBAND_ROOM = 112,  // 解散房间回调
RTC_ROOM_EVENTS_SOMEBODY_SHUTUPED = 113, // 禁言回调
RTC_ROOM_EVENTS_SOMEBODY_DISSHUTUPED = 114, // 解禁回调
RTC_ROOM_EVENTS_SOMEBODY_KICKOFFED = 115, // 踢人回调
RTC_ROOM_EVENTS_OTHERS               // 未知事件
```

返回

无

onPeerConnectStateUpdate

```
- (void)onPeerConnectStateUpdateWithType:(RtcPeerConnectionType)type stats:(int)connecStats;
```

介绍

与server连接状态通知。与server端的连接状态信息更新，包括信令/媒体／数据通道，如开始建连，连接成功，连接失败等

参数

参数	类型	描述
type	RtcPeerConnectionType	状态类型：媒体、数据、信令
connecStates	int	连接状态码

回调事件

```
 // 信令通道状态
RTC_SIGNAL_CHANNEL_STATS_ESTABLISHING = 150, // 信令通道正在建立
RTC_SIGNAL_CHANNEL_STATS_ESTABLISHED,        // 信令通道已经建立
RTC_SIGNAL_CHANNEL_STATS_CLOSED,             // 信令通道关闭
RTC_SIGNAL_CHANNEL_STATS_FAILED,             // 信令通道建立失败
RTC_SIGNAL_CHANNEL_STATS_DISCONNECT,         // 信令通道失去连接
RTC_SIGNAL_CHANNEL_STATS_OTHERS              // 其它

// 媒体通道状态
RTC_MEDIA_CHANNEL_STATS_CONNECTING = 200,    // 媒体通道正在建立
RTC_MEDIA_CHANNEL_STATS_CONNECTED,           // 媒体通道建立完成
RTC_MEDIA_CHANNEL_STATS_DISCONNECTED,        // 媒体通道断开连接
RTC_MEDIA_CHANNEL_STATS_FAILED,              // 媒体通道建立失败
RTC_MEDIA_CHANNEL_STATS_OTHERS               // 其它

// 数据通道
RTC_DATA_CHANNEL_STATS_CONNECTING = 300,     // 数据通道正在建立
RTC_DATA_CHANNEL_STATS_CONNECTED,            // 数据通道建立成功
RTC_DATA_CHANNEL_STATS_DISCONNECTED,         // 数据通道断开连接
RTC_DATA_CHANNEL_STATS_FAILED,               // 数据通道建立失败
RTC_DATA_CHANNEL_STATS_OTHERS                // 其它
```

返回

无

onStreamInfoUpdate

```
- (void)onStreamInfoUpdate:(RtcStreamInfo *)rtcStreamInfo;
```

介绍

媒体流信息通知。当有远端或本地流信息到来时，通知app层做后续处理

参数

参数	类型	描述
rtcStreamInfo	RtcStreamInfo	具体流相关的信息

回调事件

```
 RTC_VIDEO_STREAMING_STATES_ARRIVAL = 0,  // 视频流到达事件
RTC_VIDEO_STREAMING_STATES_REMOVE,        // 视频流离开事件
RTC_AUDIO_ONLY_STREAMING_STATES_ARRIVAL, // 纯音频流到达事件
RTC_AUDIO_ONLY_STREAMING_STATES_REMOVE,  // 纯音频流离开事件
RTC_VIDEO_STREAMING_STATES_OTHERS         // 其它事件
```

返回

无

onErrorInfoUpdate

```
- (void)onErrorInfoUpdate:(RtcErrorCodes)errorInfo;
```

介绍

错误信息通知。RTC通信过程中，错误信息的反馈

参数

参数	类型	描述
errorInfo	RtcErrorCodes	错误信息字符串

回调事件

```
  RTC_NO_SUCH_ROOM_ERROR = 426,          // 房间不存在
RTC_ROOM_ALREADY_EXIST_ERROR,          // 房间已存在
RTC_USERID_ALREADY_EXIST_ERROR = 436,  // 用户已存在
RTC_ERROR_OTHERS                       // 其它错误
```

返回

无

onEngineStatisticsInfo

```
- (void)onEngineStatisticsInfo:(NSArray *)statistics;
```

介绍

RTC引擎状态信息统计。该callback返回当前rtc engine的一些参数和性能信息，如传输fps,码率，网络状况,cpu等信息给app

参数

参数	类型	描述
statistics	NSArray	rtc引擎状态信息统计数组

返回

无

onTextMessageArrival2

```
- (void)onTextMessageArrival2:(RtcMessageInfo *)msg;
```

介绍

当房间的其他用户发消息时，会收到该通知

参数

参数	类型	描述
msg	RtcMessageInfo	消息结构体，具体消息在该结构体中

返回

无

onTextMessageAttribute

```
- (void)onTextMessageAttribute:(NSNumber *)userID attribute:(NSString *)attribute;
```

介绍

房间其他用户调用接口setUserAttribute，或本用户调用getUserAttribute获取用户Attribute属性时，该通知被触发，返回用户attribute属性

参数

参数	类型	描述
userID	NSNumber	用户ID
attribute	NSString*	属性值

返回

无

其他接口

version

```
+ (NSString *)version;
```

介绍

获取 iOS SDK 版本号。

参数

无

返回

返回当前版本号

setUseTestEnv

```
+ (void)setUseTestEnv:(BOOL)isUseTestEnv;
```

介绍

是否启用测试环境。

建议在初始化 SDK 前调用。建议开发者在开发阶段设置为测试环境，使用由百度提供的测试环境。上线前需切换为正式环境运营。

参数

参数	类型	描述
isUseTestEnv	boolean	是否启用测试环境，true 启用，false 不启用。默认为 false

返回

无

setVerbose

```
+ (void)setVerbose:(BOOL)bOnVerbose;
```

介绍

是否打开调试信息。

建议在初始化 SDK 前调用。建议在调试阶段打开此开关，方便调试。

参数

参数	类型	描述
bOnVerbose	boolean	是否打开调试信息，true 打开，false 不打开。默认为 false

返回

无

upLoadLog

```
+ (void)upLoadLog;
```

介绍

上报日志。

上传日志到后台便于分析问题。在初始化SDK成功后调用。

参数

无

返回

无

setBaiduRtcAppID

```
+ (void)setBaiduRtcAppID:(NSString *)appId andToken:(NSString *)tokenStr;
```

介绍

动态设置AppId和TokenStr

业务变更需要调整appId、tokenStr时，调用此api可以设置。在初始化sdk前后都可以调用。

参数

参数	类型	描述
appId	NSString*	RTC基础业务单元的唯一标识
tokenStr	NSString*	RTC Server端鉴权使用的字符串

返回

无

queryEngineStatisticsInfo

```
- (NSArray *)queryEngineStatisticsInfo;
```

介绍

查询RTC统计信息。 如CPU, FPS，video codec等

参数

无

返回

NSArray* rtc 状态统计信息

enableStatsToServer

```
- (void)enableStatsToServer:(BOOL)isEnable qualityMonitorEnv:(NSString *)qualityMonitorEnv;
```

介绍

RTC质量监控数据上报。

预置接口，监控信息上报开关 当打开开关时，上报帧率、码率、分辨率、丢包率等监控信息到服务端，console可查。

参数

参数	类型	描述
isEnable	BOOL	是否打开rtc质量监控数据上报，true 打开，false不打开。默认为 false
qualityMonitorEnv	NSString	线上环境："online" 沙盒："qa"。 默认值为 "online"

返回

无

微信小程序 SDK

概述

BRTC微信小程序端SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

API

获取应用实例 const brtc = require('baidu.rtc.mp.sdk.js');

var BRTCClient = brtc.BRTC;

实例的各API函数说明如下:

启动BRTC SDK

```
BRTC_Start()
```

介绍

启动SDK时使用。

参数

参数	类型	描述
server	string	百度的RTC 服务器， 使用默认值即可。
appid	string	百度 派发的数字 ID, 开发者的唯一标识
token	string	app server 派发的token字符串, 用来校验通信的合法性

该接口参数数量较多，请参考下面的参数详解进行了解。

返回

无

参数详解

参数	类型	描述	默认值
server	string	百度智能云RTC服务器， 使用默认值即可。	"wss://rtc.exp.bcelive.com/janus"
appid	string	百度 派发的数字 ID, 开发者的唯一标识	
token	string	app server 派发的token字符串, 用来校验对应通信的合法性	
roomname	string	房间名称	
userid	string	用户ID	数字字符串
displayname	string	显示的用户名	
aspublisher	bool	是否是发布者	默认true，是发布者, fale表示只拉流
usingvideo	bool	是否使用本地视频设备	默认true
usingaudio	bool	是否使用本地音频设备	默认true
remotevideocoming	function(id,display,attribute, pullurl)	远端用户流上线的回调	pullurl 是拉流的RTMP地址, 可以用liveplay进行播放
remotevideoleaving	function(id)	远端用户流离开的回调	
onmessage	function(msg)	消息事件回调{msg.id,msg.data}	
onattribute	function(a)	属性事件回调{a.id,a.attribute}	
userevent	bool	是否启用用户级事件	true表示启用用户级事件， 当用户一旦加入房间就会发出事件。 false 表示不启用. 默认是true
userevent_joinedroom	function(id,display,attribute)	用户加入房间的事件，此时用户还没有发布流	
userevent_leavingroom	function(id,display)	用户离开房间	
success	function(pushurl)	BRTC_Start()成功	pushurl是服务器分配的用于推流的RTMP地址, 可以用livepusher进行推流
error	function(error)	BRTC_Start()失败，或运行过程中出现了错误	
destroyed	function(error)	运行过程中出现错误被销毁的回调	
debuglevel	bool/array	是否打印调试信息	默认值为false， 可取值为: true, false, 'all', ['debug','log','error']

停止BRTC

```
BRTC_Stop()
```

介绍

停止BRTC

参数

无

返回

无

获得SDK的版本号

```
BRTC_Version();
```

介绍

获得SDK的版本号信息。

参数

无

返回

返回BRTC SDK的版本号

获得房间中的远端用户列表

```
BRTC_GetRemoteUsers();
```

介绍

获得房间中的远端用户列表

参数

无

返回

房间中的远端用户列表数组。

返回结果示例：

```
[
  {"id":100219207,"display":"Tom","attribute":""},
  {"id":100241823,"display":"brtc webclient","attribute":""}
]
```

设置用户属性

```
BRTC_SetUserAttribute(attribute);
```

介绍

属性事件回调会发生在BRTC_Start的onattribute 函数上。 属性值保存在BRTC_Start回调函数 remotevideocoming返回的参数里面。

参数

attribute, 表示用户属性的字符串，比如: "{name:'aaa',tel:'12345'}";

返回

无

获取用户属性值

```
BRTC_GetUserAttribute();
```

介绍

根据用户id获得特定用户的属性值。

参数 {onattribute: function (a){} ,feedid: id} onattribute, 设置的回调函数。 feedid， 要获取的用户id号。

返回

返回的属性值在onattribute回调中。

发送用户消息

```
BRTC_SendMessageToUser(msg,id);
```

介绍

本函数用来给特定id用户发送消息或者向房间内发送广播消息。 消息在接收端的onmessage回调函数中接收。发送用户消息的频率应小于100次/秒，超出的话用户消息可能会被丢弃。

参数

msg, 需要发送的消息内容， 为一个字符串， 比如: "{name:'aaa',tel:'12345'}" id, 需要发送消息给对端用户的id值。 **注意: 当id为0或没有id参数时表示在房间内发送广播消息。**

返回

无

FreeRtos SDK

概述

RTC FreeRtos SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

API

初始化相关接口

brtc_create_client

```
void* brtc_create_client();
```

介绍

创建 brtc client。

用户使用该sdk开发的入口，首先需要创建brtc client, 通过该api初始化sdk、发布流、订阅流等其他可提供的api操作。

参数

无

返回

void类型指针

brtc_set_server_url

```
void brtc_set_server_url(void* rtc_client, const char* url);
```

介绍

设置访问的url。

在使用brtc_login_room之前调用进行url设置。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄
url	const char*	RTC client连接的服务器的url

返回

无

brtc_init_sdk

```
bool brtc_init_sdk(void* rtc_client, const char* app_id, const char* room_name, const char* user_id, const char* token, const char* cer_path);
```

介绍

初始化SDK。

初始化FreeRtos rtc sdk, 需在使用之前调用进行初始化。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄
appId	const char*	RTC 基础业务单元的唯一标识
room_name	const char*	房间名，长度不超过255字节
user_id	const char*	用户id, 数字字符串，每个房间的用户id必须唯一
token	const char*	RTC Server 端鉴权使用的字符串
cerPath	const char*	ssl 证书存放路径，证书有百度提供

返回

true 成功， false 失败

brtc_deInit_sdk

```
void brtc_deInit_sdk(void* rtc_client);
```

介绍

销毁SDK。

sdk使用完，可调用该api进行释放，销毁操作。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄

返回

无

brtc_destroy_client

```
void brtc_destroy_client(void* rtc_client);
```

介绍

销毁brtc client。

对brtc client进行销毁操作，释放相关资源。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄

返回

无

房间相关接口

brtc_login_room

```
bool brtc_login_room(void* rtc_client);
```

介绍

房间登录。

登录房间成功后，用户可以发布流或订阅流，并能通过IRtcMessageListener接口获取房间状态信息。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄

返回

true 成功， false 失败

brtc_logout_room

```
bool brtc_logout_room(void* rtc_client);
```

介绍

房间登出。

关闭媒体通道，关闭信令通道，释放内存资源，及销毁其他申请的资源。

参数

参数	类型	描述
rtc_client	void*	RTC client句柄

返回

true 成功， false 失败

消息接口

brtc_register_message_listener

```
void brtc_register_message_listener(void* rtc_client, IRtcMessageListener msgListener);
```

介绍

注册消息监听接口。

用户可以实现参数定义的接口，并通过该接口注册，当sdk启动后，可以收到来自sdk的消息，并做相应的处理。接口详细信息可查看头文件：baidu_rtc_common_define.h

参数

参数	类型	描述
rtc_client	void*	RTC client句柄
msgListener	IRtcMessageListener	RTC sdk 消息上报接口

返回

无

Android 纯音频SDK

概述

产品介绍

RTC Android纯音频SDK 能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。

相对于全功能SDK，提供更小的包体积接入，更低的CPU和内存消耗。适用于普通场景与基于Android系统的物联网场景。

特性

极低包体积增量

- armv7下包体积增量仅938KB

更低的CPU消耗

- 相对于全功能版本，多人语音下CPU消耗减少50%

更低的内存消耗

- 相对于全功能版本，多人语音下内降低22%

应用场景

普通场景

适用于普通的语音通信场景，如语音群聊，连麦等等场景

物联网场景

适用于基于Android系统的物联网产品，提供低功耗，低包体积接入

API

初始化接口 SDK初始化

```
public static synchronized BaiduRtcRoom initWithAppID(Context context, String appId, String token)
```

初始化SDK并创建RTC房间实例，初始化SDK时调用，初始化SDK失败会导致SDK功能异常。

参数

参数	类型	描述
context	Context	Android上下文环境
appId	String	百度为App签发的App ID, 用于识别应用, 全局唯一，详见创建应用。
token	String	百度RTC服务端鉴权使用的密钥，可缺省，使用应用鉴权可使得服务更加安全。 详见应用鉴权。

返回

类型	描述
BaiduRtcRoom	实例对象，null：失败，非null：表示成功

设置媒体参数

```
public abstract void setParamSettings(RtcParameterSettings paramSettings,RtcParameterSettings.RtcParamSettingType paramType);
```

音视频参数设置。

设置音视频相关的参数。该函数在loginRtcRoomWithRoomName接口前调用，主要用于设置音视频采集，编解码相关的参数。

参数

参数	类型	描述
paramSettings	RtcParameterSettings	该参数封装了音视频的一些参数,如音频采样率等
paramType	RtcParamSettingType	参数类型，可指定设置某一项，或者所有参数都设置。

参数类型 RtcParamSettingType说明：

- RTC_PARAM_SETTINGS_ALL：设置全部paramSettings参数集，一般情况下采用该方式设置全部参数；

RtcParameterSettings 定义

- 通用属性

类型	属性	默认值	描述	说明
boolean	HasAudio	true	是否采集、发送音频流	涉及与服务端音频协商，通信后无法修改。
String	AudioCodec	"opus"	音频编码类型	推荐使用默认值。
int	AudioFrequency	48000	音频采样率	推荐使用默认值。
int	AudioChannel	1	音频通道	推荐使用默认值。
boolean	MicPhoneMuted	false	是否发送本端音频流	与HasAudio不同， 该值仅控制是否发送音频数据，通信后可切换。
int	AudioMaxkbps	-1	音频最大码率	默认值 -1 表示使用自适应码率，推荐使用默认值。
int	AudioSource	VOICE_COMMUNICATION	音频输入源类型	输入源类型定义可参考系统接口：android.media.AudioSource，通信场景推荐使用默认值。
boolean	EnableMultistream	true	开启多流模式	在多人通信场景，多路媒体流复用同一个连接，系统开销更小， 推荐使用。
boolean	AutoPublish	true	自动发布媒体流	房间登录成功后自动发布本端媒体流，设置自动发布后不应再调用startPublish接口手动发布媒体流。
boolean	AutoSubScribe	true	自动订阅媒体流	房间登录成功后自动订阅房间内其它用户媒体流，设置自动订阅后不应再调用subscribeStreaming接口手动发布媒体流。

- 扩展属性

扩展属性推荐使用默认值，您也可以根据具体应用场景选择性使用。

类型	属性	默认值	描述
int	AudioContentType	AudioAttributes.CONTENT_TYPE_SPEECH	设置音频输出类型，完整Audio Content Type定义参考系统接口android.media.AudioAttributes
RtcAudioBitrateMode	audioBitrateMode	RTC_AUDIO_BITRATE_CBR	设置opus音频编码模式：CBR/VBR
int	ConnectionTimeoutMs	5000	信令服务器连接超时时长
int	ReadTimeoutMs	5000	信令读取超时时长
boolean	EnableAudioLevel	true	开启服务端按音频增益混流
int	AudioLevelTopCount	3	与EnableAudioLevel 配合使用,控制服务端转发的音频混流路数（根据音频增益大小）

设置回调代理

```
public abstract boolean setBaiduRtcRoomDelegate(BaiduRtcRoomDelegate baiduRtcRoomDelegate);
```

代理设置。

设置RTC Room代理对象。使用RTC Room功能，初始化相关组件时需要设置代理对象，代理对象用户事件回调。未设置代理对象，或对象设置错误，可能导致无法收到相关回调。

参数

参数	类型	描述
baiduRtcRoomDelegate	BaiduRtcRoomDelegate	遵循BaiduRtcRoomDelegate协议的代理对象

返回

true 成功，false 失败

房间相关接口

登录房间

```
public abstract boolean loginRtcRoomWithRoomName(String roomName, long userId, String displayName, boolean isCompulsive);
```

登录房间成功，在同一个房间的人能进行相互音视频聊天。开启强制登录，若房间内之前存在同一userId用户，将被踢出，建议在断网重连或者初次登录失败时调用。

参数

参数	类型	描述
roomName	String	房间名，长度不可超过 255 byte，不可包括特殊字符及中文
userId	int	用户ID，每个房间的用户ID必须唯一，建议每个用户唯一ID
displayName	String	用户显示名
isCompulsive	boolean	true：强制登录；false：正常登录，默认正常登录

返回

true 成功，false 失败

退出房间

```
public abstract boolean logoutRtcRoom();
```

执行logoutRtcRoom后，会停止音视频采集，断开与房间服务器的连接，取消音视频的传输，销毁音视频传输通道以及释放其他资源。

返回

true 成功，false 失败

启动跨房间通信

```
abstract void startRoomMediaRelay(String destRoomName, long userId, String token)
```

启动跨房间通信。

注意 默认情况下，SDK允许同一房间内的用户间进行通信，若要与其它房间的用户进行通信，则需要需要该接口进行跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	long	加入目标房间时使用的userId，必须在加入房间中是唯一的
token	String	App server派发的token字符串

停止跨房间通信

```
abstract void stopRoomMediaRelay(String destRoomName, long userId, String token)
```

停止跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	long	加入目标房间时使用的userId，必须在加入房间中是唯一的

停止所有跨房间通信

```
public abstract void stopRoomMediaRelayAll()
```

停止所有跨房间通信。

解散房间

```
public abstract boolean disbandRoom();
```

解散房间。 房间管理员有权利解散整个房间，解散后，房间中的每个用户都被动退出房间。

踢出某成员

```
public abstract void kickOffUserWithId(int userId);
```

踢出某用户。

房管／主播／会议主持 把某用户踢出聊天室。

参数

参数	类型	描述
userId	int	在房间中的用户ID

禁言某成员

```
public abstract void shutUpUserWithId(int userId, bool disable);
```

是否禁言某成功。

房管/主播／会议主持 禁止某成功发言。

参数

参数	类型	描述
userId	int	在房间中的用户ID
bool	disable	禁言/取消禁言

发布/订阅相关接口

发布媒体流

```
public abstract void publishStreaming();
```

发布本地媒体流。 仅当RtcParameterSettings.AutoPublish设置为false（非自动发布）时，手动发布本地媒体流。
调用时机：接收到房间登录成功事件BaiduRtcRoomDelegate.RTC_ROOM_EVENT_LOGIN_OK之后；

订阅某成员媒体流

```
public abstract void subscribeStreaming(long feedId);
```

订阅远端成员订阅流。

仅当RtcParameterSettings.AutoSubscribe设置为false（非自动订阅）时，手动订阅远端用户媒体流。

参数

参数	类型	描述
feedId	long	远端用户ID

停止订阅某成员媒体流

```
public abstract void stopSubscribeStreaming(long feedId);
```

停止订阅流，可与subscribeStreaming接口配合使用。

参数

参数	类型	描述
feedId	long	远端用户ID

消息相关接口

向某成员发送消息

```
public abstract void sendMessageToUser(String msg, long id);
```

发送消息。

id=0，房间内广播消息，id = userID，指定用户发送消息。

参数

参数	类型	描述
msg	String	消息
id	long	流的ID号

设置用户属性

```
public abstract void setUserAttribute(String attribute);
```

用户属性设置。

设置用户属性。

参数

参数	类型	描述
attribute	String	用户属性

获取用户属性

```
public abstract void getUserAttribute(long feedid);
```

用户属性获取。

获取用户属性。

参数

参数	类型	描述
feedid	long	流ID

音频相关接口

关闭/开启麦克风静音

```
public abstract void muteMicphone(boolean muted);
```

开启/关闭 麦克风静音（不关闭手机麦克风设备）。

参数

参数	类型	描述
muted	boolean	true：静音，false：取消静音

切换扬声器/听筒

```
public abstract void switchLoundSpeaker();
```

开关扬声器。

听筒与扬声器切换(免提功能)

预置听筒/扬声器

```
public abstract void presetLoudSpeaker(boolean isPresetLoudSpeaker);
```

预置听筒/扬声器。

参数

参数	类型	描述
isPresetLoudSpeaker	boolean	true：扬声器，false：听筒

禁止/开启音频输出

```
public abstract void muteSpeaker(boolean mute)
```

开启/关闭音频输出。

参数

参数	类型	描述
mute	boolean	true：禁止音频输出，false：开启音频输出，默认为false

设置远端用户音频播放音量

```
public abstract void setUserPlaybackVolume(long userId, int volume)
```

设置远端用户音频播放音量。

参数

参数	类型	描述
userId	long	远端用户ID
volume	int	设置的音量范围 [0,100]

查询扬声器是否开启

```
public abstract boolean isSpeakerOn()
```

查询扬声器是否开启。

获取房间成员音量列表

```
public abstract RtcRoomAudioLevel[] getRemoteAudioLevels();
```

房间成员音量接口。

获取用户音量列表。

返回

RtcRoomAudioLevel[]

设置是否拉取某成员音频流

```
public abstract void setRemoteAudioPlayState(boolean stats, long userId);
```

指定远端音频暂停/恢复播放。

通过控制音频流拉取，控制指定远端用户音频暂停/恢复播放。

参数

参数	类型	描述
stats	boolean	true：拉取，false：停止拉取，默认为true
userId	long	用户ID

返回

RtcRoomAudioLevel[]

设置音频输出设备

```
public abstract void setSoundMod(RtcSoundMode soundMod)
```

设置音频输出设备，扬声器/听筒。

参数

参数	类型	描述
soundMod	RtcSoundMode	音频输出方式：扬声器/听筒

转推配置相关接口

配置媒体转推参数

```
public abstract boolean configLiveServerWithUrl(String url,
    boolean isMix,
    boolean isRecording,
    String mixTemplate,
    RtcLiveTransferMode transferMode);
```

server端推流参数配置。

该接口用于配置Server推流的参数，聊天室模式：在同一个RTC房间的所有参与者在混流后，直接转推到一个指定的直播房间；主播转推模式：主播推向不同的直播房间

参数

参数	类型	描述
url	String	rtmp推流地址
isMix	boolean	是否做混流处理
mixTemplate	String	混流模板
transferMode	RtcLiveTransferMode	转推模式： 聊天室模式， 主播转推模式

返回

true 设置成功， false 失败。

通知相关接口

登录房间回调

```
public void onLoginRtcRoom(int code, long roomId)
```

登录房间回调

参数

参数	类型	描述
code	int	结果码，0：成功，101：超时，其他：失败
roomId	long	房间RoomID

链接断开回调

```
public void onConnectionLost(int code)
```

链接断开回调

当音频通话中，发生连接断开，会回调该事件

链接状态变化回调

```
public void onConnectionStateChanged(int newState, int reason)
```

链接状态发生变化回调

参数

参数	类型	描述
newState	int	链接状态，值为 BaiduRtcRoomDelegate.ConnectionState
reason	int	房间RoomID

用户加入房间回调

```
public void onUserJoined(long uid, String displayName)
```

远端用户加入房间回调

参数

参数	类型	描述
uid	long	远端用户UserID
displayName	String	远端用户显示名字

用户离开房间回调

```
public void onUserOffline(long uid, String msg)
```

远端用户离开房间回调

参数

参数	类型	描述
uid	long	远端用户UserID
msg	String	保留使用

远端音频到达回调

```
public void onFirstRemoteAudio(long uid, String msg)
```

远端音频到达回调

参数

参数	类型	描述
uid	long	远端用户UserID
msg	String	保留使用

远端音频流离开回调

```
public void onRemoteAudioOffline(long uid, String msg)
```

远端音频流离开回调

参数

参数	类型	描述
uid	long	远端用户UserID
msg	String	保留使用

远端用户加入房间且推流回调

```
public void onRemoteStreamComing(long uid, String name)
```

远端用户加入房间且推流回调

远端用户加入房间，且开始推流，收到该回调。当收到该回调时，可以开始订阅流

参数

参数	类型	描述
uid	long	远端用户UserID
name	String	用户名

远端流离开回调

```
public void onRemoteStreamLeaving(long uid)
```

远端流离开回调

远端用户停止推流时，收到该回调事件。与onUserOffline 事件区分，用户可能未离开房间，只停止推流。

房间解散事件回调

```
public void onRoomDisband()
```

房间解散事件回调

当房间被解散时，收到该回调事件

禁言状态变化回调

```
public void onUserShutupChanged(long uid, boolean on)
```

禁言状态变化回调

当某用户禁言状态发生变化时，收到该回调事件

参数

参数	类型	描述
uid	long	远端用户UserID
on	boolean	禁言状态，true：被禁言，false：解除禁言

用户被踢出房间回调

```
public void onUserKickOffed(long uid)
```

用户被踢出房间回调

当某用户被踢出房间时，收到该回调

参数

参数	类型	描述
uid	long	远端用户UserID

转推事件回调

```
public void onLiveStreamStateChanged(String url,BaiduRtcRoom.RtcLiveTransferMode mode, int state)
```

转推事件回调

当调用configLiveServerWithUrl接口或者startLiveServerStreaming接口时，会收到该回调事件，通知转推状态

参数

参数	类型	描述
url	String	转推url
mode	RtcLiveTransferMode	转推模式： 聊天室模式， 主播转推模式
state	int	转推状态，0：idle, 1：connected, 2：dis-connected, 3：interrupt

用户信息回调

```
public void onUserMessage(long uid, String msg)
```

用户信息回调

当调用sendMessageToUser接口后，对应用户会收到该事件回调

参数

参数	类型	描述
uid	long	远端用户UserID
msg	String	收到的消息

用户属性回调

```
public void onUserAttribute(long uid, String msg)
```

用户属性回调

当用户设置属性，加入房间后，会收到该用户属性回调

参数

参数	类型	描述
uid	long	远端用户UserID
msg	String	属性

开启跨房间转推回调

```
public void onStartMediaRelayResult(String destRoomName, long userId, int code)
```

跨房间转推回调

当调用startRoomMediaRelay接口后，会收到该事件回调

参数

参数	类型	描述
destRoomName	String	目标房间号
userId	long	加入目标房间下的对应UserID
code	int	开启跨房间转推结果，0：成功，其他：失败

停止跨房间转推回调

```
public void onStopMediaRelayResult(String destRoomName, long userId, int code)
```

停止跨房间转推回调

当调用stopRoomMediaRelay接口后，会收到该事件回调

参数

参数	类型	描述
destRoomName	String	目标房间号
userId	long	加入目标房间下的对应UserID
code	int	结果，0为成功，其他为失败

推流媒体链路事件回调

```
public void onPublisherMediaStateUpdate(int states)
```

媒体链路事件回调

当推流媒体链路发生状态变化时，收到该回调

参数

参数	类型	描述
states	int	状态

state具体值

```
public static final int RTC_STATE_STREAM_CLOSED = 2008; // 链路关闭
public static final int RTC_STATE_PUBLISHER_STREAM_DOWN = 2006; // 推流链路断开
public static final int RTC_STATE_SUBSCRIBER_STREAM_DOWN = 2007; // 订阅链路断开
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL0 = 2100;  // 丢包
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL1 = 2101;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL2 = 2102;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL3 = 2103;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL4 = 2104;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL5 = 2105;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL6 = 2106;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL7 = 2107;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL8 = 2108;
public static final int RTC_STATE_STREAM_SLOW_LINK_LEVEL9 = 2109;
```

订阅媒体链路事件回调

```
public void onSubscriberMediaStateUpdate(int states)
```

订阅媒体链路事件回调

当订阅媒体链路发生状态变化时，收到该回调

参数

参数	类型	描述
states	int	状态

错误事件回调

```
public abstract void onError(/*RtcErrorCodes*/int errorInfo)
```

错误事件回调

当发生不可恢复错误时，收到该回调

参数

参数	类型	描述
errorInfo	int	错误码

ErrorCode 参数

```
public static class ErrorCode {
  /** 无错误 */
  public static final int ERROR_OK = 0;
  /** 连接超时 */
  public static final int ERROR_CONNECT_TIMEOUT = 101;
  /** okHttp onFailure */
  public static final int ERROR_CONNECT_FAIL = 102;
  /** okHttp onFailure SSLException */
  public static final int ERROR_SSL_EXCEPTION = 103;
  /** signal-server send timeout command */
  public static final int ERROR_SERVER_TIMEOUT_CMD = 104;

  /** signal-server find uid already exist */
  public static final int ERROR_USERID_ALREADY_EXIST = 200;
  /** okHttp keep-alive fail */
  public static final int ERROR_CHANNEL_KEEPALIVE_TIMEOUT = 201;

  /** signal message error */
  public static final int ERROR_SIGNAL_SERVER_ERROR = 203;

  /** 人数达到上限/订阅者人数上限，业务处理 */
  public static final int ERROR_EXCEED_SUBSCRIBERS_LIMIT = 230;
  /** 发布者人数上限，业务处理 */
  public static final int ERROR_EXCEED_PUBLISHERS_LIMIT = 241;
  /** 参数错误，业务自己处理 */
  public static final int ERROR_PUBLISHER_PARAMS_INCORRECT = 242;

  /**
   * 用户不存在，业务需要重连，整个重连
   */
  public static final int ERROR_PUBLISHER_USER_NOT_EXISTS = 260;
  public static final int ERROR_SUBSCRIBER_USER_NOT_EXISTS = 261;


  /**
   * 重复推流，上报业务无需处理
   */
  public static final int ERROR_PUBLISHER_REPEAT_PUBLISH = 270;
  public static final int ERROR_SUBSCRIBER_PARAMS_INCORRECT = 271;
  public static final int ERROR_SUBSCRIBER_REPEAT_SUBSCRIBE = 272;

  public static final int ERROR_UNSUBSCRIBE_USER_NOT_EXISTS = 280;
  public static final int ERROR_UNSUBSCRIBE_PARAMS_INCORRECT = 281;
  public static final int ERROR_UNSUBSCRIBE_USER_NOT_SUBSCRIBED = 282;


  /**
   * 用户状态错误，业务处理，需重新订阅
   */
  public static final int ERROR_SUBSCRIBER_USER_STATE_INVALID = 290;
  public static final int ERROR_UNPUBLISH_USER_NOT_EXISTS = 291;

  /** peerconnection runtime error */
  public static final int ERROR_PEER_CONNECTION_RUNTIME = 500;

}
```

警告事件回调

```
public abstract void onWarn(int code)
```

警告事件回调

参数

参数	类型	描述
code	int	错误码

其他接口

获取SDK版本

```
public static String version()
```

SDK版本号获取。

获取当前SDK版本号

返回

返回当前SDK版本号

开启/关闭调试信息输出

```
public static void setVerbose(boolean bOnVerbose)
```

是否打开调试信息。

建议在初始化SDK前调用。建议在调试阶段打开此开关，打开此开关后，将打开日志信息，方便调试。

参数

参数	类型	描述
bOnVerbose	boolean	是否打开调试信息，true：打开，false：关闭。默认为false

开启/关闭质量数据上报

```
public abstract void enableStatsToServer(boolean isEnable, String qualityMonitorEnv)
```

RTC质量监控数据上报。

预置接口，监控信息上报开关 当打开开关时，上报、码率、丢包率等监控信息到服务端，console可查。

参数

参数	类型	描述
isEnable	boolean	是否打开RTC质量监控数据上报，true：打开，false：关闭。默认为false
qualityMonitorEnv	String	online："线上环境" qa："沙盒"。默认值为online

设置本地日志文件配置

```
public static void setWriteLogConfig(WriteLogConfig config)
```

设置日志写本地文件配置

参数

参数	类型	描述
config	WriteLogConfig	本地文件配置

WriteLogConfig 定义

参数	类型	描述
logPath	String	日志文件路径
logName	String	日志文件名，不需要加后缀
maxBytes	String	单个日志最大容器
maxNum	String	最多保存日志文件个数
enableWriteFile	boolean	是否允许写文件，true：打开写入本地文件，false：关闭，默认为false

注意 需要先开启setVerbose(true)才会打开日志

开启/关闭日志自动上传

```
public abstract void enableAutoUploadLog(boolean enable);
```

开启/关闭自动上传日志文件

开启后，会自动上传日志文件到云服务器，方便排查线上问题

参数

参数	类型	描述
enable	boolean	是否开启，true：开启，自动上传云端，false：关闭

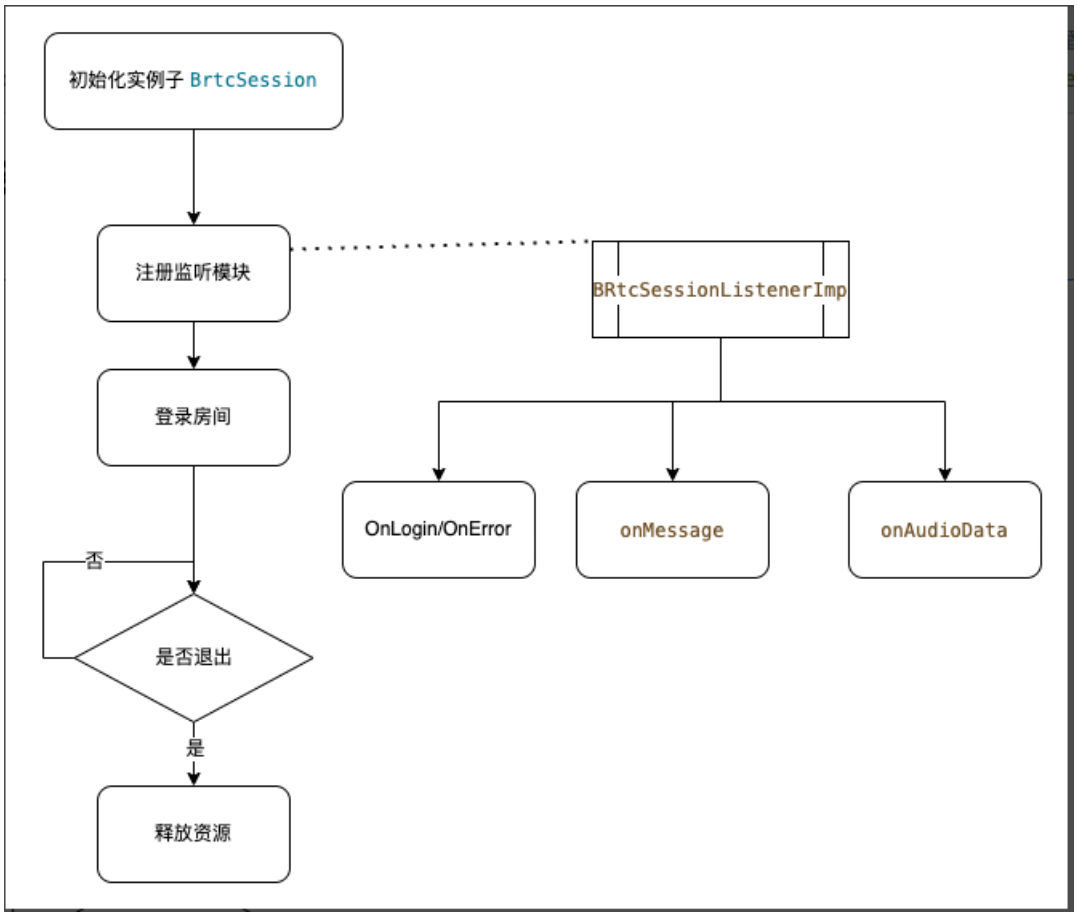
JAVA SDK

概述

RTC Linux端Java快速接入SDK能够帮助您快速集成RTC能力，通过少量代码即可完成视频房间的创建、通信与停止通信、设置音视频参数和设备参数等操作。 您可以更专注于业务创新，RTC的底层能力可以交由百度智能云来提供。 |

API

API调用流程



1 事件对象 BrtcSessionListener 监听事件回调类 1.1 onLogin 介绍

```
void onLogin(long sessionId, long userId);
```

调用login时候返回登录结果事件

参数

参数	类型	描述
sessionId	long	返回sessionID
peerId	long	返回userId

返回

无 **1.2 onRemoteUserJoin**

介绍

```
void onRemoteUserJoin(long sessionId, long peerId);
```

远端用户登录成功回调事件

参数

参数	类型	描述
sessionId	long	返回sessionID
peerId	long	返回userId

返回 无 **1.3 onStreamUp 介绍**

```
void onStreamUp(long sessionId);
```

stream流管道建立成功事件，可以发送音视频数据。

参数

参数	类型	描述
sessionId	long	返回sessionID

返回

无 **1.4 onError 介绍**

```
void onError(long sessionId, int erroCode);
```

错误回调日志 **参数**

参数	类型	描述
sessionId	long	返回sessionID
erroCode	long	返回userId

Coder类型整理：

参数	类型	描述
RTC_MESSAGE_ROOM_EVENT_LOGIN_OK	int	100
RTC_MESSAGE_ROOM_EVENT_LOGIN_TIMEOUT	int	101
RTC_MESSAGE_ROOM_EVENT_LOGIN_ERROR	int	102
RTC_MESSAGE_ROOM_EVENT_CONNECTION_LOST	int	103
RTC_MESSAGE_ROOM_EVENT_REMOTE_COMING	int	104
RTC_MESSAGE_ROOM_EVENT_REMOTE_LEAVING	int	105
RTC_MESSAGE_ROOM_EVENT_REMOTE_RENDERING	int	106
RTC_MESSAGE_ROOM_EVENT_REMOTE_GONE	int	107
RTC_MESSAGE_ROOM_EVENT_SERVER_ERROR	int	108

返回 无 **1.5 onMessage 介绍**

```
void onMessage(long sessionId, long peerId, ByteBuffer messageData, int len);
```

自定义消息通知事件

参数

参数	类型	描述
sessionId	long	返回sessionID
peerId	long	返回userId
messageData	ByteBuffer	自定义消息数据
len	int	长度

返回 无

1.6 onAudioData 介绍

```
void onAudioData(long sessionId, long peerId, ByteBuffer data, int len);
```

音频数据回调 **参数**

参数	类型	描述
sessionId	long	返回sessionID
peerId	long	返回userId
data	ByteBuffer	自定义消息数据
len	int	长度

返回 无 **2 配置参数对象 UserParams 介绍** 设置参数。

设置应用的AppID， 需要从百度智能云申请得到。

参数

参数	类型	描述
SeverSignal	string	对应得信令地址
AppID	string	应用的AppID
cerPath	string	cer证书文件的全路径
forwardUrl	string	转推的地址可以为空
roomId	long	房间号
userId	long	用户ID
videoCode	string	视频编码Video 264
audioChannel	int	音频参数声道
audioSample	int	音频采样率
isPublisher	boolean	是否推流者身份
isListener	boolean	是否订者阅身份
autoSub	boolean	是否支持自动订阅模式
enableLog	boolean	是否支持日志

返回 无 **3 会话对象 BrtcSession**

介绍 BrtcSession brtcSession = new BrtcSession(); 创建BrtcSession对象 用户首先需要创建 BrtcSession, 通过成员函数方法进行初始化，发布流，订阅流等其他可提供的api 操作。

3.1 setSessionListener 介绍

```
void setSessionListener(BrtcSessionListener brtcSessionListener)
```

绑定事件监听类 **参数**

参数	类型	描述
brtcSessionListener	BrtcSessionListener	void

返回 无 **3.2 removeSessionListener**

介绍

```
void removeSessionListener()
```

解绑定事件监听类

返回 无

3.3 login

介绍

```
long login(UserParams userParams)
```

登录房间

参数 | 参数 | 类型 | 描述 | | --- | --- | --- | |userParams| UserParams |参数对象构建|

返回 无 **3.4 logOut 介绍**

```
boolean logOut()
```

登录房间**参数**

参数	类型	描述
userParams	UserParams	参数对象构建

返回 无 **3.5 subscribeUser**

```
void subscibeUser(long feedId);
```

介绍 手动订阅远端用户流 **参数**

参数	类型	描述
feedId	long	订阅远端用户媒体数据，默认返回音频

返回 无 **3.6 unsubscribeUser**

```
void unsubscibeUser(long feedId);
```

介绍

手动停止订阅远端用户流

参数

参数	类型	描述
feedId	long	订阅远端用户媒体数据，默认返回音频

返回

无 **3.7 destory**

```
void destory()
```

介绍

销毁内部资源

返回

无

Flutter SDK

概述

概述 本文档主要介绍如何将RTC Flutter SDK集成到您的项目中，并介绍各类功能的使用方法。在使用本文档前，您需要先了解RTC的一些基本知识，并已经开通了RTC服务。若您还不了解RTC，可以参考产品描述。

RTC Flutter SDK 能够帮助您实现登录音视频通信房间并开始通信，当前可支持Android/ iOS接入。

接口概览 RTC Flutter SDK 提供以下接口。

初始化接口 | API | 描述 | |---| --- | |getInstance| 静态方法，获取Future<BaiduRtcCloud?> 实例 | |destroyInstance| 静态方法，销毁Future<BaiduRtcCloud?> 实例 | |registerListener| 注册监听器，用于事件回调 | |unRegisterListener| 取消注册监听器 |

房间相关接口 | API | 描述 | | --- | --- | |loginBRtcRoom| 登录房间 | |logoutRtcRoom| 登出房间 | |setParamSettings| 设置房间参数 | |startRoomMediaRelay| 启动跨房间通信 | |stopRoomMediaRelay| 停止跨房间通信 | |stopRoomMediaRelayAll| 停止所有跨房间通信 | |disbandRoom| 解散房间 | |kickOffUserWithId| 踢出某成员| |shutUpUserWithId|禁言某成员| |unShutUpUserWithId|解除禁言成员|

发布/订阅流相关接口 | API | 描述 | | --- | --- | | startPublish |开始实时音视频推流| | stopPublish |停止实时音视频推流| | subscribeStreaming |订阅某成员媒体流| | subscribeStreamingWithViewId |订阅实时音视频流，同时绑定View用于渲染| | stopSubscribeStreaming | 停止订阅某成员媒体流 |

消息相关接口 | API | 描述 | | - | - | |sendMessageToUser|向某成员发送消息| |setUserAttribute|设置用户属性| |getUserAttribute|获取用户属性| |sendData|发送数据|

视频相关接口 | API | 描述 | | --- | --- | | startPreview |开启本地预览| | stopPreview | 停止本地预览| |setLocalDisplay|设置本地渲染窗口| |setRemoteDisplay| 设置远端渲染窗口 | |setRemoteDisplayWithUserId| 设置远端渲染窗口，同时绑定UserId，BRTCVideoViewController类控制 | |updateDisplay| 更新远端渲染窗口，BRTCVideoViewController类控制 | |removeRemoteDisplay| 移除远端用户渲染窗口 | |changeSurfaceSize| 更新外部渲染Surface尺寸，BRTCVideoViewController类控制 | |setRemoteVideoPlayState| 设置是否拉取某成员视频流 |

摄像头相关接口 |API|描述| |-|-| |switchCamera|切换摄像头| |setCameraID|设置摄像头ID| |muteCamera|关闭/开启本地视频采集|

音频相关接口 | API | 描述 | | --- | --- | | muteMicphone |设置是否静音采集麦克风数据（不关闭麦克风）| |enableMicCapture|设置是否使用麦克风采集音频（关闭麦克风）| |switchLoundSpeaker| 切换扬声器/听筒 | |presetLoudSpeaker|预置听筒/扬声器| |muteSpeaker|禁止/开启音频输出| |setUserPlaybackVolume|设置远端用户音频播放音量| |isSpeakerOn|查询扬声器是否开启| |setSoundMod|设置音频输出设备| | enableAgc | 开启默认音频自动增益 | | enableAns | 开启默认音频自动噪声抑制 | | enableAec | 开启默认音频回声消除 | | enableAudioMix | 开启/关闭 音频混音 |

通知相关接口 | API | 描述 | | --- | --- | |onRoomEventUpdate| 房间事件更新回调 | |onPeerConnectStateUpdate| 连接状态更新回调 | |onStreamInfoUpdate| 媒体流信息更新回调 | |onErrorInfoUpdate| 错误信息更新回调 | |onRoomDataMessage| 房间数据接收回调 |

其它接口 | API | 描述 | |-| - | |setVerbose| 开启/关闭调试信息输出 | |enableErrorInfoToServer|开启/关闭错误数据上报| |enableStatsToServer|开启/关闭质量数据上报|

API

初始化接口 获取SDK实例

```
static Future<BaiduRtcCloud?> getInstance(BRTCInitParams params);
```

静态方法，获取BaiduRtcCloud 实例。

参数

参数	类型	描述
params	BRTCInitParams	参数

BRTCInitParams 结构

字段	属性	描述
appId	String	APP ID
token	String	token
cpuType	String	cpu 类型
isEnableSoLaterLoad	bool	是否开启so文件后下载

返回

BaiduRtcCloud 实例对象

销毁SDK实例

```
static Future<void> destroyInstance();
```

静态方法，销毁BaiduRtcCloud 实例。

参数

无

返回

Void

注册监听器 void registerListener(ListenerValue func);

注册事件回调。

参数

参数	类型	描述
func	ListenerValue	参数

ListenerValue 结构

```
void Function(BaiduRTCRoomDelegate type, int event, P? params);
```

BaiduRTCRoomDelegate 结构

```
  enum BaiduRTCRoomDelegate {
    onRoomEventUpdate,
    onPeerConnectStateUpdate,
    onStreamInfoUpdate,
    onErrorInfoUpdate,
    onEngineStatisticsInfo,
    onRoomDataMessage,
}
```

BaiduRTCRoomDelegate 通知回调描述，查看通知相关接口

返回

Void

取消注册监听器 void unRegisterListener(ListenerValue func);

取消注册事件回调。

参数

参数	类型	描述
func	ListenerValue	参数

返回

Void

房间相关接口

登录房间

```
  Future<bool?> loginBRtcRoom(BRTCLoginParams params);
```

登录房间。

登录成功后同一个房间的的成员能够互相看到和听到。

参数

参数	类型	描述
params	BRTCLoginParams	登录参数

BRTCLoginParams结构

参数	类型	描述
roomId	String	房间名，长度不可超过 255 byte
userId	String	用户ID，每个房间的用户ID必须唯一
displayName	String	用户显示名
isCompulsive	bool	是否强制登录

返回

true 成功，false 失败

注意

如果失败，会通过错误信息更新回调返回错误信息。

登出房间

```
Future<bool?> logoutRtcRoom();
```

退出房间。

执行logoutRtcRoom后，会停止音视频采集，断开与房间服务器的连接，取消音视频的传输，销毁音视频传输通道以及释放其他资源。

返回

true 成功，false 失败

设置媒体参数

```
Future<void> setParamSettings(BRTCCommonParam param);
```

音视频参数设置。

设置音视频相关的参数。该函数在登录房间前调用，主要用于设置音视频采集，编解码相关的参数。

参数

参数	类型	描述
param	BRTCCommonParam	该参数封装了音视频的一些参数,如video分辨率，fps, bitrate，音频采样率等

BRTCCommonParam 结构

- 通用属性

类型	属性	默认值	描述
bool	hasVideo	true	【含义】是否采集、发送视频流 【说明】涉及与服务端视频协商，通信后无法修改。开启后采集摄像头媒体数据。
bool	hasAudio	true	【含义】是否采集、发送音频流 【说明】涉及与服务端音频协商，通信后无法修改。
bool	hasScreen	false	【含义】是否采集、发送屏幕共享 【说明】涉及与服务端音频协商，通信后无法修改。
bool	hasData	false	【含义】是否开启数据通道 【说明】涉及与服务端数据通道协商，通信后无法修改。开启可向其它用户发送数据。
String	videoCodec	"h264"	【含义】视频编码类型 【说明】推荐使用默认值。
String	audioCodec	"opus"	【含义】音频编码类型 【说明】推荐使用默认值。
int	audioFrequency	48000	【含义】音频采样率 【说明】推荐使用默认值。

int	inputAudioChannel	1	【含义】输入音频通道 【说明】推荐使用默认值。
int	outputAudioChannel	1	【含义】输出音频通道 【说明】推荐使用默认值。
int	videoWidth	640	【含义】发送视频宽 【说明】推荐使用32位对齐的视频采集宽度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。
int	videoHeight	480	【含义】发送视频高 【说明】推荐使用32位对齐的视频采集高度，由于部分平台兼容性问题，非32位对齐可能存在视频编解码异常。
int	videoFps	20	【含义】视频初始帧率 【说明】通信时，实际帧可能会根据当前带宽上下波动。
bool	micPhoneMuted	false	【含义】是否发送本端音频流 【说明】与HasAudio不同， 该值仅控制是否发送音频数据，通信后可切换。
bool	cameraMuted	false	【含义】是否发送本端视频流 【说明】与HasVideo不同， 该值仅控制是否发送视频数据，通信后可切换
int	videoMaxkbps	1000	【含义】视频最高码率 【说明】通信时，实际发送的视频码率不超过最高码率。
int	videoMinkbps	10	【含义】视频最低码率 【说明】通信时，实际发送的视频码率不低于最低码率。
int	audioMaxkbps	-1	【含义】音频最大码率 【说明】默认值 -1 表示使用自适应码率，推荐使用默认值。
int	audioSource	VOICE_COMMUNICATION	【含义】音频输入源类型 【说明】输入源类型定义可参考系统接口：android.media.AudioSource，通信场景推荐使用默认值。
bool	enableMultistream	true	【含义】开启多流模式 【说明】在多人通信场景，多路媒体流复用同一个连接，系统开销更小， 推荐使用。
bool	autoPublish	true	【含义】自动发布媒体流 【说明】 房间登录成功后自动发布本端媒体流，设置自动发布后不应再调用startPublish接口手动发布媒体流。
bool	autoSubScribe	true	【含义】自动订阅媒体流 【说明】房间登录成功后自动订阅房间内其它用户媒体流，设置自动订阅后不应再调用subscribeStreaming接口手动订阅媒体流。

- 扩展属性

扩展属性推荐使用默认值，您也可以根据具体应用场景选择性使用。

类型	属性	默认值	描述
int	audioContentType	AudioAttributes.CONTENT_TYPE_SPEECH	【含义】设置音频输出类型，完整Audio Content Type定义参考系统接口 android.media.AudioAttributes
RtcVideoRenderMode	VideoRenderMode	RTC_VIDEO_RENDER_MODE_INTERNAL	【含义】设置渲染模式：内部渲染/外部渲染
bool	enableFixedResolution	false	【含义】是否使用固定分辨率
bool	enableHisiH264HW	true	【含义】是否打开Hisi平台H.264硬件编解码
bool	enableMTKH264Decode	true	【含义】是否打开MTK平台H.264硬件解码
bool	enableAacCodec	false	【含义】是否开启AAC 解码 (仅低延时播放场景)
bool	enableJitterRetransmission	false	【含义】是否开启平滑渲染，可优化下行丢包
int	encodeBitrateMode	RTC_VIDEO_CONTROLRATECONSTANT	【含义】设置编码模式
int	audioBitrateMode	0	【含义】设置opus音频编码模式：CBR(0) / VBR(1);
int	connectionTimeoutMs	5000	【含义】信令服务器连接超时时长
int	readTimeoutMs	5000	【含义】信令读取超时时长
bool	enableAudioLevel	false	【含义】开启服务端按音频增益混流
int	audioLevelTopCount	3	【含义】与EnableAudioLevel 配合使用,控制服务端转发的音频混流路数（根据音频增益大小）

启动跨房间通信

```
Future<void> startRoomMediaRelay(String destRoomName, int userId, String token);
```

启动跨房间通信。

注意 默认情况下，SDK允许同一房间内的用户间进行通信，若要与其它房间的用户进行通信，则需要需要该接口进行跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	int	加入目标房间时使用的userId，必须在加入房间中是唯一的
token	String	App server 派发的token字符串

停止跨房间通信

```
Future<void> stopRoomMediaRelay(String destRoomName, int userId);
```

停止跨房间通信。

参数

参数	类型	描述
destRoomName	String	目标房间
userId	int	加入目标房间时使用的userId，必须在加入房间中是唯一的

停止所有跨房间通信

```
Future<void> stopRoomMediaRelayAll();
```

停止所有跨房间通信。

解散房间

```
Future<void> disbandRoom();
```

解散房间。 房间管理员有权利解散整个房间，解散后，房间中的每个用户都被动退出房间。

踢出某成员

```
Future<void> kickOffUserWithId(int userId);
```

踢出某用户。

房管／主播／会议主持 把某用户踢出聊天室。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户ID

禁言某成员

```
Future<void> shutUpUserWithId(int userId);
```

禁言某人。

房管/主播／会议主持 禁止某人发言。

参数

参数	类型	描述
userId	int	在房间中的用户的用户ID

解除禁言某成员

```
Future<void> unShutUpUserWithId(int userId);
```

解除禁言某成员。

房管/主播／会议主持 解除禁止某成员发言。

参数

参数	类型	描述
userId	int	在房间中的用户的 用户ID

发布/订阅相关接口

开始直播推流

```
Future<void> startPublish();
```

开始直播推流。

发布媒体流，开始推流。
调用时机：接收到房间登录成功事件 BRtcEvents.BRTC_ROOM_EVENT_LOGIN_OK之后；

注意 该接口仅直播场景使用。与stopPublish接口配对使用。

停止直播推流

```
Future<void> stopPublish();
```

停止直播推流。

停止发布流，结束直播。

注意 该接口仅直播场景使用。与startPublish接口配对使用。

订阅某成员媒体流 (1/2)

```
Future<void> subscribeStreaming(int userId);
```

订阅远端成员订阅流。

仅当BRTCCommonParam.autoSubscribe设置为false（非自动订阅）时，手动订阅远端用户媒体流。

参数

参数	类型	描述
userId	int	远端用户ID

订阅某成员媒体流 (2/2)

```
Future<void> subscribeStreamingWithViewId(int videoviewidx, int userId);
```

订阅远端成员订阅流，同时绑定View用于渲染。

仅当BRTCCommonParam.autoSubscribe设置为false（非自动订阅）时，手动订阅远端用户媒体流。

参数

参数	类型	描述
videoviewidx	int	用于渲染画面的View id
userId	int	远端用户ID

停止订阅某成员媒体流

```
Future<void> stopSubscribeStreaming(int userId);
```

停止订阅。

停止订阅流。

参数

参数	类型	描述
userId	int	远端用户ID

消息相关接口

向某成员发送消息

```
Future<void> sendMessageToUser(String msg, int userId);
```

发送消息。

参数

参数	类型	描述
msg	String	消息
userId	int	用户id

设置用户属性

```
Future<void> setUserAttribute(String attribute);
```

用户属性设置。

设置用户属性。

参数

参数	类型	描述
attribute	String	用户属性

获取用户属性

```
Future<void> getUserAttribute(String userId);
```

用户属性获取。

获取用户属性。

参数

参数	类型	描述
userId	int	用户id

发送数据

```
Future<void> sendData(String data);
```

发送Data数据。

Data 数据通道保证顺序性和完整性。 使用数据通道能够同时传输音视频之外的相关数据。

参数

参数	类型	描述
data	String	要发送的数据

注意 发送Data 数据， 需要先打开datachannel 功能，设置BRTCCommonParam 的hasData 字段为true。

视频相关接口

开启本地预览

```
Future<void> startPreview();
```

本地预览。

打开camera，开始预览。

停止本地预览

```
Future<void> stopPreview();
```

停止预览。

关闭camera, 停止本地预览。

设置本地渲染窗口

```
Future<void> setLocalDisplay(int viewId);
```

设置本地视频预览窗口。

设置本地相机预览窗口。 外部采集模式下勿需调用。 调用时机：登录房间之前。

参数

参数	类型	描述
viewId	int	本地显示窗口,用于显示camera采集的视频数据

设置远端渲染窗口

```
Future<void> setRemoteDisplay(int viewId);
```

设置远端显示窗口。

适用于1v1场景下，使用一个BRTCCloudVideoView 拉取一个远端用户视频流。

调用时机： 登录房间之前。

参数

参数	类型	描述
viewId	int	远端画面显示窗口，用于显示远端用户传输过来的视频数据

设置远端渲染窗口，并绑定View

```
Future<void> setRemoteDisplayWithUserId(int viewId, int userId);
```

设置指定远端用户，绑定显示窗口。

可适用于1 v 1、1 v N等多种场景，支持BRTCCloudVideoView与userId绑定。 调用时机：推荐在BRtcEvents.BRTC_ROOM_EVENT_REMOTE_COMING事件回调处调用。

参数

参数	类型	描述
viewId	int	远端画面显示窗口，用于显示远端用户传输过来的视频数据
userId	int	远端成员ID

更新远端渲染窗口

```
Future<void> updateDisplay(int viewId, int userId);
```

更新远端用户显示窗口。

参数

参数	类型	描述
viewId	int	远端画面显示窗口，用于显示远端用户传输过来的视频数据
userId	int	远端用户id

移除远端用户渲染窗口

```
Future<void> removeRemoteDisplay(int userId);
```

移除远端用户渲染窗口。

移除并释放远端用户所对应的渲染窗口，可通过setRemoteDisplayWithUserId(int viewId, int userId) 接口再次设置渲染窗口。若不主动调用该接口释放渲染,则相应的渲染视图将在远端用户退出或本端登出房间时释放。

注意

窗口视图不见时，通过该接口释放渲染视图可优化系统资源占用。

参数

参数	类型	描述
userId	int	用户ID

更新渲染窗口Surface尺寸

```
Future<void> changeSurfaceSize(int width, int height);
```

BRTCVideoViewController 类成员方法，可控制 BRTCCloudVideoView 显示视频的宽高。

参数

参数	类型	描述
width	int	宽
height	int	高

指定远端画面暂停/恢复播放

```
Future<void> setRemoteVideoPlayState(bool stats, int userId);
```

指定远端用户 视频流暂停 / 恢复播放。如暂停则不下发视频流。

参数

参数	类型	描述
stats	bool	true播放 false暂停
userId	int	远端用户 id

摄像头相关接口

切换摄像头

```
Future<void> switchCamera();
```

切换摄像头。

前/后摄像头切换。

设置摄像头ID

```
Future<void> setCameraID(int cameraId);
```

设置指定的摄像头ID。 调用时机： 登录房间之前。

参数

参数	类型	描述
cameraId	int	摄像头ID

关闭/开启本地视频采集

```
Future<void> muteCamera(bool muted);
```

关闭/打开摄像头。

在关闭摄像头后，本地无法预览，且不传输本地视频数据给对方。

参数

参数	类型	描述
muted	bool	是否打开摄像头

音频相关接口

关闭/开启麦克风静音 Future muteMicphone(bool muted);

开启/关闭 麦克风静音（不关闭手机麦克风设备）。

如果需要关闭麦克风设备，请查看enableMicCapture接口。

参数

参数	类型	描述
muted	bool	true表示要静音， false表示取消静音

关闭/开启麦克风采集 Future enableMicCapture(bool enableMic);

开启/关闭 麦克风采集（关闭手机麦克风设备）。

参数

参数	类型	描述
enableMic	bool	true表示要打开麦克风设备采集， false表示关闭麦克风设备采集

切换扬声器/听筒

```
Future<void> switchLoundSpeaker();
```

开关扬声器。

听筒与扬声器切换(免提功能)

预置听筒/扬声器

```
Future<void> presetLoudSpeaker(bool isPresetLoudSpeaker);
```

预置听筒/扬声器。

参数

参数	类型	描述
isPresetLoudSpeaker	bool	true表示扬声器，false表示听筒

禁止/开启音频输出

```
Future<void> muteSpeaker(bool muted);
```

开启/关闭音频输出。

参数

参数	类型	描述
mute	bool	true：禁止音频输出， false：开启音频输出 （默认）

设置远端用户音频播放音量

```
Future<void> setUserPlaybackVolume(int userId, int volume);
```

设置远端用户音频播放音量。

参数

参数	类型	描述
userId	int	远端用户id
volume	int	设置的音量范围 [0,400] ，>100可增益

查询扬声器是否开启

```
Future<bool?> isSpeakerOn();
```

查询扬声器是否开启。

设置音频输出设备

```
Future<void> setSoundMod(int soundMod);
```

设置音频输出设备，扬声器/听筒。

参数

参数	类型	描述
soundMod	int	音频输出方式：扬声器/听筒

开启默认音频自动增益

```
Future<void> enableAgc(bool isAgc);
```

开启音频自动增益。

参数

参数	类型	描述
isAgc	bool	true:开启 false:关闭 默认 true

开启默认音频自动噪声抑制

```
Future<void> enableAns(bool isAns);
```

开启默认音频自动噪声抑制。

参数

参数	类型	描述
isAns	bool	true:开启 false:关闭 默认 true

开启默认音频回声消除

```
Future<void> enableAec(bool isAec);
```

开启默认音频回声消除。

参数

参数	类型	描述
isAec	bool	true:开启 false:关闭 默认 true

开启/关闭混音

```
Future<void> enableAudioMix(bool isEnable) ;
```

开启/关闭混音。

参数

参数	类型	描述
isEnable	bool	开启/关闭混音

通知相关接口

房间事件更新回调

```
public void onRoomEventUpdate(int roomEvents, long data, String extra_info);
```

房间用户状态信息通知。

包含本端登录、远端用户进出房间、房间管理、房间消息、房间通信质量等相关状态回调事件。

参数

参数	类型	描述
roomEvents	int	房间事件信息
data	long	一般为 userID，特殊情况另做说明
extra_info	String	额外信息说明

BRtcEvents 房间事件定义

事件	值	含义	备注
BRTC_ROOM_EVENT_LOGIN_OK	100	登录房间成功	手动发布/订阅相关的调用需要在接收到房间登录成功回调之后。
BRTC_ROOM_EVENT_LOGIN_TIMEOUT	101	登录房间超时	不可恢复错误，可进行重连。
BRTC_ROOM_EVENT_LOGIN_ERROR	102	登录房间出错	不可恢复错误，可确认鉴权信息无误后进行重连。
BRTC_ROOM_EVENT_WS_CONNECTION_ERROR	103	与信令服务器连接断开	不可恢复错误，可进行重连。
BRTC_ROOM_EVENT_REMOTE_COMING	104	开始订阅远端流	无
BRTC_ROOM_EVENT_REMOTE_LEAVING	105	远端视频正在离开	无
BRTC_ROOM_EVENT_REMOTE_RENDERING	106	远端视频流到达	无
BRTC_ROOM_EVENT_REMOTE_GONE	107	远端视频流断开	无
BRTC_ROOM_EVENT_REMOTE_AUDIO_ARRIVED	108	远端音频流到达	无
BRTC_ROOM_EVENT_REMOTE_AUDIO_REMOVED	109	远端音频流断开	无
BRTC_ROOM_EVENTS_DISBAND_ROOM	112	解散房间	无
BRTC_ROOM_EVENTS_SOMEBODY_SHUTUPED	113	禁言	无
BRTC_ROOM_EVENTS_SOMEBODY_DISSHUTUPED	114	解禁	无
BRTC_ROOM_EVENTS_SOMEBODY_KICKOFFED	115	踢人	无
BRTC_ROOM_EVENTS_LIVE_PUBLISH_SUCCESS	116	转推成功	无
BRTC_ROOM_EVENTS_LIVE_PUBLISH_FAIL	117	转推失败	无
BRTC_ROOM_EVENTS_LIVE_INTRERRUPT	118	转推中断	无
BRTC_ROOM_EVENT_AVAILABLE_SEND_BITRATE	200	当前发送端可用上行带宽	无
BRTC_ROOM_EVENT_ON_USER_JOINED_ROOM	300	用户进入房间	无
BRTC_ROOM_EVENT_ON_USER_LEAVING_ROOM	301	用户离开房间	无
BRTC_ROOM_EVENT_ON_USER_MESSAGE	302	消息	无
BRTC_ROOM_EVENT_ON_USER_ATTRIBUTE	303	用户属性	无

实现源码可参考Rtc Flutter Demo;

连接状态更新回调

```
public void onPeerConnectStateUpdate(boolean uplink, int state, String extinfo);
```

server连接状态通知。

当与server端连接状态变化时，通知业务层做后续处理。

- `uplink` 标记上行和下行
- `state` 代表不同状态
- `extinfo` 根据不同`state` 表示不同含义

参数

参数	类型	描述
connecStates	int	连接状态码
BRtcEvents.BRTC_STATE_STREAM_UP	2000	推流成功、拉流成功
BRtcEvents.BRTC_STATE_SENDING_MEDIA_OK	2001	远端流到来
BRtcEvents.BRTC_STATE_SENDING_MEDIA_FAILED	2002	远端流离开
BRtcEvents.BRTC_STATE_STREAM_DOWN	2003	ice失败
BRtcEvents.BRTC_STATE_ICE_CONNECTED	2004	ice链接成功
BRtcEvents.BRTC_STATE_ICE_DISCONNECTED	2005	ice链接断开
BRtcEvents.BRTC_STATE_STREAM_CLOSED	2008	流关闭

BRtcEvents 连接状态事件定义

事件	值	含义	备注
RTC_STATE_STREAM_UP	2000	媒体流建立	无
RTC_STATE_SENDING_MEDIA_OK	2001	媒体流发送成功	无
RTC_STATE_SENDING_MEDIA_FAILED	2002	媒体流发送失败	媒体服务器一定时间内未曾接收到媒体流，可检测网络连接后进行重连。
RTC_STATE_STREAM_DOWN	2003	媒体流断开	媒体服务器指示媒体流断开，可检测网络连接后进行重连。

媒体流信息更新回调

```
public abstract void onStreamInfoUpdate(String [] streamId);
```

媒体流信息通知。

当有远端或本地流信息到来时，通知业务层做后续处理。

参数

参数	类型	描述
streamId	string数组	流信息ID

回调事件

媒体流ID。

错误信息更新回调

```
public abstract void onErrorInfoUpdate(int errorInfo);
```

错误信息通知。

RTC通信过程中，错误信息的反馈。

参数

参数	类型	描述
errorInfo	int	错误信息

BRtcEvents 错误事件定义

事件	值	含义	备注
BRTC_ROOM_SERVER_KEEPALIVE_TIMEOUT_ERROR	402	信令心跳超时	需检查连接网络状态
BRTC_ROOM_SERVER_SIGNAL_ERROR	403	信令错误	检查信令调用
BRTC_ROOM_USERID_ALREADY_EXIST_ERROR	436	房间内已存该用户ID	不可恢复错误，可修改userId后进行重连。 也可以使用强制登录接口进行强制登录。
BRTC_ROOM_PEER_CONNECTION_ERROR	10000	媒体通道连接错误	不可恢复错误，可进行重连。

房间数据接收回调

```
public abstract void onRoomDataMessage(ByteBuffer data);
```

数据接收。

该callback返回当前RTC Engine收到的数据消息。

参数

参数	类型	描述
data	ByteBuffer	引擎返回的数据消息

其他接口

返回

返回当前SDK版本号

开启/关闭调试信息输出

```
Future<void> setVerbose(bool bOnVerbose);
```

是否打开调试信息。

建议在初始化 SDK 前调用。建议在调试阶段打开此开关，打开此开关后，将打开日志信息，方便调试。

参数

参数	类型	描述
bOnVerbose	bool	是否打开调试信息，true 打开，false 不打开。默认为 false

开启/关闭质量数据上报

```
Future<void> enableStatsToServer(bool isEnable, String qualityMonitorEnv);
```

RTC质量监控数据上报。

预置接口，监控信息上报开关 当打开开关时，上报帧率、码率、分辨率、丢包率等监控信息到服务端，console可查。

参数

参数	类型	描述
isEnable	bool	是否打开RTC质量监控数据上报，true 打开，false不打开。默认为 false
qualityMonitorEnv	String	线上环境："online" 沙盒："qa"。 默认值为 "online"

快速开始

环境准备 在实现Flutter 音视频通话前，需要有以下准备：

- VS Code 或其他你选择的IDE
- Android Studio/ Xcode（Android/iOS SDK）
- Flutter SDK (Flutter 官网下载），官方Demo使用Flutter 版本2.10.3；Pod版本1.11.3
- 开发语言：dart, java/kotlin(Android), oc/swift(iOS)，
- 申请百度智能云官网 APP ID，详见https://cloud.baidu.com/product/rtc.html （Demo 中有可用于测试的默认APP ID，仅可用于测试使用）
- 百度RTC Flutter 下载，详见https://cloud.baidu.com/doc/RTC/s/llgunowoj

创建项目 可以使用官网下载的Demo进行参考，也可自行创建项目导入SDK。下面是快速实现音视频通话的几个步骤：

创建工程

```
flutter create brtc_flutter_demo
```

找到 pubspec.yaml 文件。在该文件中，添加以下依赖项

```
dependencies:
flutter:
  sdk: flutter

// 依赖的百度rtc flutter sdk，后续更新可修改版本号
bdcloud_rtc: ^1.0.0
```

注意：在添加包的时候要注意缩进。

在项目文件夹中，运行以下命令来安装所有的依赖项:

```
flutter pub get
```

添加Android 依赖项 当前flutter sdk 已自动引入Android 依赖项，同步项目即可通过maven引入

添加iOS 依赖项 通过下载iOS SDK，将framework文件保存到Flutter工程中，ios/Frameworks 目录下

使用Flutter SDK实现音视频通话

初始化获取 BaiduRtcCloud 实例

```
 BRTCInitParams initParams = new BRTCInitParams();
initParams.appId = appId;
BaiduRtcCloud baiduRtcCloud = (await BaiduRtcCloud.getInstance(initParams))!;
```

配置参数

```
 // 设置参数，默认，可修改
BRTCCommonParam commonParam = BRTCCommonParam();
commonParam.autoPublish = true;
commonParam.autoSubscribe = true;
commonParam.enableMultistream = true;
commonParam.encodeBitrateMode = 1;
commonParam.videoResolution = BdUtils.getResolutionConfig();
commonParam.videoMaxkbps = BdUtils.getBitrateConfig();
commonParam.videoMinkbps = BdUtils.getBitrateConfig();
commonParam.videoFps = BdUtils.getFpsConfig();
commonParam.connectionTimeoutMs = 5000;
commonParam.readTimeoutMs = 5000;
commonParam.inputAudioChannel = 1;
commonParam.outputAudioChannel = 1;
commonParam.audioFrequency = 48000;
commonParam.enableJitterRetransmission = true;
commonParam.enableFixedResolution = true;
commonParam.videoWidth = 480;
commonParam.videoHeight = 640;
commonParam.isMultiPlayerModel = true;

baiduRtcCloud.setParamSettings(commonParam);
```

设置Widget 针对1v1场景，在StatefulWidget 的 Widget build(BuildContext context)方法中，设置预览摄像头Widget 和播放远端画面Widget

```
 @override
Widget build(BuildContext context) {
 final double distance = 12;
 List<String> remoteUidList = remoteUserIdMap.values.toList();
 subscriberCount = remoteUidList.length;
 return Stack(
   alignment: Alignment.topLeft,
   fit: StackFit.expand,
   children: [
     Positioned(
       top: 0,
       left: 0,
       child: Container(
         width: 1.sw,
         height: (4 / 3).sw,
         color: Colors.black,
         child: BRTCCloudVideoView(
```

```
        key: ValueKey("LocalView"),
        viewType: BRTCCloudConfig.BRTC_Video_SurfaceView_Channel,
        videoParam: BRTCVideoParams(
          x: 0,
          y: 0,
          width: 1.sw.toInt(),
          height: (4 / 3).sw.toInt(),
          isLocal: true,
        ),
        onViewCreated: (viewId) async {
          setState(() {
            localViewId = viewId;
          });
          baiduRtcCloud.setLocalDisplay(viewId);
          baiduRtcCloud.startPreview();
        },
      ),
    ),
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.end,
    mainAxisSize: MainAxisSize.min,
    children: [
      Container(
        alignment: Alignment.topRight,
        color: Colors.transparent,
        width: (1 / 3).sw,
        child: GridView.builder(
          itemCount: subscriberCount,
          shrinkWrap: false,
          gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 1,
              mainAxisSpacing: 2.0,
              crossAxisSpacing: 5.0,
              childAspectRatio: 0.75),
          itemBuilder: (BuildContext context, int index) {
            return Container(
              child: BRTCCloudVideoView(
                key: ValueKey('RemoteView_$index'),
                viewType: BRTCCloudConfig.BRTC_Video_Remote_View_Channel,
                videoParam: BRTCVideoParams(
                  x: 0,
                  y: 0,
                  width: (1 / 3).sw.toInt(),
                  height: (4 / 9).sw.toInt(),
                  isLocal: false,
                ),
                onViewCreated: (viewId) async {
                  String userId = index.toString();
                  if (remoteUidList.length > index) {
                    userId = remoteUidList[index];
                  }

                  if (remoteUserViewMap.containsKey(userId)) {
                    baiduRtcCloud.updateDisplay(
                        viewId, int.parse(userId));
                  } else {
                    baiduRtcCloud.setRemoteDisplayWithUserId(
                        viewId, int.parse(userId));
                    remoteUserViewMap[userId] = viewId;
                  }
                },
```

```
                ),
              );
            },
          ),
        ),
      ],
    ),
  ],
 );
}
```

登录房间

```
 BRTCLoginParams loginParams = BRTCLoginParams();
loginParams.userId = "999";
loginParams.roomId = "roomName";
loginParams.displayName = "UserName";
baiduRtcCloud.loginBRtcRoom(loginParams);
```

常见问题 注意事项：

1. flutter 中的 pubspec.yaml 插件版本要和gradle 版本匹配，flutter 版本匹配
2. Preferences的dart配置信息 需要设置enable true
3. Preferences 中 flutter sdk 的路径要设置正确
4. 遇到flutter 缓存问题，需要重启
5. gradle-wrapper.properties 版本问题要对应
6. pod 版本拉取失败，或者不对应，删除pod.lock 文件，删除缓存，重新拉取

RTC 服务端API

API概览

本文档用于介绍 BRTC 提供的服务端 RESTful API。

帮助您在 app 服务端也有简单快捷的使用体验。

房间管理相关接口 | 接口名称 | 功能描述 | | --- | --- | |RoomList | 查询当前活跃的房间列表 | |RoomUser | 查询房间中用户列表 | |RoomDisband | 解散房间 |

用户管理相关接口 | 接口名称 | 功能描述 | | --- | --- | |ClientUserBan | 用户禁言 | |ClientUserUnban | 用户发言解禁 | |ClientUserKickout | 指定用户踢出房间 |

云端录制相关接口 | 接口名称 | 功能描述 | | --- | --- | |CloudRecordingResourceidAcquire | 获取云端录制资源 | |CloudRecordingStart | 启动云端录制 | |CloudRecordingQuery | 查询云端录制 | |CloudRecordingStop | 停止云端录制 |

云端录制相关接口 | 接口名称 | 功能描述 | | --- | --- | |LiveStreamingByPassResourceidAcquire| 获取云端旁路转推资源 | |LiveStreamingByPassStart | 启动云端旁路转推 | |LiveStreamingByPassQuery | 查询云端旁路转推 | |LiveStreamingByPassStop | 停止云端旁路转推 |

使用说明

请求说明

服务地址 BRTC服务端 API 通过域名接入，服务域名为 **rtc.baidubce.com**。

通讯协议 BRTC 服务端 API 支持 HTTP 和 HTTPS 两种协议。为了提升数据的安全性，建议使用HTTPS协议。

数据格式 数据交换格式为JSON，所有request/response body内容均采用UTF-8编码。

API认证机制

API 认证机制的详细内容请参见鉴权认证。
如何获取 AK/SK 参见获取AK/SK。
使用常用工具快速生成签名请求，可以选择在线签名工具。
API 鉴权示例代码参见Sample-Code。
详细的签名报错及处理方式请参考常见签名认证错误排查。

公共请求头与响应头

公共请求头与响应头参数，如非必要，在每个接口单独的接口文档中不再进行说明，但每次请求均需要携带这些参数。

公共请求头

头域	说明	是否必须
host	服务域名，目前固定为rtc.baidubce.com	是
authorization	包含Access Key与请求签名。具体请参考鉴权认证	是
content-Type	application/json	是
x-bce-request-id	请求ID	否
x-bce-date	表示日期的字符串，符合API规范	否

公共响应头

头域	说明
content-type	application/json;charset=UTF-8
x-bce-request-id	对应请求ID

房间管理相关接口

查询当前活跃的房间列表

接口描述 用户根据appId查询房间列表。

请求（Request） 请求URI

```
GET /v1/room/list?appId={appId} HTTP/1.1
```

参数名	类型	是否必需	描述
appId	String	是	应用ID
orderType	String	否	房间排序方式，按照房间开始时间排序，支持取值 asc/desc，asc 表示升序，desc 表示降序，默认值为asc。

请求头域 除公共头域外，无其它特殊头域。

请求体 无 **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。

响应体 响应体是List类型，List中每个元素结构如下：

参数名	类型	描述
+appId	String	应用ID
+roomName	String	房间名
+roomId	Long	房间ID
+startTime	String	开始时间

示例 请求示例

```
 GET /v1/room/list?appId=testapp&orderType=desc  HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
[
    {
        "appId": "testapp",
        "roomName": "6543211234",
        "roomId": 883809059995648,
        "startTime": "2023-03-06 11:15:59"
    },
    {
        "appId": "testapp",
        "roomName": "6543211234342",
        "roomId": 887986989236224,
        "startTime": "2023-03-06 11:01:18"
    }
]
```

查询房间中用户列表

接口描述 用户根据appId和roomName查询当前房间中用户列表。

请求（Request） 请求URI

```
 GET /v1/room/users?appId={appId}&roomName={roomName} HTTP/1.1
```

参数名	类型	是否必需	描述
appId	String	是	应用ID
roomName	String	是	房间名称

请求头域 除公共头域外，无其它特殊头域。

请求体 无 **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。

响应体

参数名	类型	描述
appId	String	应用ID
roomName	String	房间名
roomId	Long	房间ID
startTime	Long	会话开始时间，为毫秒级别 Unix 时间戳
users	List	user 列表
+clientUserId	Long	客户端用户ID
+display	String	客户端用户名（publisher身份）
+isPublishing	Boolean	客户端用户是否正在推流

示例 请求示例

```
 GET /v1/room/users?appId=testapp&roomName=6543211234  HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "appId": "testapp",
    "roomName": "6543211234",
    "roomId": 883809059995648,
    "startTime": 1663818337808,
    "users": [
        {
            "clientUserId": 8891,
            "publishing": false
        },
        {
            "clientUserId": 69009,
            "display": "Z9zY3sfaTcph",
            "publishing": false
        },
        {
            "clientUserId": 42276,
            "display": "3WeTiZKThp5H",
            "publishing": true
        }
    ]
}
```

解散房间

接口描述 根据用户提供的appId和房间名，对相关房间进行解散的操作。

请求（Request） 请求URI

```
PUT /v1/room/disband HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名 |

响应（Response） 响应头域 除公共头域外，无其它特殊头域。

响应体 空

示例 请求示例

```
PUT /v1/room/disband HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "roomName": "testroom"
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

用户管理相关接口

用户禁言

接口描述 禁止用户发言，用户说话无法被其他用户听见。

- 需要注意该接口动作是临时动作，用户重新加入房间后，之前的禁言操作会失效。

请求（Request） 请求URI

```
PUT /v1/clientuser/ban HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名| | clientUserId | Long | 是 | 用户ID | | streams | List[String] | 否 | 具体需要禁的流
如果需要封禁视频，则添加video
如果需要封禁音频，则添加audio
如果音视频都封禁，则不需要设置stream参数 |

响应（Response） 响应头域 除公共头域外，无其它特殊头域。

响应体 无

示例 请求示例

```
 PUT /v1/clientuser/ban HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "roomName": "6543211234",
 "clientUserId": 123,
 "streams": [
       "video"
 ]
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

用户发言解禁

接口描述 对禁言的用户进行解禁。

请求（Request） 请求URI

```
 PUT /v1/clientuser/unban HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名| | clientUserId | Long | 是 | 用户ID | | streams | List[String] | 否 | 具体需要解禁的流
如果需要解禁视频，则添加video
如果需要解禁音频，则添加audio
如果音视频都解禁，则不需要设置stream参数 |

响应（Response） 响应头域 除公共头域外，无其它特殊头域。

响应体 无

示例 请求示例

```
PUT /v1/clientuser/unban HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "roomName": "6543211234",
 "clientUserId": 123,
 "streams": [
    "video"
 ]
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

踢出房间

接口描述 将指定用户踢出房间。

- 需要注意是服务器并不禁止被踢出的用户再次加入。

请求（Request） 请求URI

```
PUT /v1/clientuser/kickout HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求体 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名| | clientUserId | Long | 是 | 客户端用户ID |

响应（Response） 响应头域 除公共头域外，无其它特殊头域。

响应体 无

示例 请求示例

```
PUT /v1/clientuser/kickout HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "roomName": "6543211234",
 "clientUserId": 123
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

云端录制相关接口

获取云端录制资源

接口描述 在启动录制前，必须先调用获取云端录制资源接口，该接口会返回一个 resourceId，在启动录制时需要携带该 resourceId。

- resourceId 存在有效期，需要在有效期内调用启动录制接口。
- 一个 resourceId 仅可用于一次启动云端录制的请求。

请求（Request） 请求URI

```
GET /v1/cloudrecording/resourceid/acquire?appId={appId} HTTP/1.1
```

参数名	类型	是否必需	描述
appId	String	是	应用ID

请求头域 除公共头域外，无其它特殊头域。 **请求体** 无 **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。 **响应体**
| 字段名 | 类型 | 描述 | |----------|-------|----------| | resourceId | String | 录制资源ID | | maxValidTime | Long | 资源的最大有效时间戳，等于资源申请时间 + 5 分钟。
获得 resourceId 后，必须在最大有效时间内调用启动录制接口，超时后需要重新申请resourceId | **示例 请求示例**

```
GET /v1/cloudrecording/resourceid/acquire?appId=testapp HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "resourceId": "testResourceId",
    "maxValidTime": 1670383421767
}
```

启动云端录制

接口描述 获得录制资源ID后，可以在资源ID有效期内调用启动云端录制接口，对房间中通话进行录制。 调用该接口成功后，你可以从响应体中获得一个录制任务ID，该录制任务ID是一次录制任务的唯一标识。

请求（Request） 请求URI

```
POST /v1/cloudrecording/start HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。 **请求体** | 参数名 | 类型 | 是否必需 | 描述 | |----------|----------|-----|----------|| | resourceId | String | 是 | 录制资源ID | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名 | | clientUserId | Long | 否 | 单路录制时需要，表示要录制的客户端 uid | | taskExpiredHour | Integer | 否 | 录制任务接口可以调用的时效性，从成功启动录制任务开始计算，超时后无法调用查询和停止等接口，但是录制任务不会停止。参数的单位是小时，默认24小时，最大可设置72小时（3天），最小设置1小时。 | | recordingCfg | CloudRecordingConfig | 是 | 录制基础配置 | | +recordingMode | String | 是 | 录制模式
0表示单路录制，1表示混流录制，默认值为0。
单路录制：将指定clientUserId的音视频流直接录制到一个音视频文件，不进行转码。
混流录制：将房间内多个用户的音视频混流转码后录成一个音视频文件，目前一个混流录制中最多添加37路用户流。 | | +streamMode | String | 是 | 录制流模式
0表示音频+视频，1表示纯音频，2表示纯视频，默认值为0。 | | +maxIdleSeconds | Integer | 否 | 最大续录时长。当超过最大续录时长，没有用户流参与录制，自动停止录制，单位：秒。
默认值为 30 秒，该值需大于等于 5秒、且小于等于 7200秒(2小时)。
注意：混流模式下，续录等待期内会补黑帧和静音帧继续录制，单路模式下则不会录制。在续录等待期内，单路和混流录制会按照音频时长收取录制费用。 | | +targetOrExcludeStreamsCfg | TargetOrExcludeStreamsConfig | 否 | 参与任务的音频流和视频流的黑白名单，仅在 recordingMode="1" 时有效，详细结构见下文。 | | mixTranscodingCfg | CloudMixTranscodingCfg | 否 | 录制混流转码配置，用于配置输出音视频编码、布局和水印等。 | | +encodeCfg | EncodeConfig | 否 | 编码配置，详细结构见下文。 | | +layoutCfg | LayoutConfig | 否 | 布局配置，详细结构见下文。 | | +watermarkCfgs | WatermarkConfig数组 | 否 | 全局水印配置，最多支持10个全局水印，详细结构见下文。 | | recordingFileCfg | RecordingFileCfg | 是 | 录制文件配置 | | +type | String | 是 | 录制文件格式
在 streamMode="0" 或 streamMode="2" 时，如果 recordingMode="1"，type 支持取值 flv、mp4，但如果 recordingMode="0"，type 只支持取值 flv。
在streamMode="1" 时，type 支持取值aac、mp3，并且此时 type 取值需要和 audioCodec 取值相同。 | | +maxDuration | Integer | 否 | 单文件录制时长，单位分钟，达到该阈值时将自动切割文件，默认值为60。 | | storageCfg | StorageCfg | 是 | 文件存储平台配置 | | +vendor | String | 是 | 存储平台名，目前只支持取值0，表示百度云BOS。 | | +region | String | 是 | 区域信息，取值0表示bj区域，取值1表示gz区域，取值2表示su区域。 | | +bucket | String | 是 | 存储 bucket 取值 | | +filenameFormat | String | 是 | 录制文件命名模式，公有云则对应 bos 平台上文件的路径和文件名。
recordingMode="0" 时，默认值为 "%a/%r/%d/%T/recording%t%u.%f"，并且要求取值必须含有%T、%t，必须以.%f结尾。
recordingMode="1" 时，默认值为 "%a/%r/%d/%T/recording_%t.%f"，并且要求取值必须含有%T、%t，必须以.%f结尾。
其中%a表示appId，%r表示roomName，%d表示日期，%T表示taskId，%t表示录制文件开始时间，%f表示文件类型。 |

TargetOrExcludeStreamsConfig 结构

对混流类型任务可以设置音频和视频的黑白名单，黑白名单的内容可以是具体的 userId，也可以是特定的匹配规则。音频白名单和音频黑名单不能同时设置，视频亦然。支持通过设置 ".*$" 通配符规则，来前缀匹配用户的 userId；支持通过设置 "#allstream#" 规则来匹配所有用户。

参数名	类型	是否必需	描述
targetAudioUserIds	String数组	否	音频流白名单，指定哪些用户的音频流参与该任务，例如["1", "2", "3"], 代表 userId 1，2，3的音频流会参与任务；["1.*$"], 代表 userId 前缀为1的用户音频流参与任务；["#allstream#"] 代表所有用户的音频流都会参与任务。默认不填房间内所有的音频流都会参与任务，但实际参与任务的用户数不超过37。
excludeAudioUserIds	String数组	否	音频流黑名单，指定哪些用户的音频流不参与该任务，例如["1", "2", "3"], 代表 userId 1，2，3的音频流不参与任务；["1.*$"], 代表 userId 前缀为1的用户音频不参与任务；["#allstream#"] 代表所有用户的音频流都不会参与任务。默认不填房间内所有的音频流都会参与任务，但实际参与任务的用户数不超过37。
targetVideoUserIds	String数组	否	视频流白名单，指定哪些用户的视频流参与该任务，具体规则同音频白名单。默认不填房间内所有的视频流都会参与任务，但实际参与任务的用户数不超过37。
excludeVideoUserIds	String数组	否	视频流黑名单，指定哪些用户的视频流不参与该任务，具体规则同音频黑名单。默认不填房间内所有的视频流都会参与任务，但实际参与任务的用户数不超过37。

黑白名单示例

预期效果	推荐设置
所有用户音频流和视频流都需要参与任务	无需设置黑白名单相关参数
所有用户音频流都需要参与，只有 123 用户和 4 开头用户的视频流需要参与	targetVideoUserIds: ["123", "4.*$"]
所有用户音频流都需要参与，只有 123 用户和 4 开头用户的视频流需要排除	excludeVideoUserIds: ["123", "4.*$"]
用户 123 的音频流需要排除，只有用户 456 的视频流需要参与	excludeAudioUserIds: ["123"] targetVideoUserIds: ["456"]
所有用户音频流都需要参与，所有用户视频流都需要排除	excludeVideoUserIds: ["#allstream#"] 如果不需要使用水印等视频模式特有的功能，将 streamMode 设置为 "1" 也可达到同样的效果。

EncodeConfig结构

参数名	类型	是否必需	描述
audioCodec	String	否	音频编码，默认值为 aac。 该配置项只有在 streamMode="1" 时生效，支持取值 aac、mp3。其它模式均为默认值 aac。
width	Integer	否	混流输出画布的宽，默认值为1280，取值范围[0,1920]，单位为像素值。 需要在 streamMode 包含视频时才有效。
height	Integer	否	混流输出画布的高，默认值为720，取值范围[0,1920]，单位为像素值。 需要在 streamMode 包含视频时才有效。
videoBitrate	Integer	否	视频码率，单位 kb/s，默认值1200。 需要在 streamMode 包含视频时才有效。

LayoutConfig结构

- streamMode 中包含视频时 layoutCfg 才有效。

参数名	类型	是否必需	描述
layout	String	是	布局模板，默认值为 side_by_side_primary。 可取值 side_by_side_primary（主次平铺），side_by_side_equal（相等平铺），picture_in_picture_bottom（画中画），custom（自定义布局）。
mainUserId	Long	否	混流任务主画面窗口对应的 userId。只在 side_by_side_primary 和 picture_in_picture_bottom 布局生效。 如果未设置，会自动按照先后顺序选择主画面。
renderMode	String	否	用户窗口绘制模式，在主次平铺、相等平铺和画中画布局中有效，可取值0和1，0表示缩放后裁剪，1表示缩放并显示黑底，默认值为1。
defaultPlaceholderImageUrl	String	否	窗口默认占位图 URL 地址。 - 当用户视频流未发布、暂停采集时，窗口会使用占位图来填充； - 支持 jpg 和 png 格式； - 当占位图和窗口尺寸不一致时，会依据 renderMode 配置在窗口中绘制占位图。
generalWindowCfgs	GeneralWindowConfig 数组	否	通用布局配置，在任意布局模板下都生效，详细结构见下文。可用于对每个小窗口设置属性，比如流级别水印。
customWindowCfgs	CustomWindowConfig 数组	否	窗口自定义布局设置，在 layout=custom 时生效，详细结构见下文。

GeneralWindowConfig结构 | 参数名 | 类型 | 是否必需| 描述 | | --- | --- | --- | ---| | userId | Long | 是 | 窗口对应的客户端 userId | | watermarkCfgs | WatermarkConfig数组 | 否 | 流级别水印设置，置于用户的小窗口内。每个用户最多支持10个流级别水印。详细结构见下文。
流级别水印建议使用相对坐标。另外出于效果考虑，不建议使用流级别图片类型水印。 |

CustomWindowConfig 结构

- **如果某个混流源的 userId 在 customWindowCfgs 结构中不存在，那么会将该用户窗口平铺到整个画布上**。

参数名	类型	是否必需	描述
userId	Long	是	窗口对应的客户端 userId
renderMode	String	否	窗口绘制模式，可取值0和1，0表示缩放后裁剪，1表示缩放并显示黑底，默认值为1。
xpos	int	否	该窗口在输出时矩形左上角的X坐标，单位为像素值，默认值为0。
ypos	int	否	该窗口在输出时矩形左上角的Y坐标，单位为像素值，默认值为0。
zorder	int	否	窗口在输出时的层级，不填默认为0。zorder越大，处于画面越上方，取值为0时窗口位于最底层。
width	int	否	窗口在输出时的宽度，单位为像素值，默认值为0。
height	int	否	窗口在输出时的高度，单位为像素值，默认值为0。
placeholderImageUrl	String	否	窗口占位图 URL 地址。 - 自定义布局中，可以对不同窗口设置不同的占位图，会覆盖窗口默认占位图设置。 - 支持 jpg 和 png 格式； - 当占位图和窗口尺寸不一致时，会依据自定义布局的 renderMode 配置在窗口中绘制占位图。

WatermarkConfig结构

- recordingMode="0"时，不会进行转码，全局水印和流级别水印均无效。

参数名	类型	是否必需	描述
type	String	是	水印类型，可取值image，text，clock。image 表示图片水印，text 表示文字水印，clock 表示时间戳水印。
posMode	Integer	否	水印坐标类型，支持取值0或1，默认值为0。 当 posMode=0 时，xpos、ypos 为绝对值坐标，单位是像素值； 当 posMode=1时，xpos、ypos 表示坐标占全局窗口/用户窗口width、height的百分比，取值范围[0,1]，最多精确到小数点后3位。 **流级别水印建议使用 posMode=1**。
xpos	Float	是	该水印在输出时矩形左上角的X坐标，默认值为0。详见 posMode 描述。
ypos	Float	否	该水印在输出时矩形左上角的Y坐标，默认值为0。详见 posMode 描述。
imageUrl	String	否	type=image 时必须携带，图片水印 http url，水印图片只支持缩放。
width	Integer	否	type=image 时生效，该水印在输出时的宽度，单位为像素值，不填默认为图片宽度。
height	Integer	否	type=image 时生效，该水印图片在输出时的高度，单位为像素值，不填默认为图片高度。
color	String	否	水印颜色，type=clock 或 type=text 时生效，采用big-endian ARGB的方式，并使用16进制表示。背景颜色取值共10位，前2位固定为0x，接下来2位表示透明度，后6位表示颜色值，例如0xFF000000。
size	Integer	否	字体大小，type=clock 或 type=text 时生效。
font	string	否	字体类型，type=clock 或 type=text 时生效，暂时只支持取值Normal。
text	String	否	文字水印内容，type=text 时表示文字水印内容，type=clock 时在时间戳前面添加该文字。
textTransfer	Boolean	否	表示 text 字段是否进行转义，默认值为false。 如果 textTransfer=true，那么 text 中 %D 会被转义成客户端用户的 display （昵称）属性。 例如 text="用户_%D"，display="Bob"，textTransfer=true，text 内容会被转义成 "用户_Bob"。
timeFormat	String	否	时间戳格式，默认 %Y-%m-%d %H:%M:%S，时间戳格式参考：https://man7.org/linux/man-pages/man3/strftime.3.html

响应（Response） **响应头域** 除公共头域外，无其它特殊头域。 **响应体** | 字段名 | 类型 | 描述 | | --- | --- | --- | | taskId | String | 录制任务ID | **示例** **请求示例** **单路录制示例**

```
 POST /v1/cloudrecording/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "resourceId": "testResourceId",
 "appId": "testapp",
 "roomName": "65432112341",
 "clientUserId": 68885,
 "recordingCfg": {
  "maxIdleSeconds": 60,
  "recordingMode": "0",
  "streamMode": "0"
 },
 "mixTranscodingCfg": {
  "encodeCfg": {
   "audioCodec": "aac"
  }
 },
 "recordingFileCfg": {
  "type": "flv"
 },
 "storageCfg": {
  "vendor": "0",
  "region": "0",
  "bucket": "test-record"
 }
}
```

混流录制示例

```
 POST /v1/cloudrecording/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "resourceId": "testResourceId",
 "appId": "testapp",
 "roomName": "65432112341",
 "recordingCfg": {
  "maxIdleSeconds": 60,
  "recordingMode": "1",
  "streamMode": "0"
 },
 "mixTranscodingCfg": {
  "encodeCfg": {
   "audioCodec": "aac"
  }
 },
 "recordingFileCfg": {
  "type": "flv"
 },
 "storageCfg": {
  "vendor": "0",
  "region": "0",
  "bucket": "test-record"
 }
}
```

自定义布局混流录制示例

```
 POST /v1/cloudrecording/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "resourceId":"testResourceId",
    "appId":"testapp",
    "roomName":"65432112341",
    "recordingCfg":{
        "maxIdleSeconds":60,
        "recordingMode":"1",
        "streamMode":"0"
    },
    "mixTranscodingCfg":{
        "encodeCfg":{
            "audioCodec":"aac"
        },
        "layoutCfg":{
            "layout":"custom",
            "customWindowCfgs":[
                {
                    "userId":123,
                    "renderMode":"0",
                    "xpos":0,
                    "ypos":0,
                    "zorder":1,
                    "width":300,
                    "height":300
                },
                {
                    "userId":456,
                    "renderMode":"1",
                    "xpos":300,
                    "ypos":300,
                    "zorder":1,
                    "width":300,
                    "height":300
                }
            ]
        }
    },
    "recordingFileCfg":{
        "type":"flv"
    },
    "storageCfg":{
        "vendor":"0",
        "region":"0",
        "bucket":"test-record"
    }
}
```

包含全局水印和流级别水印配置示例

```
 POST /v1/cloudrecording/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

```
{
    "resourceId":"testResourceId",
    "appId":"testapp",
    "roomName":"65432112341",
    "taskExpiredHour":24,
    "recordingCfg":{
        "recordingMode":"1",
        "streamMode":"0",
        "maxIdleSeconds":60
    },
    "mixTranscodingCfg":{
        "encodeCfg":{
            "audioCodec":"aac",
            "width":1280,
            "height":720,
            "videoBitrate":1200
        },
        "layoutCfg":{
            "layout":"side_by_side_equal",
            "mainUserId": 123123,
            "generalWindowCfgs": [
                {
                    "userId": 123456,
                    "watermarkCfgs": [
                        {
                            "type":"text",
                            "posMode": 1,
                            "text":"用户_%D",
                            "textTransfer": true,
                            "xpos":0.1,
                            "ypos":0.1,
                            "font":"Normal",
                            "size":22,
                            "color":"0xFFFFFFFF"
                        },
                        {
                            "type":"clock",
                            "posMode": 1,
                            "xpos":0.1,
                            "ypos":0.5,
                            "timeFormat":"%Y-%m-%d %H:%M:%S",
                            "text":"北京时间：",
                            "font":"Normal",
                            "size":22,
                            "color":"0xFFFFFFFF"
                        }
                    ]
                }
            ]
        },
        "watermarkCfgs":[
            {
                "type":"image",
                "imageUrl":"https://brtc-tools.cdn.bcebos.com/shuiyin.jpeg",
                "xpos":0,
                "ypos":0,
                "width":200,
                "height":200
            },
            {
                "type":"text",
                "text":"Hello world!",
```

```
            "xpos":0,
            "ypos":100,
            "font":"Normal",
            "size":22,
            "color":"0xFF000000"
        },
        {
            "type":"clock",
            "timeFormat":"%Y-%m-%d %H:%M:%S",
            "text":"北京时间：",
            "xpos":0,
            "ypos":200,
            "font":"Normal",
            "size":22,
            "color":"0xFF000000"
        }
    ]
  },
  "recordingFileCfg":{
    "type":"flv",
    "maxDuration":60
  },
  "storageCfg":{
    "vendor":"0",
    "region":"0",
    "bucket":"test-record",
    "filenameFormat":"%a/%r/%d/%T/recording_%t.%f"
  }
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
   "taskId": "testTaskId"
}
```

查询云端录制任务

接口描述 录制任务有效期内，可以通过调用查询录制任务接口来查询录制状态，和生成的录制文件列表。

请求（Request） 请求URI

```
GET /v1/cloudrecording/{appId}/{taskId}/query HTTP/1.1
```

参数名	类型	是否必需	描述
appId	String	是	应用ID
taskId	String	是	录制任务ID

请求头域 除公共头域外，无其它特殊头域。 **请求体** 无 **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。 **响应体**

| 字段名 | 类型 | 描述 | | --- | --- | --- | | appId | String |应用ID | | taskId | String | 录制任务ID | |status | String | 任务状态。

STARTED：录制已开启，但是可能用户没有推流所以未进行；

IN_PROGRESS：录制任务进行中；

STOPPED：录制已停止。| | createdTime | Long | 任务创建时间，为毫秒级别 Unix 时间戳 | | stopTime | Long | 任务停止时间，为毫秒级别 Unix 时间戳 | | expiredTime | Long | 任务过期时间，为毫秒级别 Unix 时间戳 | | fileList | List | 录制任务生成的文件列表。

需要注意的是，**此接口查询录制文件列表并不是实时的，可能有1分钟左右的延时**。

此外，**此接口最多查询 1000 个文件**，需要注意文件切割时间和录制任务持续时间，避免生成录制文件数目超过1000。 | | +startTs | Long | 录制文件开始时间，为毫秒级别 Unix 时间戳 |

| +endTs | Long | 录制文件结束时间，为毫秒级别 Unix 时间戳 | | +fileUrl | String | 文件下载 url。

需要注意的是，**文件下载 url，目前仅适配百度智能云对象存储 BOS，并且不包含 url 鉴权等信息**。 |

示例 请求示例

```
 GET /v1/cloudrecording/testapp/testTaskId/query HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "appId": "testapp",
    "taskId": "testTaskId",
    "status": "STOPPED",
    "createdTime": 1717401011497,
    "stopTime": 1717401169080,
    "expiredTime": 1717404611497,
    "fileList": [
        {
            "startTs": 1717401015000,
            "endTs": 1717401077000,
            "fileUrl": "https://***/***/***"
        },
        {
            "startTs": 1717401076000,
            "endTs": 1717401134000,
            "fileUrl": "https://***/***/***"
        }
    ]
}
```

停止云端录制任务

接口描述 成功启用云端录制任务后，可以使用此接口来停止录制任务。

请求（Request） 请求URI

```
 POST /v1/cloudrecording/stop HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。 **请求体** | 参数名 | 类型 | 是否必需 | 描述 | |------|-------|-----|-----| | appId | String | 是 | 应用ID | | taskId | String | 是 | 录制任务ID | **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。 **响应体** 无 **示例 请求示例**

```
POST /v1/cloudrecording/stop HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "taskId": "testTaskId"
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

云端旁路转推相关接口

获取云端旁路转推资源

接口描述

本接口用于获取云端旁路转推资源。

旁路转推，是指将实时音视频流以 rtmp 等协议转推到直播等服务，旁路转推也可简称**转推**。

在启动转推前，必须先调用获取云端旁路转推资源接口，该接口会返回一个 resourceId，在启动转推时需要携带该 resourceId。

- resourceId 存在有效期（5分钟），需要在有效期内调用启动转推接口。
- 一个 resourceId 仅可用于一次启动转推的请求。

请求结构

```
GET /v1/livestreamingBypass/resourceid/acquire?appId={appId} HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求参数

参数名	类型	是否必需	描述
appId	String	是	应用ID

响应头域 除公共头域外，无其它特殊头域。

响应参数 | 字段名 | 类型 | 描述 | | --- | --- | --- | | resourceId | String | 资源ID | | maxValidTime | Long | UNIX时间戳，单位毫秒，表示资源的最大有效时间戳，等于申请时间 + 5 分钟。
获得 resourceId 后，必须在最大有效时间内调用 start 接口开始转推，超时后需要重新请求一个resourceId。 |

请求示例

```
GET /v1/livestreamingBypass/resourceid/acquire?appId=testapp HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "resourceId": "testResourceId",
    "maxValidTime": 1682391803241
}
```

启动云端旁路转推

接口描述 本接口用于启动云端旁路转推任务。

获取转推资源ID后，可以在资源ID有效期内调用启动云端旁路转推接口，对房间中用户进行直播转推。 调用该接口成功后，可以从响应体中获得一个**转推任务ID**，该转推任务ID是一次转推任务的唯一标识。

请求结构

```
POST /v1/livestreamingBypass/start HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求参数 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | resourceId | String | 是 | 转推资源ID | | appId | String | 是 | 应用ID | | roomName | String | 是 | 房间名 | | clientUserId | Long | 否 | 单路转推时需要，表示要转推的客户端uid | | pushUrl | String | 是 | 直播推流地址，只支持RTMP协议 | | taskExpiredHour | Integer | 否 | 转推任务接口可以调用的时效性，单位小时。
从成功启动转推任务开始计算，超时后无法调用查询、更新和停止等接口，但是转推任务不会停止。
参数的单位是小时，默认24小时，最大可设置72小时（3天），最小设置1小时。 | | bypassCfg | LiveStreamingBypassConfig | 否 | 转推基础配置 | | +bypassMode | String | 否 | 转推模式。
0表示单流转推，1表示混流转推，默认值为0。
单路转推：将指定的clientUserId用户的音视频流直接转推到直播服务，不会进行转码。
混流转推：将房间内多个用户的音视频混流转码后转推到转推服务，目前一个混流转推中最多添加6路用户流。 | | +streamMode | String | 否 | 转推流模式。
0表示音频+视频，1表示纯音频，2表示纯视频，默认值为0。 | | +maxIdleSeconds | Integer | 否 | 最大续播时长，单位：秒。
默认值为 30 秒，该值需大于等于 5 秒、且小于等于 7200秒(2小时)。
当超过最大续播时长，没有用户流参与转推，自动停止转推。
注意：混流模式下，续播等待期内会补黑帧和静音帧继续转推，单路模式下则不会转推。在续播等待期内，单路和混流转推会按照音频时长收取转推/混流费用。 | | +targetOrExcludeStreamsCfg | TargetOrExcludeStreamsConfig | 否 | 参与任务的音频流和视频流的黑白名单，仅在 bypassMode="1" 时有效，详细结构见下文。 | | mixTranscodingCfg | CloudMixTranscodingCfg | 否 | 混流转码配置，用于配置输出音视频编码、布局和水印等。 | | +encodeCfg | EncodeConfig | 否 | 编码配置，详细结构见下文。 | | +layoutCfg | LayoutConfig | 否 | 布局配置，详细结构见下文。 | | +watermarkCfgs | WatermarkConfig数组 | 否 | 全局水印配置，最多支持10个全局水印，详细结构见下文。 |

TargetOrExcludeStreamsConfig 结构

对混流类型任务可以设置音频和视频的黑白名单，黑白名单的内容可以是具体的 userId，也可以是特定的匹配规则。音频白名

单和音频黑名单不能同时设置，视频亦然。支持通过设置 ".*$" 通配符规则，来前缀匹配用户的 userId；支持通过设置 "#allstream#" 规则来匹配所有用户。

参数名	类型	是否必需	描述
targetAudioUserIds	String 数组	否	音频流白名单，指定哪些用户的音频流参与该任务，例如["1", "2", "3"], 代表 userId 1，2，3的音频流会参与任务；["1.*$"], 代表 userId 前缀为1的用户音频流参与任务；["#allstream#"] 代表所有用户的音频流都会参与任务。默认不填房间内所有的音频流都会参与任务，但实际参与任务的用户数不超过37。
excludeAudioUserIds	String 数组	否	音频流黑名单，指定哪些用户的音频流不参与该任务，例如["1", "2", "3"], 代表 userId 1，2，3的音频流不参与任务；["1.*$"], 代表 userId 前缀为1的用户音频不参与任务；["#allstream#"] 代表所有用户的音频流都不会参与任务。默认不填房间内所有的音频流都会参与任务，但实际参与任务的用户数不超过37。
targetVideoUserIds	String 数组	否	视频流白名单，指定哪些用户的视频流参与该任务，具体规则同音频白名单。默认不填房间内所有的视频流都会参与任务，但实际参与任务的用户数不超过37。
excludeVideoUserIds	String 数组	否	视频流黑名单，指定哪些用户的视频流不参与该任务，具体规则同音频黑名单。默认不填房间内所有的视频流都会参与任务，但实际参与任务的用户数不超过37。

黑白名单示例

预期效果	推荐设置
所有用户音频流和视频流都需要参与任务	无需设置黑白名单相关参数
所有用户音频流都需要参与，只有 123 用户和 4 开头用户的视频流需要参与	targetVideoUserIds: ["123", "4.*$"]
所有用户音频流都需要参与，只有 123 用户和 4 开头用户的视频流需要排除	excludeVideoUserIds: ["123", "4.*$"]
用户 123 的音频流需要排除，只有用户 456 的视频流需要参与	excludeAudioUserIds: ["123"] targetVideoUserIds: ["456"]
所有用户音频流都需要参与，所有用户视频流都需要排除	excludeVideoUserIds: ["#allstream#"] 如果不需要使用水印等视频模式特有的功能，将 streamMode 设置为 "1" 也可达到同样的效果。

EncodeConfig 结构 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | width | Integer | 否 | 混流输出画布的宽，默认值为1280，取值范围[0,1920]，单位为像素值。
需要在streamMode包含视频时才有效。 | | height | Integer | 否 | 混流输出画布的高，默认值为720，取值范围[0,1920]，单位为像素值。
需要在streamMode包含视频时才有效。 | | videoBitrate | Integer | 否 | 视频码率，单位 kb/s，默认值1200。
需要在streamMode包含视频时才有效。 |

LayoutConfig结构

- streamMode包含视频时，layoutCfg才有效。

参数名	类型	是否必需	描述
layout	String	是	布局模板，默认值为 side_by_side_primary。 可取值 side_by_side_primary（主次平铺），side_by_side_equal（相等平铺），picture_in_picture_bottom（画中画），custom（自定义布局）。
mainUserId	Long	否	混流后主画面窗口对应的userId。只在 side_by_side_primary 和 picture_in_picture_bottom 布局生效。 如果未设置，会自动按照先后顺序选择主画面。
renderMode	String	否	用户窗口绘制模式，在主次平铺、相等平铺和画中画布局中有效，可取值0和1，0表示缩放后裁剪，1表示缩放并显示黑底，默认值为1。
defaultPlaceholderImageUrl	String	否	窗口默认占位图 URL 地址。 - 当用户视频流未发布、暂停采集时，窗口会使用占位图来填充； - 支持 jpg 和 png 格式； - 当占位图和窗口尺寸不一致时，会依据 `renderMode` 配置在窗口中绘制占位图。
generalWindowCfgs	GeneralWindowConfig 数组	否	通用布局配置，在任意布局模板下都生效，详细结构见下文。可用于对每个小窗口设置属性，比如流级别水印。
customWindowCfgs	CustomWindowConfig 数组	否	窗口自定义布局设置，在 layout=custom 时生效，详细结构见下文。

GeneralWindowConfig结构 | 参数名 | 类型 | 是否必需| 描述 | | --- | --- | --- | ---| | userId | Long | 是 | 窗口对应的客户端 userId | | watermarkCfgs | WatermarkConfig数组 | 否 | 流级别水印设置，置于用户的小窗口内。每个用户最多支持10个流级别水印。详细结构见下文。
流级别水印建议使用相对坐标。另外出于效果考虑，不建议使用流级别图片类型水印。 |

CustomWindowConfig 结构

- **如果某个混流源的 userId 在 customWindowCfgs 结构中不存在，那么会将该用户窗口平铺到整个画布上**。

参数名	类型	是否必需	描述
userId	Long	是	窗口对应的客户端 userId
renderMode	String	否	窗口绘制模式，可取值0和1，0表示缩放后裁剪，1表示缩放并显示黑底，默认值为1。
xpos	int	否	该窗口在输出时矩形左上角的X坐标，单位为像素值，默认值为0。
ypos	int	否	该窗口在输出时矩形左上角的Y坐标，单位为像素值，默认值为0。
zorder	int	否	窗口在输出时的层级，不填默认为0。zorder越大，处于画面越上方，取值为0时窗口位于最底层。
width	int	否	窗口在输出时的宽度，单位为像素值，默认值为0。
height	int	否	窗口在输出时的高度，单位为像素值，默认值为0。
placeholderImageUrl	String	否	窗口占位图 URL 地址。 - 自定义布局中，可以对不同窗口设置不同的占位图，会覆盖窗口默认占位图设置。 - 支持 jpg 和 png 格式； - 当占位图和窗口尺寸不一致时，会依据自定义布局的 `renderMode` 配置在窗口中绘制占位图。

WatermarkConfig结构

- bypassMode="0"时，不会进行转码，全局水印和流级别水印均无效。

参数名	类型	是否必需	描述
type	String	是	水印类型，可取值image，text，clock。image 表示图片水印，text 表示文字水印，clock 表示时间戳水印。
posMode	Integer	否	水印坐标类型，支持取值0或1，默认值为0。 当 posMode=0 时，xpos、ypos 为绝对值坐标，单位是像素值； 当 posMode=1时，xpos、ypos 表示坐标占全局窗口/用户窗口width、height的百分比，取值范围[0,1]，最多精确到小数点后3位。 **流级别水印建议使用 posMode=1**。
xpos	Float	是	该水印在输出时矩形左上角的X坐标，默认值为0。详见 posMode 描述。
ypos	Float	否	该水印在输出时矩形左上角的Y坐标，默认值为0。详见 posMode 描述。
imageUrl	String	否	type=image 时必须携带，图片水印 http url，水印图片只支持缩放。
width	Integer	否	type=image 时生效，该水印在输出时的宽度，单位为像素值，不填默认为图片宽度。
height	Integer	否	type=image 时生效，该水印图片在输出时的高度，单位为像素值，不填默认为图片高度。
color	String	否	水印颜色，type=clock 或 type=text 时生效，采用big-endian ARGB的方式，并使用16进制表示。背景颜色取值共10位，前2位固定为0x，接下来2位表示透明度，后6位表示颜色值，例如0xFF000000。
size	Integer	否	字体大小，type=clock 或 type=text 时生效。
font	string	否	字体类型，type=clock 或 type=text 时生效，暂时只支持取值Normal。
text	String	否	文字水印内容，type=text 时表示文字水印内容，type=clock 时在时间戳前面添加该文字。
textTransfer	Boolean	否	表示 text 字段是否进行转义，默认值为false。 如果 textTransfer=true，那么 text 中 %D 会被转义成客户端用户的 display （昵称）属性。 例如 text="用户_%D"，display="Bob"，textTransfer=true，text 内容会被转义成 "用户_Bob"。
timeFormat	String	否	时间戳格式，默认 %Y-%m-%d %H:%M:%S，时间戳格式参考：https://man7.org/linux/man-pages/man3/strftime.3.html

响应头域 除公共头域外，无其它特殊头域。 **响应参数** | 字段名 | 类型 | 描述 | | --- | --- | --- | | taskId | String | 转推任务ID |

请求示例 单路转推示例

```
 POST /v1/livestreamingBypass/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "resourceId":"testResourceId",
    "appId":"testApp",
    "roomName":"65432112341",
    "clientUserId": 123456,
    "pushUrl": "rtmp://***/***/***",
    "taskExpiredHour":24,
    "bypassCfg":{
        "bypassMode":"0",
        "streamMode":"0",
        "maxIdleSeconds":60
    }
}
```

混流转推示例

```
 POST /v1/livestreamingBypass/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "resourceId":"testResourceId",
    "appId":"testApp",
    "roomName":"65432112341",
    "pushUrl": "rtmp://***/***/***",
    "taskExpiredHour":24,
    "bypassCfg":{
        "bypassMode":"1",
        "streamMode":"0",
        "maxIdleSeconds":60
    },
    "mixTranscodingCfg":{
     "layoutCfg": {
      "layout": "side_by_side_primary",
      "mainUserId": 123456
     }
    }
}
```

自定义布局混流转推示例

```
 POST /v1/livestreamingBypass/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "resourceId":"testResourceId",
    "appId":"testApp",
    "roomName":"65432112341",
    "pushUrl": "rtmp://***/***/***",
    "taskExpiredHour":24,
    "bypassCfg":{
        "bypassMode":"1",
        "streamMode":"0",
        "maxIdleSeconds":60
    },
    "mixTranscodingCfg":{
      "layoutCfg":{
            "layout":"custom",
            "customWindowCfgs":[
                {
                    "userId":"123",
                    "renderMode":"0",
                    "xpos":300,
                    "ypos":0,
                    "zorder":1,
                    "width":300,
                    "height":300
                },
                {
                    "userId":"456",
                    "renderMode":"1",
                    "xpos":300,
                    "ypos":400,
                    "zorder":1,
                    "width":300,
                    "height":300
                }
            ]
        }
    }
}
```

包含全局水印和流级别水印配置示例

```
 POST /v1/livestreamingBypass/start HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "resourceId":"testResourceId",
    "appId":"testApp",
    "roomName":"65432112341",
    "pushUrl": "rtmp://***/***/***",
    "taskExpiredHour":24,
    "bypassCfg":{
        "bypassMode":"1",
        "streamMode":"0",
        "maxIdleSeconds":60
    },
```

```
  "mixTranscodingCfg":{
   "encodeCfg":{
        "width":1280,
        "height":720,
        "videoBitrate":1200
     },
     "layoutCfg":{
        "layout":"side_by_side_primary",
        "mainUserId": 123123,
        "generalWindowCfgs": [
           {
              "userId": 123456,
              "watermarkCfgs": [
                 {
                    "type":"text",
                    "posMode": 1,
                    "text":"用户_%D",
                    "textTransfer": true,
                    "xpos":0.1,
                    "ypos":0.1,
                    "font":"Normal",
                    "size":22,
                    "color":"0xFFFFFFFF"
                 },
                 {
                    "type":"clock",
                    "posMode": 1,
                    "xpos":0.1,
                    "ypos":0.5,
                    "timeFormat":"%Y-%m-%d %H:%M:%S",
                    "text":"北京时间：",
                    "font":"Normal",
                    "size":22,
                    "color":"0xFFFFFFFF"
                 }
              ]
           }
        ]
     },
     "watermarkCfgs":[
        {
           "type":"image",
           "imageUrl":"https://brtc-tools.cdn.bcebos.com/shuiyin.jpeg",
           "xpos":0,
           "ypos":0,
           "width":200,
           "height":200
        },
        {
           "type":"text",
           "text":"Hello world!",
           "xpos":0,
           "ypos":100,
           "font":"Normal",
           "size":22,
           "color":"0xFF000000"
        },
        {
           "type":"clock",
           "timeFormat":"%Y-%m-%d %H:%M:%S",
           "text":"北京时间：",
           "xpos":0,
```

```
            "ypos":200,
            "font":"Normal",
            "size":22,
            "color":"0xFF000000"
        }
    ]
  }
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
   "taskId": "testTaskId"
}
```

停止云端旁路转推

接口描述 本接口用于停止已启动的旁路转推任务。

请求结构

```
 POST /v1/livestreamingBypass/stop HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求参数

参数名	类型	是否必需	描述
appId	String	是	应用ID
taskId	String	是	转推任务ID

响应头域 除公共头域外，无其它特殊头域。

响应参数 无

请求示例

```
 POST /v1/livestreamingBypass/stop HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
 "appId": "testapp",
 "taskId": "testTaskId"
}
```

响应示例

```
HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

查询云端旁路转推

接口描述 本接口用于查询已启动的旁路转推任务状态。

请求结构

```
GET /v1/livestreamingBypass/{appId}/{taskId}/query HTTP/1.1
```

请求头域 除公共头域外，无其它特殊头域。

请求参数 | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用ID | | taskId | String | 是 | 转推任务ID |

响应头域 除公共头域外，无其它特殊头域。

响应参数

字段名	类型	描述
appId	String	应用ID
taskId	String	转推任务ID
status	String	任务状态。 STARTED：转推已开启，但是可能用户没有推流所以未进行； IN_PROGRESS：任务进行中； STOPPED：转推已停止。
createdTime	Long	任务创建时间，为毫秒级别 Unix 时间戳
expiredTime	Long	任务过期时间，为毫秒级别 Unix 时间戳

请求示例

```
GET /v1/livestreamingBypass/testapp/testTaskId/query HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "appId": "testapp",
    "taskId": "testTaskId",
    "status": "STOPPED",
    "createdTime": 1670830712433,
    "expiredTime": 1670917112433
}
```

云播放器相关接口

创建云播放器

接口描述

- 云播放器功能现阶段是内测功能，如需开通请联系技术人员。

通过创建云播放器，可以实现向百度智能云 RTC 房间输入在线媒体流的能力。云播放器会将媒体流推送到指定的房间内，房间内其它用户可以观看该媒体流。
创建成功后，百度智能云 RTC 服务会返回云播放器 ID，作为 App 范围内云播放器的唯一标识。

请求（Request） 请求URI `POST /v1/cloudplayer HTTP/1.1`

请求头域 除公共头域外，无其它特殊头域。 **请求体** | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | playerName | String | 是 | 云播放器名称 | | appId | String | 是 | 应用 ID | | roomName | String | 是 | 房间名 | | userId | Long | 是 | 云播放器流在房间中 uid | | streamUrl | String | 是 | 流地址，现阶段只支持 http/https/rtmp/rtmps | | display | String | 否 | 云播放器流在房间中昵称 | | streamType | String | 否 | 流的类型，0表示点播，1表示直播。 | | streamMode | String | 否 | 输出流的模式，0表示音频+视频，1表示纯音频，2表示纯视频，默认值为0。 | | playTs | Long | 否 | 云端播放器开始播放输入的在线媒体流时间，单位秒，默认值为 0。
playTs=0，表示创建后立即播放。
playTs>0，则表示秒级别播放时间，且最大值为：当前秒级时间+600。 | | repeatTimes | Integer | 否 | 循环播放次数，仅对点播类型有效。取值范围[1,100]，默认值为 1。 | | maxIdleTime | Integer | 否 | 最大无流状态的超时时间，单位秒。取值范围 [5,60]，默认值为 30。 | | audioOptions | AudioOptions | 否 | 音频输出配置（音频输出编码统一为 opus，暂不支持配置） | | +bitrate | Integer | 否 | 音频码率，默认值为 50 kb/s。 | | +channel | Integer | 否 | 音频声道，支持取值 1 和 2。 | | videoOptions | VideoOptions | 否 | 视频输出配置（视频输出编码统一为 h264，暂不支持配置） | | +width | Integer | 否 | 视频宽，默认值为 1280。 | | +height | Integer | 否 | 视频高，默认值为 720。 | | +bitrate | Integer | 否 | 视频码率，默认值为 500kb/s | | dataStreamOptions | DataStreamOptions | 否 | 数据流配置 | | +enable | Boolean | 否 | 是否传入 DataStream 类型的 SEI 信息，默认值为 false。 | | +interval | Integer | 否 | 传输 StreamData 数据的 SEI 信息发送频率（毫秒），默认值为 1000，表示每隔 1 秒发送 1 个携带 StreamData 数据的 SEI 信息。 |

响应（Response） 响应头域 除公共头域外，无其它特殊头域。 **响应体** | 字段名 | 类型 | 描述 | | --- | --- | --- | | playerId | String | 云播放 ID，用于在 App 下标识唯一的云播放器实例 |

示例 请求示例

```
 POST /v1/cloudplayer HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "playerName":"player1",
    "appId":"testAppId",
    "roomName":"123456",
    "userId":123123,
    "streamUrl":"http://****/**",
    "display":"Jack",
    "streamType":"1",
    "streamMode":"0",
    "playTs":0,
    "repeatTimes":1,
    "maxIdleTime":30,
    "audioOptions":{
        "bitrate":50,
        "channel":1
    },
    "videoOptions":{
        "width":1280,
        "height":720,
        "bitrate":500
    },
    "dataStreamOptions":{
        "enable":false,
        "interval":1000
    }
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "playerId": "testPlayerId"
}
```

销毁云播放器

接口描述 调用此接口，传入应用 ID 和云播放器 ID，可以销毁指定的云播放器实例。 **请求（Request） 请求URI** `DELETE /v1/cloudplayer HTTP/1.1` **请求头域** 除公共头域外，无其它特殊头域。 **请求体** | 参数名 | 类型 | 是否必需 | 描述 | | --- | --- | --- | --- | | appId | String | 是 | 应用 ID | | playerId | String | 是 | 云播放器 ID | **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。 **响应体** 无 **示例 请求示例**

```
 DELETE /v1/cloudplayer HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
{
    "appId": "testAppId",
    "playerId": "testPlayerId"
}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
```

查询云播放器

接口描述

云播放器查询接口，可以查询云播放器的基础信息和状态。

请求（Request） 请求URI `GET /v1/cloudplayer?appId={appId}&playerId={playerId} HTTP/1.1`

参数名	类型	是否必需	描述
appId	String	是	应用ID
playerId	String	是	云播放器ID

请求头域 除公共头域外，无其它特殊头域。 **请求体** 无 **响应（Response） 响应头域** 除公共头域外，无其它特殊头域。 **响应体**
| 字段名 | 类型 | 描述 | | --- | --- | --- | | playerId | String | 云播放器 ID | | playerName | String | 云播放器名称 | | streamUrl | String | 流地址 | | roomName | String | 房间名 | | userId | Long | 云播放器 uid | | createTime | Long | 创建时间，unix 时间戳，单位毫秒 | | status | String | 云播放器状态，支持如下取值：
created：已创建；
idle: 空闲中；
running：运行中；
failed：播放失败；
deleted：已删除。|

示例 请求示例

```
 GET /v1/cloudplayer?appId=testAppId&playerId=testPlayerId HTTP/1.1
host: rtc.baidubce.com
content-type: application/json
authorization: {bce-authorization-string}
x-bce-request-id: {bce-request-id}
```

响应示例

```
 HTTP/1.1 200 OK
x-bce-request-id: b06a9214-04d6-4a08-9f5d-966b04604cfb
date: Mon, 05 Sep 2022 03:25:43 GMT
transfer-encoding: chunked
content-type: application/json;charset=UTF-8
cache-control: no-cache
{
    "playerId": "testPlayerId",
    "playerName": "player1",
    "streamUrl": "http://***/***",
    "roomName": "123456",
    "userId": 123123,
    "createTime": 1707031561638,
    "status": "running"
}
```

高级功能

服务端事件回调

云端录制回调

录制回调配置 百度智能云实时音视频 RTC 支持通过控制台自助配置录制回调地址，来接收云端录制事件回调。

配置录制回调步骤

1. 访问 RTC 控制台。
2. 访问 应用管理 -> 应用 -> 回调配置。
3. 编辑 录制回调，输入回调地址，点击确认。

注意事项

1. 录制回调地址必须是公网可以访问的 HTTP/HTTPS 地址；
2. 需要在开启录制、并且配置了录制回调的情况下，回调服务器才能收到百度智能云 RTC 服务器发送的录制事件回调。

回调内容 回调 HTTP 方法 百度智能云 RTC 服务器，向回调地址发送请求，统一使用 HTTP **POST** 方法。

回调请求体 录制事件回调请求体主要字段及描述如下：

字段名称	类型	描述
ts	Long	事件发送时间，为毫秒级别 Unix 时间戳
eventType	String	事件类型，当前仅支持取值 `RECORDING_FILE_UPLOAD_FINISH`
recordingMode	String	录制模式，"0"表示单路录制，"1"表示混流录制
appId	String	应用ID
roomName	String	房间名
clientUserIdList	List	参与混流的客户端用户ID列表，单路录制时列表中只包含单个用户，混流录制时包含所有参与混流的用户
eventInfo	EventInfo	事件信息。在不同类型事件中，事件信息字段不同，取值含义也不同
+eventTs	Long	事件发生时间，为毫秒级别 Unix 时间戳
+status	Integer	事件状态
+errMsg	String	错误信息。当 status 存在且不为 0 时，对应的错误信息

当 eventType=RECORDING_FILE_UPLOAD_FINISH 时，eventInfo 中除了包含 eventTs、status 和 errMsg 字段以外，还包含一些录制文件相关字段，具体字段及取值含义:

字段名称	名称	描述
eventTs	Long	事件发生时间，为毫秒级别 Unix 时间戳
status	Integer	事件状态。 0：代表录制文件正常上传至存储平台 1：代表录制文件滞留在服务器或者备份存储上
errMsg	String	错误信息。当 status 存在且不为 0 时，对应的错误信息
startTs	Long	录制文件开始时间，为毫秒级别 Unix 时间戳
endTs	Long	录制文件结束时间，为毫秒级别 Unix 时间戳
fileInfo	RecordingFileInfo	存储平台和文件信息
+vendor	String	存储平台，取 "0" 表示百度智能云对象存储 BOS
+host	String	存储平台服务地址
+bucket	String	存储文件 bucket
+filePath	String	文件路径
+fileUrl	String	文件下载 url。 需要注意的是，文件下载 url，目前仅适配百度智能云对象存储 BOS，并且不包含 url 鉴权等信息。

回调内容示例

```
{
  "eventType":"RECORDING_FILE_UPLOAD_FINISH",
  "appId":"testapp",
  "roomName":"654321",
  "recordingMode": "1",
  "clientUserIdList": [123456,123123],
  "eventInfo":{
     "eventTs":1663595954367,
     "status":0,
     "startTs":1663595909000,
     "endTs":1663595954224,
     "fileInfo":{
        "vendor":"0",
        "host":"bj.bcebos.com",
        "bucket":"test",
        "filePath":"/path/test.flv",
        "fileUrl":"https://test.bj.bcebos.com/path/test.flv"
     }
  },
  "ts":1663595945097
}
```

通知（回调）验证

百度智能云 RTC 服务可以通过发送通知的方式，向用户指定的服务地址，发送一些关键的事件，比如录制回调。
对于通知功能，客户可以通过开启验证，对通知模板设置密钥，并依据通知请求 header 中字段、body 内容、通知密钥等来判断该通知请求是否是合法请求。

- 由于在百度智能云 RTC 服务中，通知功能往往表现为相关事件回调。如不作特殊说明，本文中通知也可以称作回调。

1. 通知密钥 客户在创建通知时，可以设置开启验证，并设置密钥。
例如，对于录制回调而言，客户可以通过控制台应用管理 -> 应用 -> 回调配置，设置验证类型为签名验证，并设置回调密钥。

2. 开启验证的通知请求 对通知模板设置了开启验证后，百度智能云 RTC 服务在发送通知请求时，会携带如下3个 header：

- notification-auth-user：验证标识，目前固定为客户的百度云账号的 ID；
- notification-auth-expire：通知事件的过期时间戳（并非精准的过期时间，仅用于计算token）；
- notification-auth-token：使用特定算法生成的 token，用于客户侧校验通知请求的合法性。

3. notification-auth-token 计算过程 3.1 计算参数解释

- POST：通知请求使用的 http 方法，固定为 POST；
- endpoint：客户设置的用于接收通知的地址。比如对于录制回调而言，endpoint 就是回调地址；
- body：通知请求的 body 内容；
- notification-auth-expire：通知请求的 notification-auth-expire header；
- notification-auth-user：通知请求的 notification-auth-user header；
- authKey：客户设置的通知密钥。对于录制回调而言，authKey 就是回调密钥；
- SHA256-HEX：是一个Hash算法，详见：https://datatracker.ietf.org/doc/html/rfc4868 。

3.2 计算步骤 第一步计算 content。

```
content=POST;endpoint;body;notification-auth-expire;notification-auth-user
```

第二步计算 notification-auth-token。

```
notification-auth-token=SHA256-HEX(authKey,content)
```

3.3 计算示例 下面给出 Java 语言计算和验证 notification-auth-token 的代码示例，其他语言版本可以自行实现。

（1）引入 apache 工具包 这里借助 apache 工具包来实现 SHA256-HEX 算法，客户也可自行实现。

```
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.12</version>
</dependency>
```

（2）验证 notification-auth-token

```
@RequestMapping(value = "/***", method = RequestMethod.POST)
public boolean process(HttpServletRequest servletRequest, @RequestBody String callbackBody) {
    // http header 字段名不区分大小写
    String authExpire = servletRequest.getHeader("notification-auth-expire");
    String authUser = servletRequest.getHeader("notification-auth-user");
    String authToken = servletRequest.getHeader("notification-auth-token");
    log.debug("Received callback, authExpire={}, authUser={}, authToken={}, body={}",
            authExpire, authUser, authToken, callbackBody);

    if (StringUtils.isEmpty(authExpire) || StringUtils.isEmpty(authUser) || StringUtils.isEmpty(authToken)) {
        // header 中无相关验证信息，请确认该通知模板是否开启了验证，未开启验证的话无需校验验证信息。
        log.debug("Fail to auth callback, error auth params");
        return false;
    }

    // notificationEndpoint 和 authKey 是客户配置的通知地址和通知密钥。
    String notificationEndpoint = "http://***/***";
    String authKey = "testkey";
    String content = String.format("POST;%s;%s;%s;%s", notificationEndpoint, callbackBody, authExpire, authUser);
    String generatedToken = new HmacUtils(HmacAlgorithms.HMAC_SHA_256, authKey).hmacHex(content);
    log.debug("Generated auth token={}", generatedToken);

    if (!authToken.equals(generatedToken)) {
        log.debug("Fail to auth callback, token is error");
        return false;
    }

    // token 校验合法，处理通知请求的逻辑省略。。。
    return true;
}
```

云端旁路转推回调

旁路转推回调配置 百度智能云实时音视频 RTC 支持通过控制台自助配置旁路转推回调地址，来接收云端旁路转推事件回调。

配置旁路转推回调步骤

1. 访问 RTC 控制台。

2. 访问 应用管理 -> 应用 -> 回调配置。

3. 编辑 旁路转推回调，输入回调地址，点击确认。

注意事项

1. 回调地址必须是公网可以访问的 HTTP/HTTPS 地址；
2. 配置了旁路转推回调后，不论是通过服务端 OpenAPI 创建的云端旁路转推任务，还是通过 SDK API 发起的云端旁路转推任务，回调服务器都能收到百度智能云 RTC 服务器发送的旁路转推事件回调。

回调内容 回调 HTTP 方法 百度智能云 RTC 服务器，向回调地址发送请求，统一使用 HTTP **POST** 方法。

回调请求体 旁路转推事件回调请求体格式如下：

字段名称	类型	描述
version	String	事件结构的版本号，当前事件格式下取值固定为 "2"
eventId	String	事件 ID，用于日志追踪等
eventGroupId	int	事件分组，取值 1 时表示云端旁路转推事件
eventType	String	事件类型，支持的取值有：`LIVESTREAMING_BYPASS_STARTED`、`LIVESTREAMING_BYPASS_STATE_CHANGED` 和 `LIVESTREAMING_BYPASS_STOPPED`，具体事件描述见下文。
ts	Long	事件发送时间，为毫秒级别 Unix 时间戳
eventInfo	EventInfo	事件信息。对不同的事件分组、事件类型，事件信息也不同，旁路转推事件信息结构见下文。

EventInfo 结构 | 字段名称 | 类型 | 描述 | | --- | --- | --- | | appId | String | 应用ID | | roomName | String | 房间名 | | userId | Long | 客户端 uid。
对服务端 OpenAPI 形式启动的转推任务，只有单路转推任务的事件中会包含该字段。
对 SDK API 形式启动的转推任务，只有主播模式转推任务的事件中会包含该字段。| | taskId | String | 服务端 OpenAPI 形式转推任务 ID。
SDK API 形式启动的转推任务的事件中无该字段。 | | pushUrl | String | 推流地址 | | state | String | 推流状态，`LIVESTREAMING_BYPASS_STATE_CHANGED` 事件中会包含该字段。支持的取值和含义：
`RUNNING`：推流中；
`RECOVERING`：推流中断后正在重连；
`FAILURE`：推流失败。|

云端旁路转推事件描述 | 事件 | 含义 | 详情 | | --- | --- | --- | | LIVESTREAMING_BYPASS_STARTED | 云端旁路转推启动事件 | 对于服务端 OpenAPI 创建的旁路转推任务，在调用 start 接口启动转推任务时，会触发该事件。
对于 SDK API 发起的旁路转推任务，在转推任务被启动、并且相关客户端实际推流时，会触发该事件。 | | LIVESTREAMING_BYPASS_STATE_CHANGED | 云端旁路转推状态变化事件 | 当转推任务正常进行时，转推状态发了变化会触发该事件。
转推状态变化原因主要包括：直播服务器主动断开推流、百度服务器网络异常导致重连等。 | | LIVESTREAMING_BYPASS_STOPPED | 云端旁路转推结束事件 | 对于服务端 OpenAPI 创建的旁路转推任务，在调用 stop 接口停止转推任务、或者相关客户端无流时间超出最大续播时间等原因导致任务结束时，会触发该事件。
对于 SDK API 发起的旁路转推任务，在调用停止转推任务、或者相关客户端推流退出导致任务结束时，会触发该事件。 |

回调内容示例 转推启动事件示例

```
{
 "version" : "2",
 "eventGroupId" : 1,
 "eventType" : "LIVESTREAMING_BYPASS_STARTED",
 "eventId" : "testEventId",
 "ts" : 1718958094142,
 "eventInfo" : {
  "appId" : "testapp",
  "roomName" : "654321",
  "userId" : 123456,
  "taskId" : "testTaskId",
  "pushUrl" : "rtmp://***/***/***",
  "eventTs" : 1718958094128
 }
}
```

转推状态变化事件示例

```
{
 "version" : "2",
 "eventGroupId" : 1,
 "eventType" : "LIVESTREAMING_BYPASS_STATE_CHANGED",
 "eventId" : "testEventId",
 "ts" : 1718958096395,
 "eventInfo" : {
  "appId" : "testapp",
  "roomName" : "654321",
  "userId" : 123456,
  "taskId" : "testTaskId",
  "pushUrl" : "rtmp://***/***/***",
  "state" : "RUNNING",
  "eventTs" : 1718958096000
 }
}
```

转推结束事件示例

```
{
 "version" : "2",
 "eventGroupId" : 1,
 "eventType" : "LIVESTREAMING_BYPASS_STOPPED",
 "eventId" : "testEventId",
 "ts" : 1718958158093,
 "eventInfo" : {
  "appId" : "testapp",
  "roomName" : "654321",
  "userId" : 123456,
  "taskId" : "testTaskId",
  "pushUrl" : "rtmp://***/***/***",
  "eventTs" : 1718958158093
 }
}
```

下载专区

RTC SDK&Demo下载

合规指南

- 开发者： 北京百度网讯科技有限公司
- SDK名称： 百度云实时音视频SDK
- 版本号： Android SDK: 3.4.0; iOS SDK: 3.3.0;
- 主要功能： 提供稳定、超低延时、高质量的实时音视频通信服务
- 个人信息处理规则： 百度云RTC SDK 隐私政策
- 合规使用说明： 百度RTC SDK合规指南

下载SDK

本文档提供最新版本的 SDK。如果您需要老版本的 SDK，请联系技术支持获取。

平台	SDK文件名	文件MD5	下载
Android	BRTC.Android.SDK.v3.4.0.zip	344bb6ae5892d543e788d7b97c29b388	点击下载
Android 纯音频	android-pure-audio_1.0.0.4_com.baidu.rtc.aar	9a9e48d7f84e67be6d1ca54bffad3940	点击下载
iOS	BRTC.iOS.SDK.v3.3.0.zip	cade50324dd974243f318f7199dcf21f	点击下载
Windows	rtc-windows-sdk-2.3.0.0.zip	6ce7144276298a0522df835a3f4adfcc	点击下载
macOS	rtc_mac_demo_1.7.8.zip	bbb482b276243e58adae42cdcd0919cb	点击下载
Web	BRTC.Web.SDK.V1.2.3.zip	ac4726fe8082cf7a29e60ec2592f4805	点击下载
	从CDN引入 或 npm方式接入		点击跳转
Linux	BRTC.Linux.SDKV0.2.20.tar.gz（x86_64）	93366f6ef33f548373d7e014828cfeb6	点击下载
	BRTC.Linux.SDKV1.0..tar.gz (x86_64)多卡版本	0bc91d3e9d344b99ccd8a4a0d2b2b9dc	点击下载
	BRTC.Linux.SDK.armv7-softfp.V0.2.20.tar.gz	231ee52c2b0a1c4bb20c6e337083ed38	点击下载
	BRTC.Linux.SDK.armv7-soft-uclibc.V0.2.20.tar.gz	9f19be91364940a7d3435caed5bae50d	点击下载
	BRTC.Linux.SDK.armv7.V0.2.20.tar.gz	de656b6df15f764100878b8ccddce8c0	点击下载
	BRTC.Linux.SDK.arm64.V0.2.20.tar.gz	baed872df9b95ab068a3f12178c37b1f	点击下载
	rtc-linux-arm64-sdk-v1.0.2.tar.gz（多卡版本）	9a24a7440840e7996fc624c4c53731aa	点击下载
Java	rtc-java-linux_v1.0.3.tar.gz	e9dfc34f8c2cf9fccaa6b09bcf7ee64f	点击下载
微信小程序	BRTC.MiniApp.SDKV1.0.2.zip	6d26d15a2933ba38a7d3a5f7f1811a6e	点击下载
RTOS	BRTC.RTOS.SDK.V2.0.0.3_20221118.zip	e7dc5ec7b797f8817e7738b328f43b2a	点击下载
Flutter	BRTC.Flutter.SDK.V1.0.0.zip	4627b886ec480781ee1a8d01a8f4e515	点击下载

下载快速开始Demo

快速开始 Demo 是 RTC 提供的基本音视频通话功能的开源示例工程文件。获取该工程文件后，您可以快速构建应用，感受 RTC 的通话效果；也能通过阅读代码，了解基本音视频通话的最佳实践。

平台	SDK文件名	文件MD5	下载
Android	BRTC.Android.SDK.V3.4.0_fastdemo.zip	5bf7ca000a53de75dc44579cc3bd2e2c	点击下载
Android 纯音频	android-pure-audio_1.0.0.4_com.baidu.rtc.aar	9a9e48d7f84e67be6d1ca54bffad3940	点击下载
iOS	BRTC.iOS.SDK.V3.3.0_fastdemo.zip	b64df850058d01e1112635bb2bcbbbc9	点击下载
Windows	rtc-windows-release-fastdemo-2.3.0.0.zip	53964968b9cac5a0486f3948985d1771	点击下载
macOS	rtc_mac_demo_1.7.8.zip	bbb482b276243e58adae42cdcd0919cb	点击下载
Linux	BRTC.Linux.SDKV0.2.20.tar.gz（x86_64）	93366f6ef33f548373d7e014828cfeb6	点击下载
	BRTC.Linux.SDK.armv7-softfp.V0.2.20.tar.gz	231ee52c2b0a1c4bb20c6e337083ed38	点击下载
	BRTC.Linux.SDK.armv7-soft-uclibc.V0.2.20.tar.gz	9f19be91364940a7d3435caed5bae50d	点击下载
	BRTC.Linux.SDK.armv7.V0.2.20.tar.gz	de656b6df15f764100878b8ccddce8c0	点击下载
	BRTC.Linux.SDK.arm64.V0.2.20.tar.gz	baed872df9b95ab068a3f12178c37b1f	点击下载
微信小程序	BRTC.MiniApp.SDKV1.0.2.zip	6d26d15a2933ba38a7d3a5f7f1811a6e	点击下载
RTOS	BRTC.RTOS.SDK.V2.0.0.3_20221118.zip	e7dc5ec7b797f8817e7738b328f43b2a	点击下载
Flutter	BRTC.Flutter.SDK.V1.0.0.zip	4627b886ec480781ee1a8d01a8f4e515	点击下载

下载进阶功能Demo

进阶功能 Demo 是 RTC 提供的高级功能的开源示例工程文件。获取该工程文件后，您可以了解例如屏幕分享、自定义消息、美颜等功能的用法，使用这些能力实现更复杂的业务场景。

平台	SDK文件名	文件MD5	下载
Android	BRTC.Android.SDK.V3.4.0.demo.zip	a729b8030ffcd498b7c374c33308477f	点击下载
iOS	BRTC.iOS.SDK.V3.3.0_demo.zip	9e9d8035079de71929a7cce0a492d379	点击下载
Windows	rtc-windows-release-2.3.0.0.zip	712a35f9d34c14c4069bcdbf2c03c182	点击下载
Web	BRTC.Web.SDK.V1.2.13.Meeting_demo.zip	d31326a51288a180bcce224f614bf838	点击下载

Demo体验

平台	下载地址
Andriod	
iOS	
Web	点击体验

相关协议

实时音视频RTC SDK隐私政策

欢迎使用实时音视频RTC软件开发工具包（SDK）（简称“实时音视频RTC SDK”）服务！

实时音视频RTC SDK 是一款为移动应用开发者（以下简称“开发者”）提供稳定、超低延时、高质量的实时音视频通信服务的软件开发工具包（SDK）。开发者在其移动应用内集成实时音视频RTC SDK后，可通过实时音视频RTC SDK平台向其移动应用（以下简称“开发者应用”）的最终用户（以下简称“最终用户”）提供本隐私政策所说明的功能及服务。开发者在其移动应用集成并使用实时音视频RTC SDK服务时，委托实时音视频RTC SDK处理开发者应用相关数据信息，其中可能包括开发者应用最终用户（以下简称“最终用户”）的个人信息。此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策“附录1：名词解释”来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用实时音视频RTC SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用实时音视频RTC SDK服务的情况，以及实时音视频RTC SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于实时音视频RTC SDK收集、使用最终用户相关个人信息的完整、合法、在使用实时音视频RTC SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于实时音视频RTC SDK收集、使用最终用户相关个人信息的完整、合法、在使用实时音视频RTC SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

（4）关于上述承诺的具体落地执行可参考《实时音视频RTC SDK开发者个人信息保护合规指引》。

2、我们希望集成并使用实时音视频RTC SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的实时音视频RTC SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，实时音视频RTC SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用实时音视频RTC SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订
8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用实时音视频RTC SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	网络质量监测，根据网络状态调整网络策略	WIFI状态（必选）	SDK直接采集	iOS及Android
2	网络质量监测，在WIFI切换时实现快速断网重连	WIFI-SSID（必选）	SDK直接采集	iOS及Android
3	设备性能检测，为了更好的用户体验，进行机型分级，针对不同的机型的性能提供不同的功能模型及服务策略	设备型号（必选）	SDK直接采集	iOS及Android
4	SDK通信质量检测	IP地址（必选）	SDK直接采集	iOS及Android

2. 设备权限调用

为了保证最终用户能正常使用实时音视频RTC SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能，各项功能及功能对设备权限的调用情况如下：

Android系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
录屏	录屏功能，采集屏幕数据	同上

iOS系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
录屏	录屏功能，采集屏幕数据	同上
相册	录制、截屏功能将视频和图片存储到相册	同上

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用实时音视频RTC SDK产品和服务时遇到的问题，改进及优化实时音视频RTC SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用实时音视频RTC SDK服务时遇到的问题，改进及优化实时音视频RTC SDK的服务体验，提高实时音视频RTC SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；

5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用实时音视频RTC SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与实时音视频RTC SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息：

仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商

或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；
2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；
3. 如果我们或我们的关联公司涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；
2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；
6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；
8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在实时音视频RTC SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按

照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。
2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。
3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系，**并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。
4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过实时音视频RTC SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加密，注意确保您个人信息的安全。**
5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**
6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案，**由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。
2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。
3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

实时音视频RTC SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成实时音视频RTC SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及《百度儿童个人信息保护声明》，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过工单联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用实时音视频RTC SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；

2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；

3. 如果我们停止提供产品或者服务，或者保存期限已届满；

4. 如果最终用户撤回同意；

5. 法律、行政法规规定的其他情形。

当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

实时音视频RTC SDK愿一直陪伴开发者和最终用户，若因特殊原因导致实时音视频RTC SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过【八、如何联系我们】中的方式与我们取得联

系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在实时音视频RTC SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用实时音视频RTC SDK产品和服务，若开发者继续集成并使用实时音视频RTC SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用实时音视频RTC SDK服务期间持续有效的授权同意。**若最终用户不同意该等变更，可以停止使用实时音视频RTC SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用实时音视频RTC SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；
2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；
3. 个人信息共享、转让或公开披露的主要对象发生变化；
4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；
5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；
6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

实时音视频RTC SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对实时音视频RTC SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过工单反馈开发者及最终用户对实时音视频RTC SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台(http://help.baidu.com/personalinformation) 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信

息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。 敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行实时音视频RTC SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等唯一设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问实时音视频RTC SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的**精确位置信息**，也包括通过IP地址或其他网络信息等获取的**粗略地理位置信息**。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。

匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度手机助手及其他百度系产品https://www.baidu.com/more/)等网站、程序、服务、工具及客户端。

关联公司是指实时音视频RTC SDK的经营者【北京百度网讯科技有限公司】及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对实时音视频RTC SDK的信任和使用！

北京百度网讯科技有限公司

【2023】年【12】月【15】日更新

【2023】年【12】月【15】日生效

实时音视频RTC SDK开发者个人信息保护合规指引

亲爱的开发者：

感谢您在所开发的移动应用中集成并使用百度旗下软件开发工具包（SDK）。

百度非常重视用户个人信息保护，包括集成百度旗下实时音视频RTC软件开发工具包（SDK）（以下简称“百度SDK”）的移动应

用的最终用户（以下简称“最终用户”）个人信息保护，特制定《实时音视频RTC SDK个人信息保护合规开发者指引》，以供您在所开发的移动应用（以下简称“移动应用”）中集成并使用百度SDK时参照执行。

一、 个人信息保护合规基本要求

在所开发的移动应用中集成并使用百度SDK，您需要首先遵守以下个人信息保护合规基本要求：

1. 您应遵守收集、使用最终用户个人信息有关的所有可适用法律法规及规范性文件要求，包括但不限于《个人信息保护法》、《网络安全法》、《App违法违规收集使用个人信息行为认定方法》、《工业和信息化部关于开展APP侵害用户权益专项整治工作的通知》（工信部信管函［2019］337号）、《工业和信息化部关于开展纵深推进APP侵害用户权益专项整治行动的通知》（工信部信管函〔2020〕164号）、《工业和信息化部关于进一步提升移动互联网应用服务能力的通知》（工信部信管函〔2023〕26号）等，保护用户个人信息安全。

2. 您的APP需要制定一份独立的隐私政策。隐私政策的内容建议通俗易懂，对您的APP收集、使用个人信息的目的、方式、范围，清晰、完整、准确地向个人信息主体进行明示告知，并充分给予最终用户独立的选择权，确保在获得个人信息主体授权同意后方可进行个人信息的处理活动。《隐私政策》应由用户自主选择是否同意，不应以默认勾选同意的方式或是以欺骗诱导的方式取得用户授权。但请您注意，仅是改善服务质量、提升用户体验、定向推送信息、研发新产品还不足以能成为要求用户同意收集其个人信息的理由。

3. 您应将在移动应用中集成并使用百度SDK服务的情况，以及百度SDK对最终用户必要个人信息的收集、使用和保护规则（具体请见实时音视频RTC SDK隐私政策），在移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（具体请参考本指引第二部分 “如何告知最终用户”及第三部分“告知文案示例”），并获得最终用户对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。如果最终用户是未满14周岁的未成年人，请您务必确保获得最终用户的父母或其他监护人对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。

4. 您应向最终用户提供易于操作的访问、更正、删除其个人信息，撤销或更改其授权同意、注销其个人账号等用户权利实现机制。

5. 您应确保在移动应用首次运行时，应在最终用户阅读并同意移动应用隐私政策之后，方可初始化百度SDK进行最终用户信息采集。

6. 百度SDK会根据产品升级优化、提升安全性能、法律及监管要求等原因，不断升级迭代SDK版本，不同版本的SDK获取的字段信息可能会有差异。为了保证合法合规开展合作，并切实履行保护用户个人信息的责任与义务，请您务必确保您已将APP中的百度SDK升级至官方最新版本，以避免因使用旧版本SDK而出现违法违规的问题，导致您的APP遭受监管处罚的风险。百度SDK更新后会及时通过官网通知、站内信、公告、邮件、短信等有效方式对您进行告知，请您及时关注并尽快更新。

除了上述个人信息保护合规基本要求外，您还应遵守您所开发的移动应用所集成并使用的实时音视频RTC SDK隐私政策。

SDK基本业务功能与拓展业务功能：

实时音视频RTC SDK基本业务功能为提供音视频通话、屏幕共享、水印、数据统计、质量监控。实时音视频RTC SDK扩展业务功能为云端录制、旁路转推、云播放器、美颜、变声、背景分割、音视频审核等。

以下是实时音视频RTC SDK获取您的APP的最终用户的个人信息以及权限，由于不同SDK版本采集的字段与是否可选可能存在一定差异，具体采集情况以实际接入的SDK版本为准：

SDK收集个人信息情况：

功能及服务	个人信息类型	个人信息字段（区分必选信息和可选信息）
网络质量监测，根据网络状态调整网络策略	WiFi状态	WiFi状态（必选）
网络质量监测，在WIFI切换时实现快速断网重连	WiFi-SSID	WiFi-SSID（必选）
设备性能检测，为了更好的用户体验，进行机型分级，针对不同的机型的性能提供不同的功能模型及服务策略	设备型号	设备型号（必选）
SDK通信质量检测	IP地址	IP地址（必选）

安卓操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集视频数据，进行视频通话	登录房间时
录音	开启麦克风	用于采集音频数据，进行音频通话	登录房间时
录屏	录屏功能	用于采集屏幕数据，共享屏幕给远端	共享屏幕时

IOS操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集视频数据，进行视频通话	登录房间时
录音	开启麦克风	用于采集音频数据，进行音频通话	登录房间时
录屏	录屏功能	用于采集屏幕数据，共享本地屏幕给远端	共享屏幕时
相册	存储图片或视频	录制或截屏时，将图片或视频存储在相册中	启动本地录制或截屏时

SDK初始化设置：

请务必确保您已将实时音视频RTC SDK升级到最新版。上述版本调用初始化函数不会采集用户个人信息，也不会向百度联盟后台上报数据。请确保用户同意《隐私政策》后再进行start/autoTrace采集配置，方可采集用户个人信息并上报数据。

调用初始化函数文档： https://cloud.baidu.com/doc/RTC/s/Hk0gh4qck#sdk%E5%88%9D%E5%A7%8B%E5%8C%96

二、 如何告知最终用户

为帮助您明确地告知最终用户百度SDK个人信息收集、使用和保护相关事宜，我们为您提供了以下告知方式，供您参考执行：

1. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并显示相应**百度SDK隐私政策链接**以告知最终用户，百度SDK收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

2. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并参考相应百度SDK隐私政策内容，**以条款或表格等形式列明**收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

3. 当最终用户在移动应用中首次打开/使用相应功能/服务时，以**弹窗、页面提示**方式显示相应**百度SDK隐私政策链接**，以告知最终用户相应功能/服务由百度SDK提供，百度SDK为提供相应功能/服务而收集、使用的最终用户个人信息类型、目的及用途，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

三、 告知文案示例

为帮助您明确地告知最终用户百度SDK个人信息保护规则相关事宜，我们为您提供了以下告知方文案示例，供您参考执行：

文案示例A

为向您提供实时音视频功能/服务，我们集成了实时音视频RTC SDK。在为您提供实时音视频功能/服务时，实时音视频RTC SDK需要收集、使用您必要的个人信息。关于实时音视频RTC SDK收集、使用的个人信息类型、目的及用途，以及实时音视频RTC SDK将如何保护所收集、使用的个人信息，请您仔细阅读《实时音视频RTC SDK隐私政策》了解。

文案示例B

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	使用目的	官网链接/隐私政策链接
实时音视频RTC SDK	提供实时音视频功能/服务	https://cloud.baidu.com/doc/RTC/s/ilq6a9x9r

文案示例C

为向您提供实时音视频功能/服务，我们集成了实时音视频RTC SDK。在为您提供实时音视频功能/服务时，实时音视频RTC SDK收集、使用您的wifi状态、设备型号、ip地址信息，用于网络质量、设备性能、SDK通信质量检测目的/用途，具体请您仔细阅读《实时音视频RTC SDK隐私政策》了解。

文案示例D

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	业务功能	个人信息类型	调用权限类型	具体目的/用途
实时音视频RTC SDK	音视频通话	WIFI状态、WIFI-SSID、设备型号、IP地址	相机、录音、录屏、相册	用于通信质量和设备性能检测，保障用户正常使用音视频通话、本地录制及屏幕共享等功能

四、 使用SDK服务的合规注意事项

1. 您接入百度SDK前，应当仔细阅读SDK的协议约定、本合规规范、用户协议、隐私政策等内容，并依据相关内容对您APP的《隐私政策》及您APP收集、存储、使用、共享等处理个人信息的情况进行合规自查。
2. 您知悉并认可百度SDK具备收集个人信息的功能，并认可该等信息的收集均为双方合作之必要目的所需。
3. 您承诺已制定并按照相关要求公示您APP的《隐私政策》，并已清晰、明确、显著地说明有关通过SDK收集个人信息的必要性、收集数据的范围、方式以及用途。同时，您应确保在APP首次运行时以弹窗等合规方式显著提示用户阅读您APP的《隐私政策》并取得用户的合法授权同意，经过合法授权后再初始化百度SDK进行个人信息的收集与处理。
4. 您已认真阅读并理解百度SDK平台协议、合作规范、隐私政策、接入文档等约定和要求，并承诺在您使用百度SDK服务期间，针对百度SDK收集、使用、处理、共享、转让相关个人信息，您已取得了用户持续有效的授权和同意，并保证您不会违反国家相关法律法规、相关国家标准以及双方约定的目的。
5. 如果您的APP面向不满十四周岁的儿童及未成年人用户提供服务，您承诺遵守儿童个人信息保护及未成年人保护相关的法律法规，您承诺已采取相关措施并保证已获得其监护人的授权同意。
6. 如因您违反百度SDK的平台协议、合作规范、隐私政策、接入文档等约定，导致您的用户或第三方对百度主张任何形式的索赔或权利要求，或导致百度因此产生任何法律纠纷的，您将负责解决并承担全部责任，如因此给百度及其关联主体造成损失的，您应赔偿因此给百度及其关联主体造成的全部损失。
7. 您保证对于您从百度SDK获取的数据，无论是在合作期间或是合作停止后，均承担保密义务，不擅自提供、泄露、透露给任何第三方，并应采取一切合法措施以使上述数据免于散发、传播、披露、复制、滥用及被无关人员接触，避免导致相关数据被超出双方合作目的使用。

常见问题

功能相关

RTC与音视频直播LSS的区别？

音视频直播LSS：用于直播流的分发，单向分发，CDN架构，LSS可以理解为直播CDN，延迟3秒~10秒； 实时音视频RTC：多向传输架构，支持双向通信，也可以做单向直播，延时300ms以内

RTC产品必须要接入客户端SDK吗？

必须。因为RTC实时音视频通信包括信令的通信和媒体流的通信，而信令并没有统一的标准，所以每一个厂商都会自己定义客户端与服务端交互的信令，所以使用哪一家的RTC服务，必须要集成对应的客户端SDK

RTC产品支持私有化吗？

支持

RTC支持哪些客户端？

支持Android、IOS、Web、Windows、MacOS、微信小程序、Flutter、Linux、Rtos等多平台互通

RTC可以体验Demo吗？

RTC提供Android、IOS、Web端Demo进行体验，具体请参见：https://cloud.baidu.com/doc/RTC/s/llgunowoj

是否支持同时接入多路车载摄像头的画面吗？

支持

性能相关

RTC的性能指标？

- 端到端延时：300ms以内
- 抗弱网：音视频抗丢包达70%，抗网络抖动达1000ms
- 首屏时长：1s以内
- SDK CPU占用：1v1通信场景，Android端中高端机型CPU占用15%，iOS端中高端机型CPU占用40%
- SDK内存占用：1v1通信场景，Android端中高端机型内存占用80MB，iOS端中高端机型内存占用60MB

Android 与 IOS 相关

BRTC Android 端能不能支持64位的 arm64-v8a 架构？

支持

BRTC 移动端怎么实现录屏（屏幕分享）？

Android 端：Version 2.8 及以上版本支持手机录屏，支持使用多进程方式屏幕分享；
iOS 端：Version 2.8 及以上版本支持 App 内录屏和手机录屏两种方式;

是否支持 Android、iOS 和 Web 端互通？

支持， 使用相同AppID进入相同房间即可以互通

是否支持纯音频方式？

支持
Android端：配置关闭视频，`RtcParameterSettings#HasVideo = false;`
iOS端：配置关闭视频，`RtcParameterSettings#hasVideo = false;`

如何查询SDK版本号？

Android端：`String version = BaiduRtcRoom.version();`

iOS端：`NSString *version = [BaiduRtcRoomApi  version];`

是否支持转推RTMP？

支持，双端支持配置转推地址，支持按照模板转推；

BRTC 是否支持原生鸿蒙系统？

如果有对接需求可以先使用 BRTC web SDK，webrtc 在鸿蒙的 webview 里可以使用，暂不支持屏幕共享。

集成RTC SDK后，包体增加多少？

安卓端app体积增加5M左右，IOS端app体积增加6M左右