网易智企 | 网易云信

音视频通话 2.0 开发指南

版本号：V4.6.10

发布日期：2022-08-15

目 录

一、 产品简介

1.1 产品介绍

音视频通话 2.0（NetEase Real-Time Communication，NERTC）是网易云信推出的实时音视频开发平台。网易云信基于网易多年的即时通讯和实时音视频通话能力的技术积累，为您提供稳定流畅、高品质、全平台的点对点和多人实时音视频通话服务。

网易云信音视频通话 2.0 适用于各种实时音视频场景，例如社交行业的视频聊天、视频相亲、视频交友、在线 KTV，教育行业的小班教学、一对一视频教学，视频会议、远程医疗、游戏语音等。针对不同场景，网易云信提供了通过 NERTC SDK 功能实现的一系列产品、技术解决方案，包括：点对点通话、多人通话、互动直播、高接通方案等。此外，我们还可以根据您后续的内部需求，为您打造企业专属的音视频通话服务。

经过多年在社交娱乐、教育等行业的深耕与功能打磨，音视频通话产品现已全新升级至音视频通话 2.0。

作为音视频通话的全新品牌升级产品，音视频通话 2.0 采用新一代音视频技术架构进行了全流程的技术升级，包括新一代音视频融合通信服务端系统、音视频 SDK 以及新一代音视频引擎、网易云信深度优化的自研音视频编码器。音视频通话 2.0 在功能和性能等多方面进行大幅度能力提升：

- 接入更加便捷。

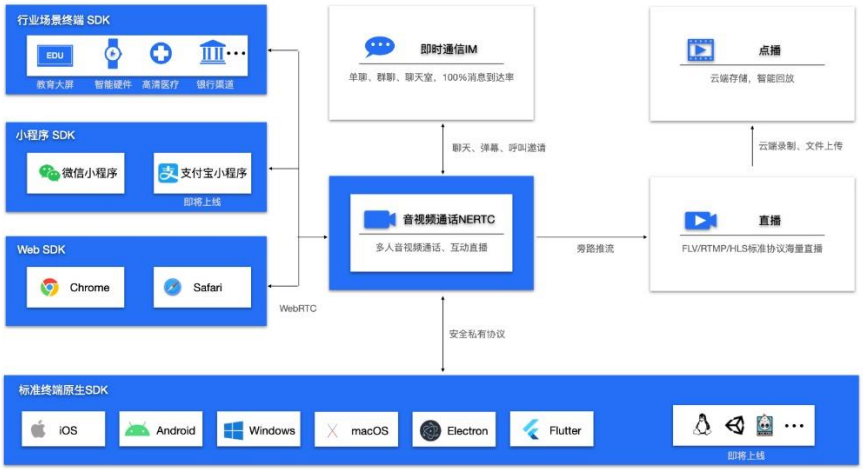
  在接入的便捷性上，音视频通话 2.0 进行了进一步优化，更加简单易用的接口和 SDK 接口层统一调用逻辑，帮助开发者降低接入成本，提升研发效率。

- 更高质量的音视频通话体验。
  - 端到端延时小于 200 ms，最高可抗 1000 ms 网络抖动，在网络丢包 80% 时仍能正常通话。
  - 视频清晰度提升至 1080p，音频采样率最高支持 48kHz，支持全频带编解码和智能语音前处理算法。
  - 保障用户在各种场景下流畅高品质的音视频体验，尤其是在音乐场景的特殊优化，可保证经过网络传输的音乐仍能保持 CD 级高音质。

1.2 产品架构

网易云信 NERTC SDK 主打全平台互通的多人音视频通话，提供多平台框架的 SDK，包括 Android、iOS、Windows、macOS、Linux、Web、小程序等平台，Unity、Flutter 和 Electron 框架，以便开发者快速集成并实现多种业务场景。通过更多的高级功能和扩展功能，开发者可以轻松的扩展出更加丰富的业务场景。

音视频通话 2.0（NERTC SDK）的产品架构如下图所示。



1.3 功能特性

音视频通话 2.0 支持一对一单聊和多人群聊，可实现纯语音通话和视频通话功能。

1.3.1 基础功能

功能	功能说明	典型适用场景
语音通话	支持一对一或多人音频聊天。 具有默认流畅、低延时优先的特征，所有用户都可以发言。	语音社交 语音会议 语音客服
视频通话	支持一对一或多人的音视频通话。	视频通话 视频客服

	具有默认流畅、低延时优先的特征，所有用户都可以自由沟通。	1 对 1 教学

1.3.2 高级功能

高级功能是 NERTC SDK 提供的各类音视频高级功能，帮助开发者实现特殊场景中的功能。

主要功能	功能说明	典型适用场景
高音质	支持 48 kHz 采样、128 kbps 码率双声道的立体声高音质。 采用业内 Top 级的实时高音质，为用户带来逼真的听觉感受。	在线 K 歌 多人语聊房 音乐直播 音乐教学
高画质	支持 1080P 的高清画质视频。 高清视频带给用户“身临其境”的视觉体验。	视频通话 互动直播 在线教育 智慧医疗
互动连麦	支持观众和主播连麦聊天、其他观众围观、上下麦切换平滑。丰富观众和主播的	互动直播

	互动性。	在线教育 多人语聊房
本地服务端录制	支持在客户本地服务器上进行音视频的录制。 将音频内容、视频内容存储在本地，保障数据的隐私性。 如需使用，请联系云信商务经理。	金融双录 远程业务存档
消息抄送	支持将实时音视频、互动直播、云端录制的消息/事件等数据同步给第三方开发者服务器。 满足通信场景、直播场景的问题排查、用户在线状态分析、特殊事件处理等需求，以及录制场景中录制文件问题排查的诉求。	视频通话 互动直播 在线教育 智慧医疗
伴音	支持在用户声音中加入本地音乐文件或在线音频，一起发送给房间内其他用户播放。丰富语音玩法的多样性，能够帮助主播更好地吸引观众。	在线 K 歌 多人语聊房 音乐直播 音乐教学

音效	支持通话中播放自定义的较短持续时间的氛围音，例如鼓掌、欢呼。通过增加音频沟通过程的趣味性，带动房间内的气氛，帮助主播更好地和观众互动。	在线 K 歌 多人语聊房 互动直播 在线教育
美声	支持多种美声效果，例如浑厚、低沉、高亢等效果。突出主播声音的特性，有利于帮助主播打造独特的嗓音。	在线 K 歌 音乐直播 多人语聊房
变声	支持多种变声特效，例如萝莉、大叔等效果。通过增加音频沟通过程的趣味性，来带动房间内的气氛。帮助主播更好地与观众互动。	在线 K 歌 秀场直播 多人语聊房
混响	支持多种混响效果，例如演唱会、录音棚、KTV 等效果。帮助主播模拟出不同环境中的声音效果，使主播的声音更加有现场感。	在线 K 歌 音乐直播 多人语聊房 远程会议
耳返	采集声音监听，让主播听见自己发出的声音。帮助主播在演唱的同时获得一个真	音乐直播 在线 K 歌

	实、及时的反馈，用来鉴定音准。	多人语聊房
设置音频属性	支持设置音频编码属性，实现不同的音质效果，适用于不同场景的音质、声道需求。	音乐直播 在线 K 歌 多人语聊房 互动直播 视频会议 在线教育
设置视频属性	支持设置视频编码属性，根据不同场景下的用户喜好与需求，调整发送视频画面的清晰度以及流畅度，带给用户最优的通话体验。	互动直播 秀场直播 视频会议 在线教育
自定义音频数据	支持第三方音频源的音频输入，例如本地音频流、第三方音频采集模块。提高音频链路模块之间的灵活性。具有定制能力的客户可以根据自己需要实现音频采集模块并能通过云信进行音频传输。	语音处理 自定义音频效果 非标音频设备接入

自定义视频数据	支持多格式的自定义的视频源输入，满足多场景的非摄像头的视频源的输入，例如自定义视频文件、外接视频设备、自定义的美颜库或有前处理库等。丰富互动直播、视频会议等场景的用户体验。	第三方视频前处理 自定义数据输入源 灵活的设备管理 图像识别
音频设备管理	支持自由切换音频设备，客户可以根据实际的场地情况选择音频的采集和播放设备，比如听筒、蓝牙耳机、扬声器或有线耳机。	灵活的设备管理
视频设备管理	支持整套的摄像头管理方法，以便用户灵活切换前、后置摄像头，并对摄像头进行缩放、对焦和曝光等相关配置。	摄像头管理
屏幕共享	支持将本地电脑屏幕、应用窗口、指定画面区域分享给远端用户。通过屏幕共享的功能，以视频画面的方式分享给其他参会者或观众观看，以提高沟通效率。	视频会议 远程协作 在线教育 智慧医疗

视频水印	支持在视频的任意位置添加和删除水印。 视频水印功能，可以用于产品的版权保护、广告宣传、录制文件信息记录等。 如需使用，请联系云信商务经理。	视频通话 互动直播 在线教育 智慧医疗
视频截图	支持通过视频截图功能截取实时视频流画面，以便后续的存档分析、事件备忘、证据留存。	在线教育 安全审核
音视频加密	支持音视频通信媒体流加密功能。保证数据流的隐私，提高安全性。 如需使用，请联系云信商务经理。	视频会议 金融双录 远程协作
视频流回退	支持开启大小流模式，可指定弱网环境下本地或接收端媒体流的回退情况，优先保障通话流畅平稳、维持用户体验。	视频会议 远程会议
跨房间媒体流转发	支持将主播角色的媒体流同时转发至多个房间中，实现跨房间实时互动的场景。	在线教育 互动直播
加入多房间	支持用户同时加入多个房间，提供多房间	在线教育

	管理功能，隔离多个房间的消息和回调。	互动直播
通话前网络质量探测	支持在通话前进行网络调试，例如检测网络质量，提前识别网络问题，保证高质量的音视频通话体验。	视频会议 远程会议
通话中质量监测	支持提供当前通话的通话质量、设备状态等信息，帮助用户监测通话的整体质量。	视频会议 远程会议
媒体补充增强信息	支持将自定义信息通过流媒体通道与视频内容打包在一起，发送给远端用户，实现音画同步。	在线教育 互动直播 音乐直播 在线 k 歌

1.3.3 扩展功能

扩展功能是 NERTC SDK 与其他云产品一起为开发者提供的增值服务，将由其他云产品根据各自的计费规则分别收取相关费用。

主要功能	功能说明	典型适用场景

旁路推流	支持将音视频房间中多主播的媒体流合成单路流，并经 CDN 转发，实现 CDN 直播。 在该模式下，观众数量无上限。 网易云信通过互动直播 2.0 产品提供旁路推流功能，详细信息请查看互动直播 2.0。	互动直播 大型会议 直播 PK 大班课
云端录制	支持帮助开发者快速、灵活地实现云端录制服务，实现多人音视频通话或直播的录制，包括录音或录像。 通过云端录制功能，用户可以将语音聊天、视频通话以及直播的录音或者录像进行云端存储，以便其他人进行回看。	视频通话 互动直播 在线教育 金融双录
互动白板	支持多人在线的实时白板，任意用户的轨迹操作都会实时同步给其他用户，可实现场景包括：文档共享、资料标注、沟通涂鸦。 丰富可视化的沟通方式，提高线上沟通的便捷性。	视频会议 在线教育

	网易云信互动白板产品详细信息请查看互动白板。	
IM 即时通讯	可以通过 IM 的单聊、群聊的聊天室，实现房间内聊天、评论、弹幕、赠送礼物等功能。 可以通过 IM 进行信令交互，实现通话呼叫、房间用户数统计等功能。 网易云信 IM 即时通讯产品详细信息请查看 IM 即时通讯。	互动直播
第三方美颜	支持开发者通过采集数据回调的 API 方法，在 App 内轻松实现接入网易云信或第三方美颜、滤镜厂商的功能。 通过第三方美颜功能，用户在进行视频通话或直播的时候，可以向对方呈现良好的肌肤状态和精神面貌。	视频通话 互动直播 在线教育
第三方变声	音频链路开放原始音频数据，可以任意对接拥有变声能力的第三方厂商。	在线 K 歌 秀场直播 多人语聊房

	通过开放的接口形式，快速满足主播对于更多声音效果的需求。	
安全通（内容安全审核）	支持内置的安全通功能，低成本接入，实现多维度、全方位音视频检测，保障应用的合规性。 可用于实时语音、实时视频内容安全检测，快速协助客户实现业务合规。 详细说明请参考 安全通。	内容监控 合规 安全检测
SIP 互通	• SDK 支持 SIP 音视频呼入呼出、会控服务。 • 接入 NERTC SDK 后即可体验高质量、抗弱网、低延迟的 SIP 音视频通讯能力，与传统 SIP 智能硬件设备实现双向互通。 • 如需使用，请 联系云信商务经理。	视频会议

1.4 产品优势

本文为您介绍音视频通话 2.0 产品的优势项和核心亮点技术，帮助您快速了解音视频通话 2.0 的主要产品优势。

1.4.1 功能优势

1.4.1.1 全平台互通

- 音视频通话 2.0 支持手机端（iOS 和 Android）、桌面端（Windows、macOS 和 Linux）和 Web 端（桌面浏览器、手机浏览器 H5）、小程序等全平台，Flutter 、Electron 和 Unity 框架，提供简单易用的的客户端 SDK 和云端 API，保障全平台互通。
- 5000 余款终端机型适配，兼容目前主流的开发架构。

1.4.1.2 超低时延

- 音视频通话 2.0 通过网易云信自研实时传输网络，网络节点覆盖全球。
- 采用最优寻址算法辅以全局实时调度能力，保证端到端平均时延 ＜200 ms。

1.4.1.3 超低卡顿

- 网易云信拥有领先行业的精准带宽侦测、智能拥塞控制、前向纠错、编码优化等技术，实测抗丢包率可达 80%。
- 适应复杂多变的网络环境，在丢包严重的弱网环境仍可正常通话，确保通话体验流畅稳定。

1.4.1.4 超高品质

- 音视频通话 2.0 支持 1080P 高清画质，80% 丢包率下仍可正常视频。
- 音频方面支持 48 kHz 采样率，128 kbps 码率，80% 丢包率可正常语音。
- 拥有行业一流的音频 3A 算法处理，即回声消除 AEC、自动噪声抑制 ANS、自动增益控制 AGC，支持纯人声模式下的啸叫检测，帮助用户消除通话中的回声和啸叫，为用户提供最纯净的通话体验。
- 自研的 AI 音频降噪算法，针对嘈杂人声、键盘声等非稳态噪声进行定向降噪，提升对于环境稳态噪声的抑制，保留更纯粹的人声。

1.4.1.5 易用接口

- 音视频通话 2.0 拥有极简的接口设计，三行代码即可轻松接入。
- 功能间对应清晰的接口，减少开发者的理解成本，帮助开发者快速接入，实现完整的业务场景。

1.4.2 亮点技术

1.4.2.1 分层编码（SVC）

音视频通话 2.0 支持 H.264 SVC 部分的时域分层编码，通过改变帧率来实现伸缩性。通信过程中，各端可以根据需要生成多种层级的视频码流，或者提取出不同层级的码流。在网络丢包较为严重的环境下，接收端抛弃部分时域层的码流以实现网络适应性，从而保证弱网环境下的视频质量和用户体验。

1.4.2.2 视频全闭环控制（VQC）

音视频通话 2.0 支持通过对视频编码参数的动态配置以及编解码器的动态选择，对通话过程中的视频质量进行质量控制，在同等码率下可以输出更加优质的视频图像，带给用户面对面的沟通体验。

1.4.2.3 视频超分（SR）

音视频通话 2.0 支持视频超分策略，可以改善因网络带宽限制或实时性的要求导致视频分辨偏低的问题，通过信号的抽取或者插入，实现低分辨率视频在传输后进行细节补充的效果，以优化接收端的视频清晰度，全方位提升用户体验。

1.4.2.4 音频 AI 降噪

音视频通话 2.0 支持自研的音频 AI 降噪算法，可以针对嘈杂人声、键盘声等非稳态噪声进行定向降噪，提升对于环境稳态噪声的抑制能力，保留更纯粹的人声。

1.4.2.5 视频降噪

音视频通话 2.0 支持自研视频降噪功能。通过视频前处理技术，消除低照度或暗背景较多的场景下的画面噪点，提升暗光场景下视频画面的清晰度以及舒适度，优化用户体验。

1.4.2.6 软件编码 NEVC 协议

音视频通话 2.0 支持桌面端的软件编解码、支持 NE264，NEVC 的能力协商等功能。相较 H264，软件编码 NEVC 协议提高了 30% 的编码压缩效率，同等码率下提升视频整体清晰度、提高鲁棒性和错误恢复能力。

音视频通话 2.0 媒体服务器全链路支持 NEVC，包括视频流的媒体转发、互动直播、录制。

二、 体验 Demo

通过体验 Demo，您可以快速了解音视频通话 2.0 产品的基本能力，并直观感受到基于产品能力可以实现的多种场景。

您也可以通过跑通 Sample Code，体验网易云信音视频通话功能，包括一对一音视频通话和多人音视频通话。

标准化产品方案 Demo 可体现基本产品能力，帮助您快速体验一对一通话或多人通话的基本功能。

- 一对一音视频通话 Demo
- 多人音视频通话 Demo

场景方案 Demo 可体现基本场景模型，帮助您直观体验语聊房、在线 KTV、PK 直播等多种场景方案的产品能力。

- 语聊房 Demo
- 多人视频连麦 Demo
- 在线 KTV Demo
- PK 直播 Demo
- 视频会议 Demo
- 一对一互动教学 Demo
- 一对多互动小班课 Demo
- 互动直播大班课 Demo

另外，音视频通话 2.0 为您提供 Web 端体验页面，供您快速了解体验产品的基本音视频通话能力，包括单人通话和多人通话，同时提供音视频质量采集、通话统计、旁路推流、伴音、屏幕共享等多种进阶功能，单击 Web 体验页立即体验。

三、 下载 SDK 和 示例代码

3.1.1 SDK 下载

NERTC 的 SDK 包 按功能可分为音视频 SDK 和纯音频 SDK，您可以根据需要下载对应的 SDK 包。

- 音视频 SDK：包含音视频完整能力，功能更丰富。
- 纯音频 SDK：只包含音频通话能力，SDK 包的体积会更小。

根据不同行业、不同场景的不同业务需求，NERTC 提供不同版本的 SDK 包，请在官网首页通过 QQ、在线消息或电话等方式联系云信商务经理，为您推荐针对您的场景最合适、最优的版本。

各平台最新稳定版 SDK 包的下载地址：SDK 下载中心。

3.1.2 示例代码

项目	说明	下载地址
Web 体验页	Web 端的快速体验页面，提供简易的示例项目，帮助您快速体验 SDK 的能力，了解如何使用 NERTC SDK 的基本功能，例如美颜、屏幕共享、旁路推流、音	下载 Web 示例项目

	视频质量管理、伴音、自定义视频采集等功能。	
基础示例项目	基础类示例项目的源码，帮助您快速集成NERTC SDK，实现一对一视频通话、多人视频通话功能。	下载基础示例项目
进阶示例项目	进阶类示例项目的源码，帮助您快速集成NERTC SDK，实现屏幕共享、旁路推流、音视频质量管理、伴音、自定义视频采集等功能。	下载进阶示例项目

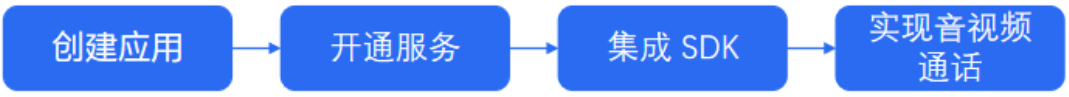
四、 快速开始

4.1 接入流程

网易云信音视频通话 2.0 为您提供搭建音视频通话的全套 SDK，帮助您快速实现基于基础音视频通话的音视频通话场景。

音视频通话的快速入门主要为您展示音视频通话 2.0 的基本接入流程，包括开通服务与创建应用的操作步骤、集成 SDK 的操作步骤，以及实现音视频通话基本功能的操作流程。您可以根据需要查看指定任务对应的各端文档。

4.2 接入流程



步骤	操作	描述
1	创建应用并获取 App Key	在网易云信控制台中创建应用，查看该应用的 App Key。
2	开通服务	在网易云信控制台中为已创建的应用开通服务。

3	集成 SDK	通过网易云信音视频通话 2.0 产品提供的 NERTC Android SDK，快速集成客户端 SDK。
4	实现音视频通话基本功能	基于音视频通话 SDK 快速实现音视频通话的基本功能。快速入门以典型业务场景为例，为您介绍基于该业务流程的 SDK 实现步骤。

创建应用并开通音视频通话服务后，应用默认的鉴权方式为安全模式。网易云信建议您在应用接入和测试期间使用调试模式，并在正式上线前改为安全模式。如果在控制台中为指定应用开启了安全模式，则对应 App 用户在加入房间时，需要通过 Token 进行身份校验。具体鉴权方式请参考 Token 鉴权。

4.3 创建应用

创建应用是体验或使用网易云信各款产品和服务的首要前提，您可以参考本文档在网易云信控制台创建一个应用，并查看该应用的 App Key。

应用是网易云信各项服务的基础业务单元，每个应用有惟一的 App Key。同一套账号体系创建一个应用即可，例如买家和卖家属于同一账号体系，只需创建一个应用。

4.3.1 注意事项

- 应用创建后不能删除，请根据您的实际需求创建应用。

- 接入网易云信的过程中，调用服务端 API 和集成客户端 SDK，都需要使用到应用对应的 AppKey。请妥善保管。
- 创建应用之后，如果在控制台单击**刷新秘钥**，刷新了 App Secret，开发者的代码中也要相应更换为新的 App Secret。

4.3.2 创建应用

1. 登录网易云信控制台。
2. 在左侧导航栏中找到**应用**，并单击**创建**。

   如果您是首次创建应用，请单击**创建第一个应用**。

3. 填写应用的基本信息。

配置	说明

应用名称	应用的名称，支持中文、英文、数字和特殊字符，长度为1~20 个字符。
行业类型	该应用的对应行业。
简介	应用的备注说明或简单介绍，可选项，便于您标识该应用。

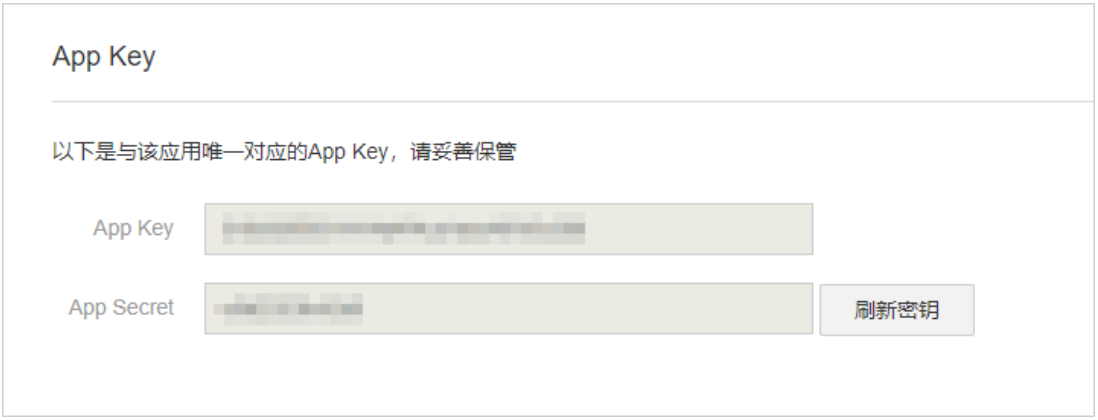
4.

5. 单击**创建**。

创建应用后，您可以在左侧导航栏中查看该应用。

4.3.3 获取 App Key

1. 在左侧导航栏中找到该应用，并单击应用名称。
2. 单击 **App Key 管理**。
3. 查看该应用的 App Key。



4.4 开通服务

若您之前从未开通过任何网易云信的产品服务，请先在官网首页通过 QQ、在线消息或电话等方式联系商务经理，一对一沟通您的具体需求。

1. 登录网易云信控制台。
2. 在左侧导航栏中单击指定应用名称，进入该应用的详情页面。
3. 在**未开通功能**中找到**音视频通话 2.0**。
4. 在对应的位置单击**立即开通**。

5. 设置**充值金额**。

   具体的计费策略，请根据页面提示咨询商务经理。

6. 在页面右侧勾选**我已阅读并同意《网易云信服务条款》**，并单击**提交订单**。

7. 在**支付台**中确认金额、选择支付方式，并单击**立即支付**。

4.5 快速跑通 Sample Code

4.5.1 Android

您可以通过跑通 Sample Code，体验网易云信音视频通话功能，包括一对一音视频通话和多人音视频通话。

4.5.1.1 前提条件

请确认您已完成以下操作：

- 创建应用并获取 App Key。
- 开通音视频通话 2.0 服务。
- 集成 SDK（Android），其中需要**添加必要的设备权限**。

4.5.1.2 快速跑通 Sample Code

在运行示例项目之前，请在云信控制台中为指定应用开通调试模式。调试模式建议只在集成开发阶段使用，请在应用正式上线前改回安全模式。

1. 在一对一通话示例代码或多人通话示例代码下载页面下载需要的 Demo 源码工程。
2. **创建项目**及**添加依赖**等步骤具体请参考集成 SDK。
3. 在 app/src/main/res/values/strings.xml 文件中配置 AppKey。

```
<!-- 替换为你自己的 AppKey -->

<string name="app_key">YOUR APP KEY</string>
```

4. 添加必要的设备权限。
5. 运行工程。

4.5.2 iOS

您可以通过跑通 Sample Code，体验网易云信音视频通话功能，包括一对一音视频通话和多人音视频通话。

4.5.2.1 前提条件

请确认您已完成以下操作：

- 创建应用并获取 App Key。
- 开通音视频通话 2.0 服务。
- 集成 SDK（iOS），其中需要**设置签名**并**添加媒体设备权限**。

4.5.2.2 快速跑通 Sample Code

在运行示例项目之前，请在云信控制台中为指定应用开通调试模式。调试模式建议只在集成开发阶段使用，请在应用正式上线前改回安全模式。

1. 在一对一通话示例代码或多人通话示例代码下载页面下载需要的 Demo 源码工程。

   以 Objective-C 工程源码为例。Podfile 文件中包括以下内容：

```
# platform :ios, '9.0'



target 'NERTC1to1Sample' do

  # Comment the next line if you don't want to use dynamic frameworks

  use_frameworks!

  pod 'NERtcSDK', 'x.x.x'

end
```


2. **创建项目**等步骤具体请参考集成 SDK。

3. 执行安装。

```
pod install
```

4. 双击 NERTC1to1Sample.xcworkspace，通过 Xcode 打开工程。
5. 在 AppKey.h 文件中填入您的 AppKey，并注释 NETSAppDelegate.m 中的 NSAssert 语句。
6. 设置签名并添加媒体设备权限。
7. 运行工程。

4.6 集成 SDK

4.6.1 Android

本文为您介绍了 Android 端集成 SDK 的操作步骤，帮助您快速集成 SDK 并实现实时音视频通话的基本功能。

4.6.1.1 前提条件

在开始运行工程之前，请您准备以下开发环境：

- Android SDK API 等级 19 或以上。
- Android Studio 3.0 或以上版本。
- Android 系统 4.4 或以上版本的移动设备。

4.6.1.2 SDK 目录

- 目录结构

```
Android
```

```
|
|-- lib                          //推荐放置在 /app/libs/ 目录
|   |
|   |-- nertc-sdk-4.x.xx.jar
|
|-- partlib                      //推荐放置在 /app/libs/ 目录
|   |
|   |-- nertc-sdk-part-4.x.xx.jar
|   |-- yunxin-catcher-1.0.2.jar
|   |-- yunxin-nos-1.0.3.jar
|   |-- yunxin-report-2.1.0.jar
|   |-- GrowDevice-1.7.2.4.jar
|
|-- jniLibs                      //推荐放置在 /app/src/main/jniLibs/ 目录
|   |
|   |-- armeabi-v7a
```

```
|       |
|       |   |-- libnertc_sdk.so
|       |   |-- libgrowease.so
|   |
|   |-- arm64-v8a
|       |
|       |   |-- libnertc_sdk.so
|       |   |-- libgrowease.so
|   |
|   |-- x86
|       |
|       |   |-- libnertc_sdk.so
|       |   |-- libgrowease.so
```

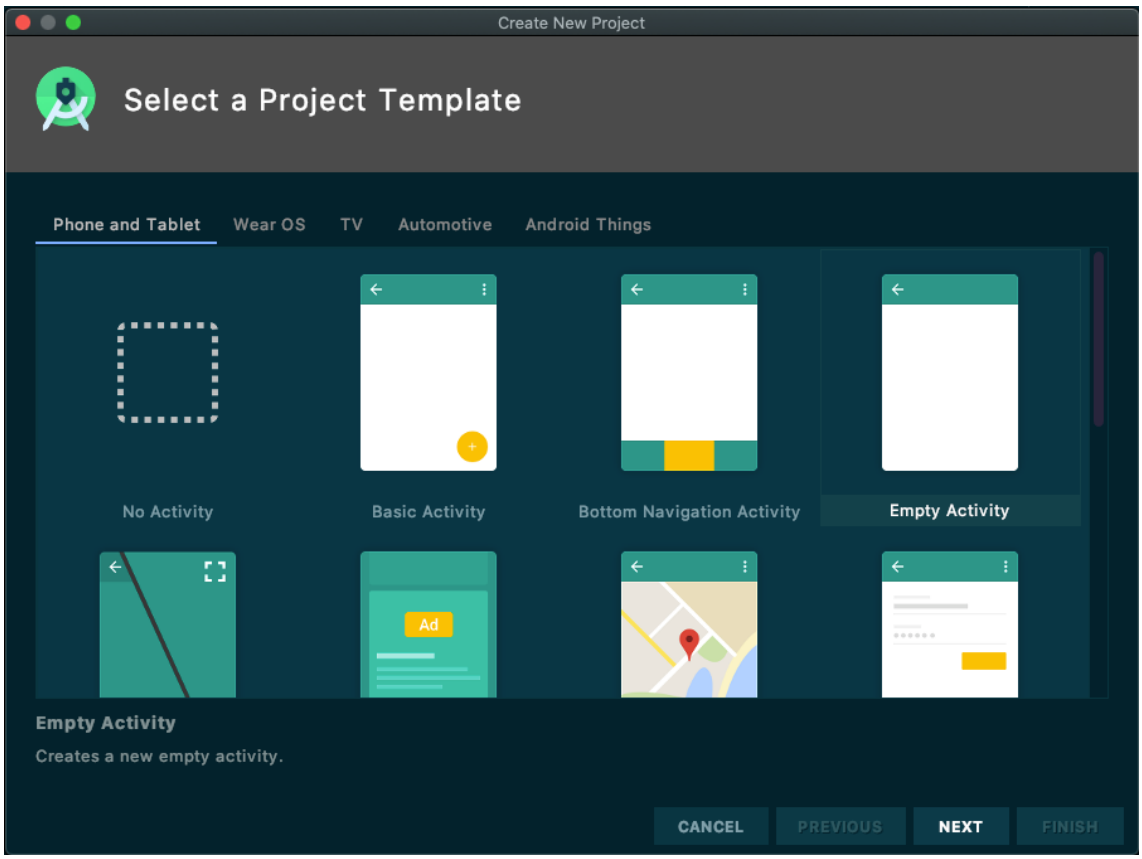

- 目录内容

文件/文件夹名称	是否必选	说明
lib-nertc-sdk	是	音视频库。
libnertc_sdk.so	是	
partlib-nertc-sdk-part	是	G1 兼容库。
partlib-yunxin-catcher	是	崩溃捕获。
partlib-yunxin-nos	是	日志上报。
partlib-yunxin-report	是	
partlib-GrowDevice	是	智企库。
libgrowease.so	是	

4.6.1.3 集成 SDK

Maven 集成（推荐）

1. 若您需要创建新项目，在 Android Studio 里，在顶部菜单依次选择 File > New > New Project 新建工程，再依次选择 Phone and Tablet > Empty Activity，单击 Next。



创建 Android 项目成功后，Android Studio 会自动开始同步 gradle，您需要等同步成功后再进行下一步操作。

2. 在项目对应模块的 build.gradle 中加入以下行。

```
api 'com.netease.yunxin:nertc-full:.x.x'
```

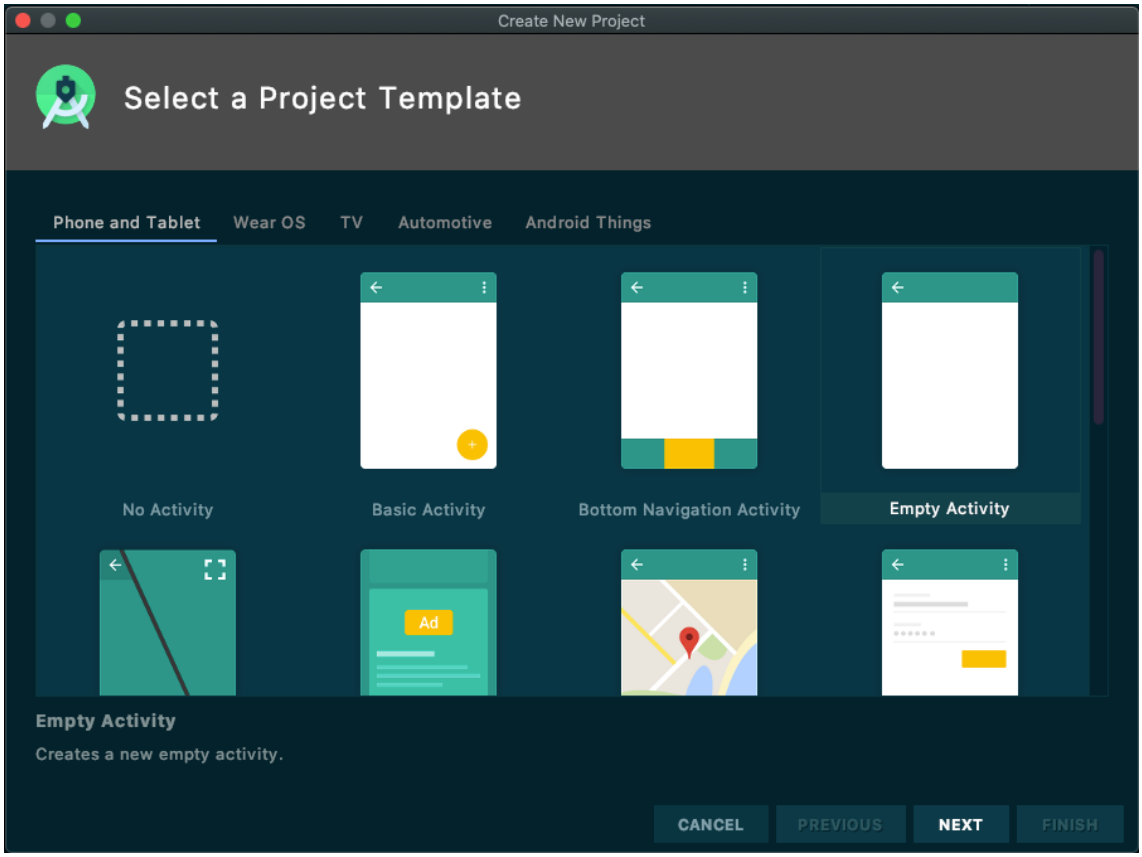
其中，x.x.x 为 NERTC SDK 对应版本的版本号。建议使用最新版本，您可以在网易云信 SDK 下载中心查看 NERTC SDK 最新版本的版本号。若要使用其他版本，请联系网易云信技术支持获取对应的版本号。

若您集成的 NERTC SDK 为 V4.6.10 之前的版本，请参考如下代码实现。

```
api 'com.netease.yunxin:nertc:x.x.x'
```

手动集成

1. 若您需要创建新项目，在 Android Studio 里，在顶部菜单依次选择 File > New > New Project 新建工程，再依次选择 Phone and Tablet > Empty Activity，单击 Next。



创建 Android 项目成功后，Android Studio 会自动开始同步 gradle，您需要等同步成功后再进行下一步操作。

2. 前往 SDK 下载页面获取当前最新版本 SDK，或联系网易云信技术支持获取对应版本的 SDK。
3. 解压后将对应的文件拷贝至项目路径中。

   若无对应文件夹，您需要在对应路径下新建文件夹。

   无特殊情况，可忽略 part 文件夹。

文件/文件夹	项目路径
nertc-sdk-x.x.x.jar	/app/libs/
arm64-v8a arm64-v7a x86	/app/src/main/jniLibs/

4. 在 app/build.gradle 文件中设置 libs 路径。

```
android {
        ...
    compileOptions {
        // SDK 依赖的 JDK 版本为 Java 8
```

```
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    ...

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


5. 单击 File > Sync Project With Gradle Files 按钮，直到同步完成。

4.6.1.4 添加权限

打开 app/src/main/AndroidManifest.xml 文件，添加必要的设备权限。

例如：

```
//网络相关
```

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

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

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

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

//防止通话过程中锁屏

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

//视频权限

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

//录音权限

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

//修改音频设置

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

//蓝牙权限
```

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

//蓝牙连接权限，此权限还需在运行应用时动态申请，否则 Android 12 及以上的设备蓝牙无法正常工作

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

//外置存储卡写入权限

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

//蓝牙 startBluetoothSco 会用到此权限

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

//获取设备信息

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

//允许应用程序使用 camera 硬件资源

<uses-feature android:name="android.hardware.camera"/>

//自动对焦
```

```
<uses-feature android:name="android.hardware.camera.autofocus"/>

.....//APP 需要的其他设备权限
```


4.6.1.5 防止代码混淆

在 proguard-rules.pro 文件中，为 NERTC SDK 添加 -keep 类的配置，可以防止混淆 NERTC SDK 公共类名称。

```
-keep class com.netease.lava.** {*;}

-keep class com.netease.yunxin.** {*;}
```


4.6.1.6 后续步骤

实现音视频通话

4.6.2 iOS

本文为您介绍了 iOS 端集成 SDK 的操作步骤，帮助您快速集成 SDK 并实现实时音视频通话的基本功能。

4.6.2.1 前提条件

在开始运行工程之前，请您准备以下开发环境：

- Xcode 10 及以上版本。
- iOS 9.0 及以上版本的 iOS 设备。

4.6.2.2 SDK 目录

- 目录结构

```
iOS
|
|-- NERtcSDK.framework                          //推荐放置在 /app/libs/ 目录
|-- NMCBasicModuleFramework.framework           //推荐放置在 /app/libs/ 目录
```

- 目录内容

文件/文件夹名称	是否必选	说明
NERtcSDK.framework	是	音视频库。

文件/文件夹名称	是否必选	说明
NMCBasicModuleFramework.framework	是	基础模块库。

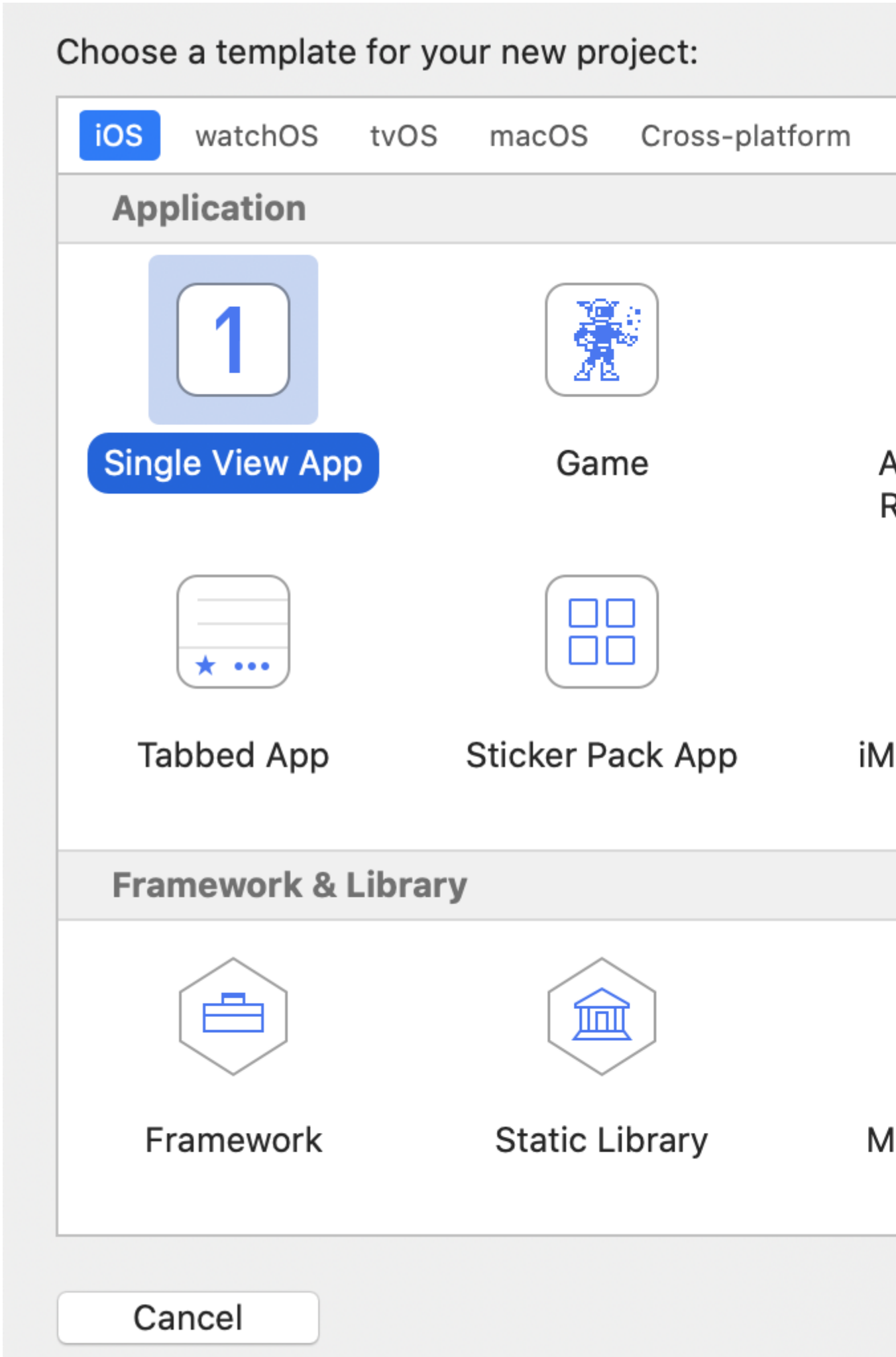
4.6.2.3 集成 NERTC SDK

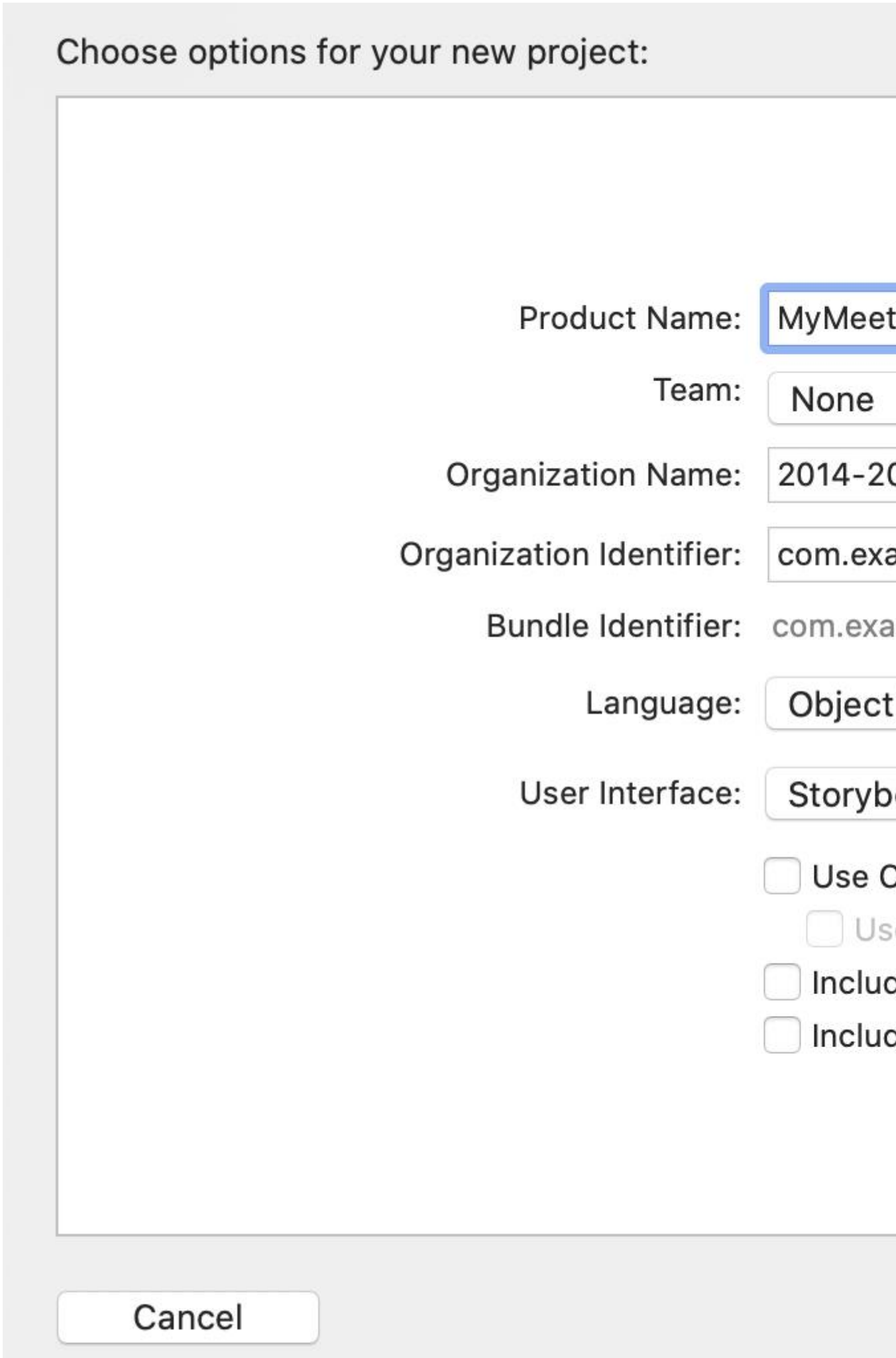
方法一 CocoaPods 集成

请确保您的 Mac 已经安装 Ruby 环境。

1. 创建项目。

   若您需要创建新项目，在 XCode 里，依次选择 **Create a New XCode Project** > **Single View App** > **Next** 新建工程，再配置工程相关信息和合适的工程本地路径，单击 **Create**。


Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Application
Single View App
Game
Tabbed App
Sticker Pack App
Framework & Library
Framework
Static Library
Cancel


Choose options for your new project:
Product Name:
MyMeet
Team:
None
Organization Name:
2014-20
Organization Identifier:
com.exa
Bundle Identifier:
com.exa
Language:
Object
User Interface:
Storyb
Use C
Us
Includ
Includ
Cancel

2. 安装 CocoaPods。

   在终端窗口中输入如下命令：

```
sudo gem install cocoapods
```


3. 创建 Podfile 文件。

   从 Terminal 中进入您所创建项目所在路径，运行以下命令创建 Podfile 文件。

```
pod init
```


4. 编辑 Podfile 文件。

```
# platform :ios, '9.0'

target '{YourApp}' do

    pod 'NERtcSDK', '~> {version}'

end
```


- YourApp：您的 Target 名称。

- version：待集成的 NERTC SDK 版本号。建议使用最新版本，请从 SDK 下载中心查看 NERTC SDK 最新版本的版本号。若要使用其他版本，请联系网易云信技术支持获取对应的版本号。

9. 执行以下命令查询本地库版本。

```
pod search NERtcSDK
```


11. 若不是最新版本，可以执行以下命令更新本地库版本。

```
pod repo update
```


13. 执行以下命令安装安装 SDK。

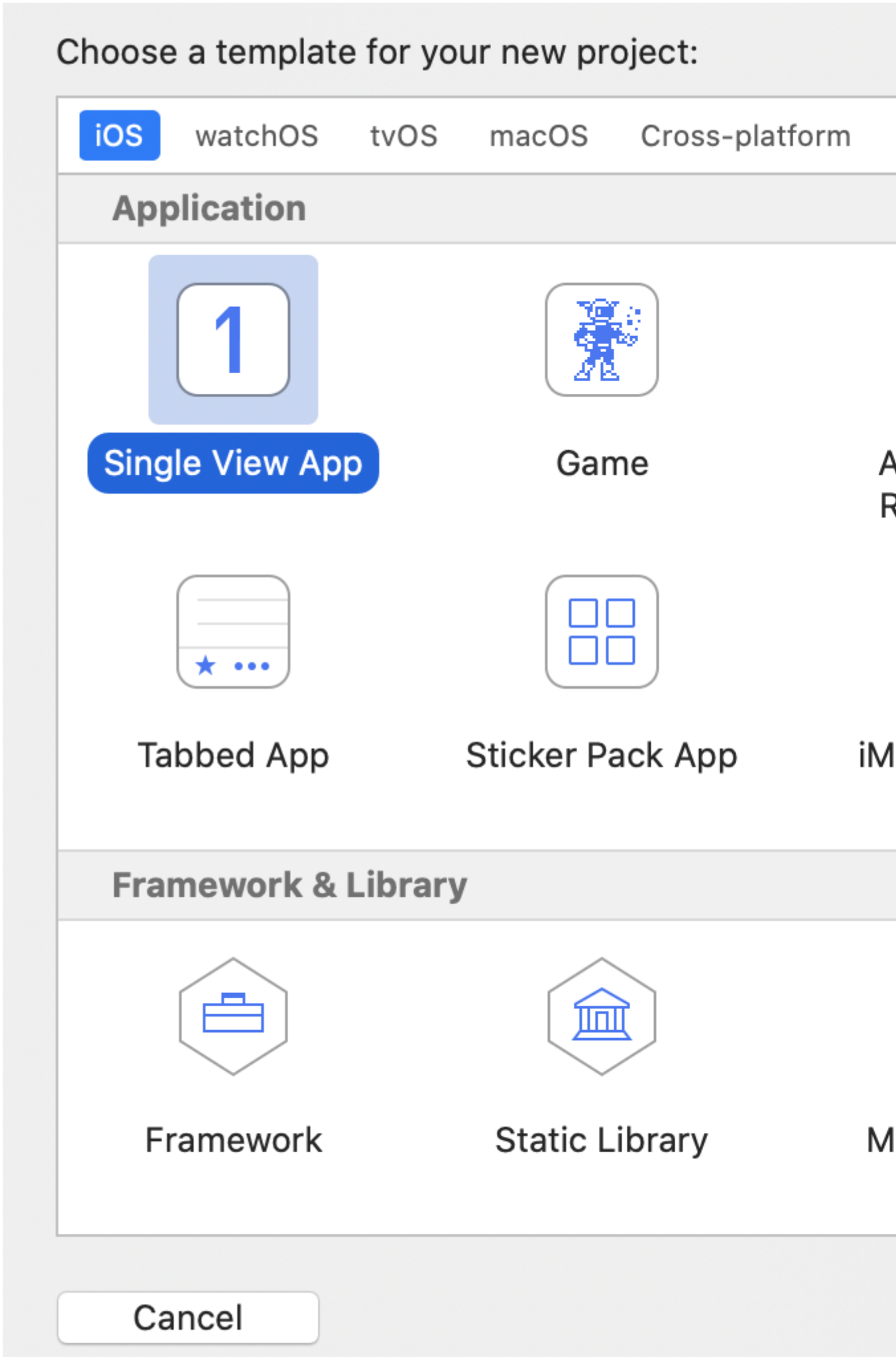
```
pod install
```

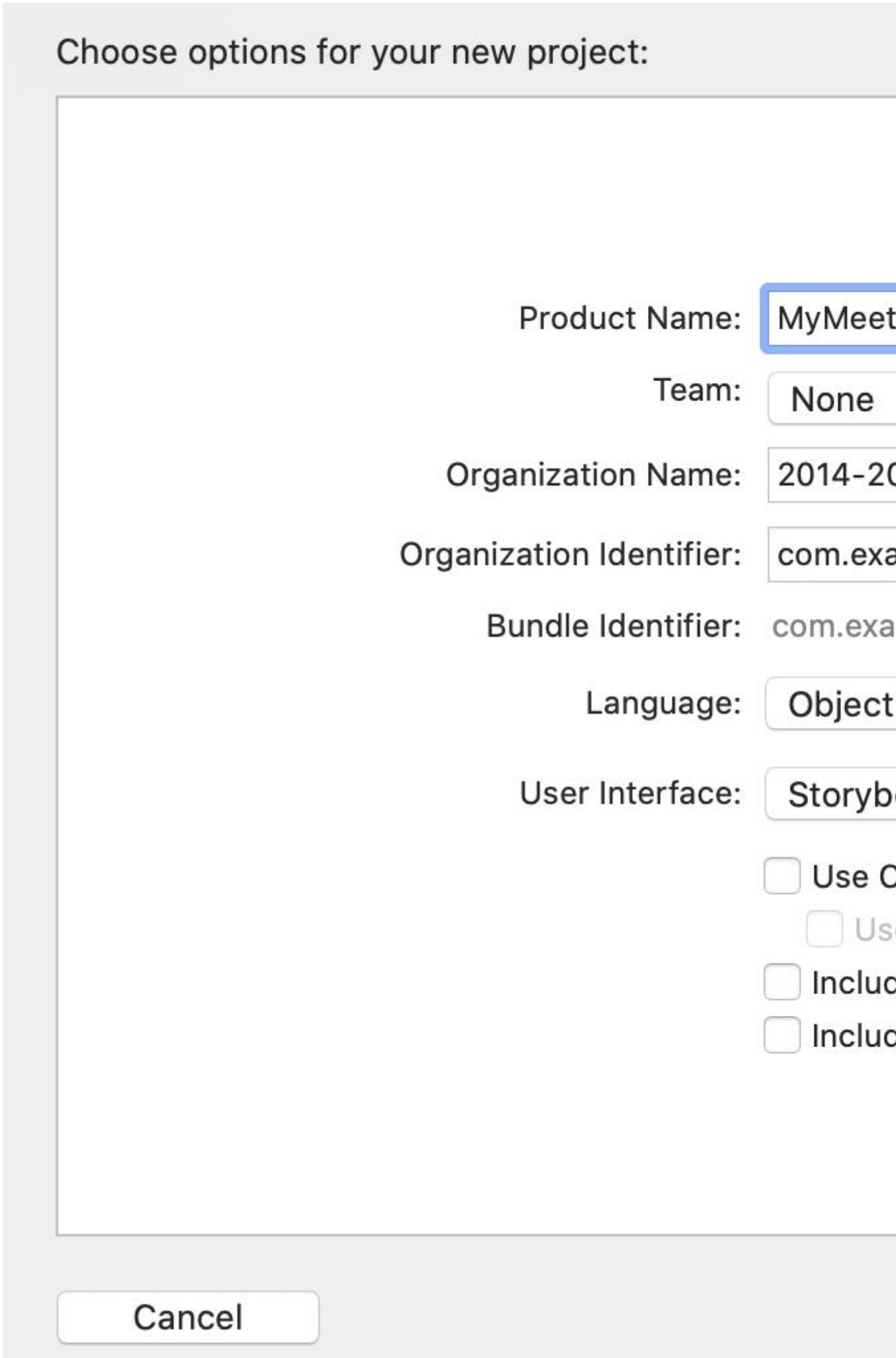

15. 设置签名和添加设备权限，具体请参见设置签名并添加媒体设备权限。

方法二 手动集成

1. 创建项目。

   若您需要创建新项目，在 XCode 里，依次选择 Create a New XCode Project > Single View App > Next 新建工程，再配置工程相关信息和合适的工程本地路径，单击 Create。


Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Application
Single View App
Game
Tabbed App
Sticker Pack App
Framework & Library
Framework
Static Library
Cancel


Choose options for your new project:
Product Name:
MyMeet
Team:
None
Organization Name:
2014-20
Organization Identifier:
com.exa
Bundle Identifier:
com.exa
Language:
Object
User Interface:
Storyb
Use C
Us
Includ
Includ
Cancel

2. 前往 网易云信 SDK 下载中心获取当前最新版本，或联系网易云信技术支持获取对应版本的 SDK 安装包。
3. 将解压之后的 NMCBasicModuleFramework.framework 和 NERtcSDK.framework 文件加入到工程路径下。
4. 以 Xcode Version 11.5 为例，进入 **TARGETS > Project Name > General > Frameworks, Libraries, and Embedded Content** 菜单，点击 +，再点击 **Add Other…**，将上述解压得到的 SDK 文件添加进去。
5. 将 **Embed** 属性设置为 **Embed & Sign**，以使得 SDK 动态库和应用签名保持一致。
6. 设置签名和添加设备权限，具体请参见设置签名并添加媒体设备权限。

在手动导入 SDK 的情况下，由于 SDK 包含模拟器版本，会导致打包失败。所以需要在打包之前将模拟器版本剥去。

1. 在工程里创建 nim_strip_archs.sh 脚本到指定目录，例如 Supporting Files 目录。
2. 在 Build Phases 中增加过程，类型为 New Run Script Phase。需要把去掉模拟器的 Run Script 脚本放在 Embed Frameworks 之后。
3. 在工程里添加内容: /bin/sh 您的脚本路径，例如 /bin/sh "${SRCROOT}/NIMDemo/Supporting Files/nim_strip_archs.sh"。
4. 将如下内容复制到脚本：

```
#!/bin/sh

# Strip invalid architectures
```

```
8.
9.  strip_invalid_archs() {
10.binary="$1"
11. echo "current binary ${binary}"
12.# Get architectures for current file
13.archs="$(lipo -info "$binary" | rev | cut -d ':' -f1 | rev)"
14.stripped=""
15.for arch in $archs; do
16.if ! [[ "${ARCHS}" == *"$arch"* ]]; then
17.if [ -f "$binary" ]; then
18.# Strip non-valid architectures in-place
19.lipo -remove "$arch" -output "$binary" "$binary" || exit 1
20.            stripped="$stripped $arch"
21.fi
22.            fi
23.            done
```

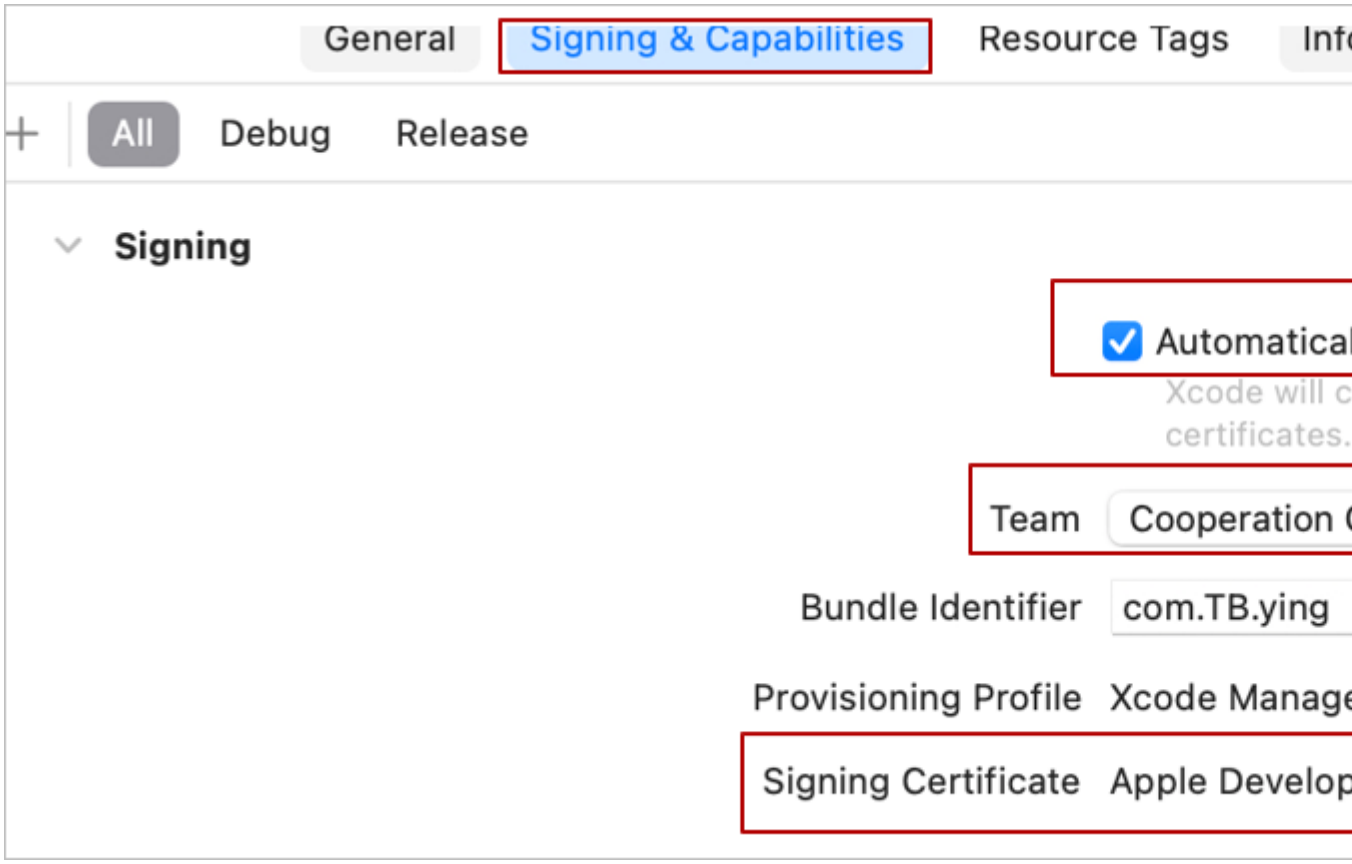
```
24.         if [[ "$stripped" ]]; then
25.         echo "Stripped $binary of architectures:$stripped"
26.         fi
27.         }
28.
29.         APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME
    }"
30.
31.# This script loops through the frameworks embedded in the
    application and
32.         # removes unused architectures.
33.         find "$APP_PATH" -name '*.framework' -type d | while read -
    r FRAMEWORK
34.         do
35.         FRAMEWORK_EXECUTABLE_NAME=$(defaults read
    "$FRAMEWORK/Info.plist" CFBundleExecutable)
```

```
36.         FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$F
    RAMEWORK_EXECUTABLE_NAME"
37.         echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"
38.
39.         strip_invalid_archs "$FRAMEWORK_EXECUTABLE_PATH"
40.         done
```


4.6.2.4 设置签名并添加媒体设备权限

1. 设置签名。

i.在 Xcode 中，选择目标 TARGETS 和 Project Name，单击 **Signing & Capabilities** 页签，选中 **Automatically manage signing**。



ii. 在 Team 中选择您的开发团队。

2. 在 Signing & Capabilities 页面，打开后台音频权限。

   打开后台音频权限之后，应用在手机中后台运行时，SDK 默认在后台也可以继续处理音频流，维持通话。

3. 编辑 info.plist 文件，授权麦克风、摄像头和 WiFi 的使用权限。

   使用 SDK 的音视频功能，需要授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下三项。

- Privacy – Microphone Usage Description，并填入麦克风使用目的的提示语。
- Privacy – Camera Usage Description，并填入摄像头使用目的的提示语。
- Application uses Wi-Fi，设置为 YES。

4.6.2.5 后续步骤

实现音视频通话

4.7 实现音视频通话

4.7.1 Android

网易云信音视频通话产品的基本功能包括高质量的实时音视频通话。当您成功初始化 SDK 之后，您可以简单体验本产品的基本业务流程。本文档为您展示音视频通话提供的基本业务流程。

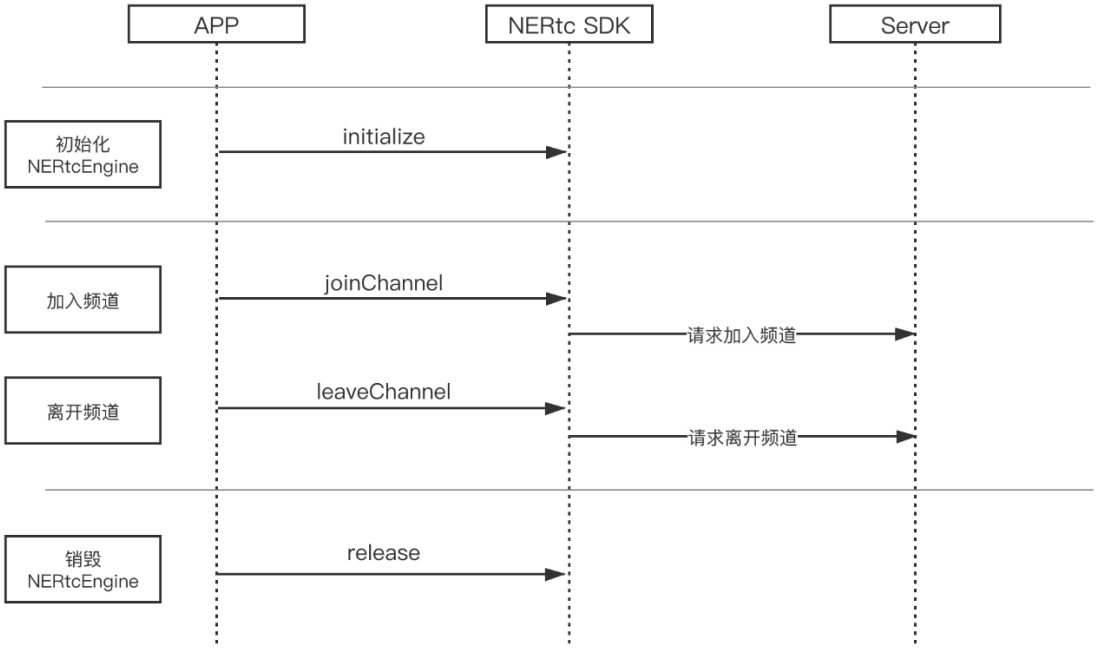
4.7.1.1 前提条件

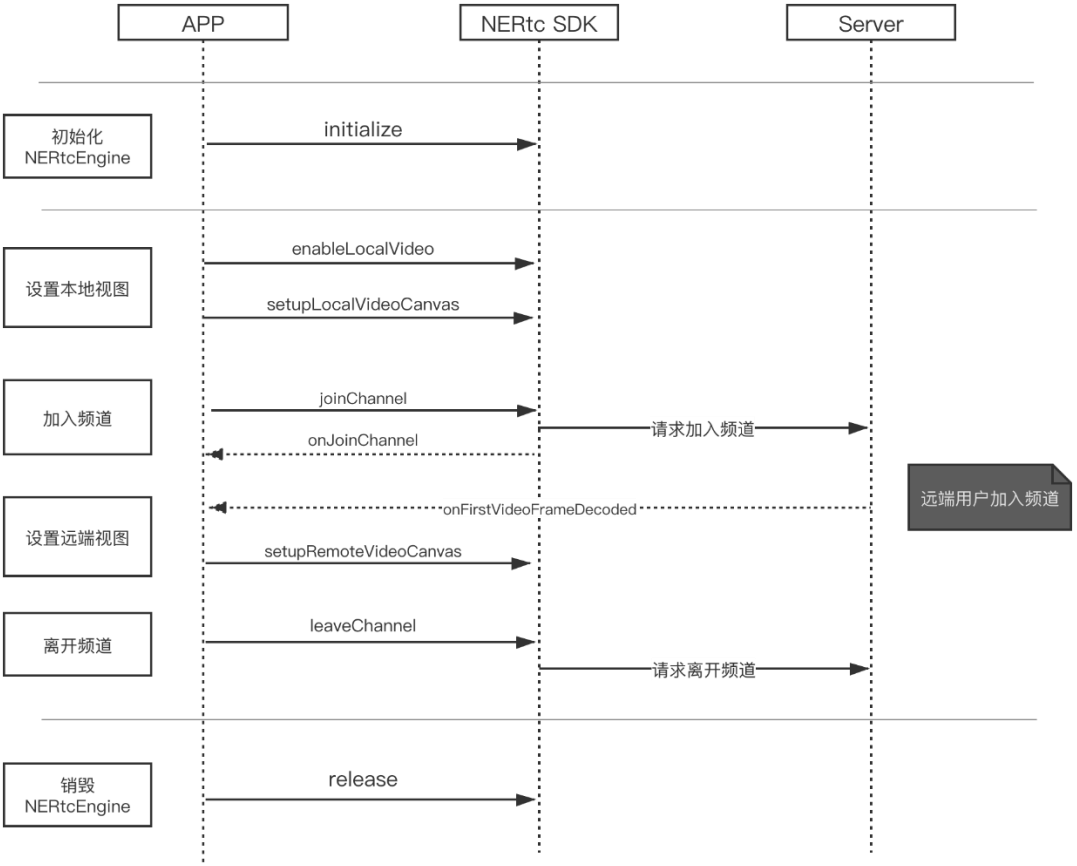
请确认您已完成以下操作：

- 已创建应用并开通服务、获取 App Key。
- 集成 SDK（Android）。

4.7.1.2 实现音视频通话

本节主要介绍如何使用 NERTC SDK 实现音视频通话，主要流程如下图所示：


APP
NERtc SDK
Server
初始化
NERtcEngine
initialize
加入频道
joinChannel
请求加入频道
离开频道
leaveChannel
请求离开频道
销毁
NERtcEngine
release


APP
NERtc SDK
Server
初始化
NERtcEngine
initialize
设置本地视图
enableLocalVideo
setupLocalVideoCanvas
加入频道
joinChannel
请求加入频道
onJoinChannel
远端用户加入频道
设置远端视图
onFirstVideoFrameDecoded
setupRemoteVideoCanvas
离开频道
leaveChannel
请求离开频道
销毁
NERtcEngine
release

1. 导入类

在您的工程中对应的 Activity 文件里添加如下代码先导入以下重要类：

```
import com.netease.lava.nertc.sdk.NERtcCallbackEx;

import com.netease.lava.nertc.sdk.NERtcConstants;

import com.netease.lava.nertc.sdk.NERtcEx;

import com.netease.lava.nertc.sdk.NERtcParameters;

import com.netease.lava.nertc.sdk.video.NERtcRemoteVideoStreamType;

import com.netease.lava.nertc.sdk.video.NERtcVideoView;
```

2. 初始化

默认情况下，请先执行 init 方法完成初始化。

```
// 示例

private void initializeSDK() {

        try {


NERtcEx.getInstance().init(getApplicationContext(),Config.APP_KEY,callback

,null);
```

```
        } catch (Exception e) {
            showToast("SDK 初始化失败");
            finish();
            ...
            return;
        }
        ...
}
```

3. 设置本地视图

初始化成功后，可以设置本地视图，来预览本地图像。您可以在加入房间之前预览，或在加入房间后预览。

- 加入房间前预览。

1. 通过 startVideoPreview 与 setupLocalVideoCanvas 方法，在加入房间前设置本地视图，预览本地图像。

```
// 示例
NERtcEx.getInstance().startVideoPreview();
```

```
NERtcVideoView localView =
(NERtcVideoView)findViewById(R.id.local_view);

NERtcEx.getInstance().setupLocalVideoCanvas(localView);
```

2. 若要结束预览，或者准备加入房间时，调用 stopVideoPreview 停止预览。

- 加入房间后预览。

  在成功加入房间后，通过 enableLocalVideo 方法进行视频的采集发送与预览。

```
// 示例

// 开启本地视频采集并发送

NERtcEx.getInstance().enableLocalVideo(true);

// 设置本地预览画布

NERtcVideoView localView =
(NERtcVideoView)findViewById(R.id.local_view);

NERtcEx.getInstance().setupLocalVideoCanvas(localView);
```

4. 加入房间

加入房间前，请确保已完成初始化相关事项。若您的业务中涉及呼叫邀请等机制，建议通过信令实现。

通过 joinChannel 方法加入房间。

调用 joinChannel 之后，NERTC SDK 会通过 Android 的 AudioManager.setMode() 方法调整音频模式（audio mode），此后请勿修改 SDK 调整过的音频模式，否则会导致音频路由错误等问题。

```
// 示例

NERtcEx.getInstance().joinChannel(token,channelName,uid);
```

参数说明：

参数	说明
token	安全认证签名（NERTC Token）。可设置为： null。调试模式下可设置为 null。安全性不高，建议在产品正式上线前在云信控制台中将指定应用的鉴权方式恢复为默认的安全模式。 已获取的 NERTC Token。
channelName	房间名称，设置相同房间名称的用户会进入同一个通话房间。 **注意**：您也可以在加入通道前，通过创建房间接口创建房间。加入房间时，若传入的 {channelName} 未事先创建，则云信服务器内部将为其自动创建一个名为

	{channelName} 的通话房间。
uid	用户的唯一标识 id，房间内每个用户的 uid 必须是唯一的。

SDK 发起加入房间请求后，服务器会进行响应，开发者可以通过 NERtcCallback 的 onJoinChannel 回调监听加入房间的结果，同时该回调会抛出当前通话房间的 channelId 与加入房间总耗时（毫秒）。

注意：请在初始化方法中传入原型为 NERtcCallback / NERtcCallbackEx 的 callback。

5. 设置远端视图

音视频通话过程中，除了要显示本地的视频画面，通常也要显示参与互动的其他连麦者/主播的远端视频画面。

1. NERtcCallback 通过以下回调获取相关信息：

- onUserJoined：监听远端用户加入通话房间的事件，并抛出对方的 uid。当本端加入房间后，也会通过此回调抛出通话房间内已有的其他用户。
- onUserVideoStart：监听远端用户发布视频流的事件，回调中携带对方的 uid 与发布的视频分辨率。

2. 在监听到远端用户加入房间或发布视频流后，本方可以通过 setupRemoteVideoCanvas 方法设置远端用户视频画布，用于显示其视频画面。

```
// 示例
```

```
NERtcVideoView remoteView =
(NERtcVideoView)findViewById(R.id.remote_view);

NERtcEx.getInstance().setupRemoteVideoCanvas(remoteView,uid);
```

3. 在监听到远端用户发布视频流后，本方可以通过 subscribeRemoteVideoStream 方法对其发起视频流的订阅，来将对方的视频流渲染到视频画布上。

```
NERtcEx.getInstance().subscribeRemoteVideoStream(uid, profile,
subscribe);
```

4. 监听远端用户离开房间或关闭视频功能。

- onUserLeave：用户离开房间回调。
- onUserVideoStop：远端用户关闭视频功能回调。

6. 音频流

在 NERtcSDK 中，本地音频的采集发布和远端音频订阅播放是默认启动的，正常情况下无需开发者主动干预。

7. 退出通话房间

通过 leaveChannel() 接口退出通话房间。

```
// 示例

// 退出通话房间
```

```
NERtcEx.getInstance().leaveChannel();
```

NERtcCallback 提供 onLeaveChannel 来监听当前用户退出房间的结果。

8. 销毁实例

当确定 App 短期内不再使用音视频通话实例时，可以通过 release() 接口释放对应的对象资源。

```
// 示例

// 销毁实例

NERtcEx.getInstance().release();
```

4.7.2 iOS

网易云信音视频通话产品的基本功能包括高质量的实时音视频通话。当您成功初始化 SDK 之后，您可以简单体验本产品的基本业务流程。本文档为您展示音视频通话提供的基本业务流程。

4.7.2.1 前提条件

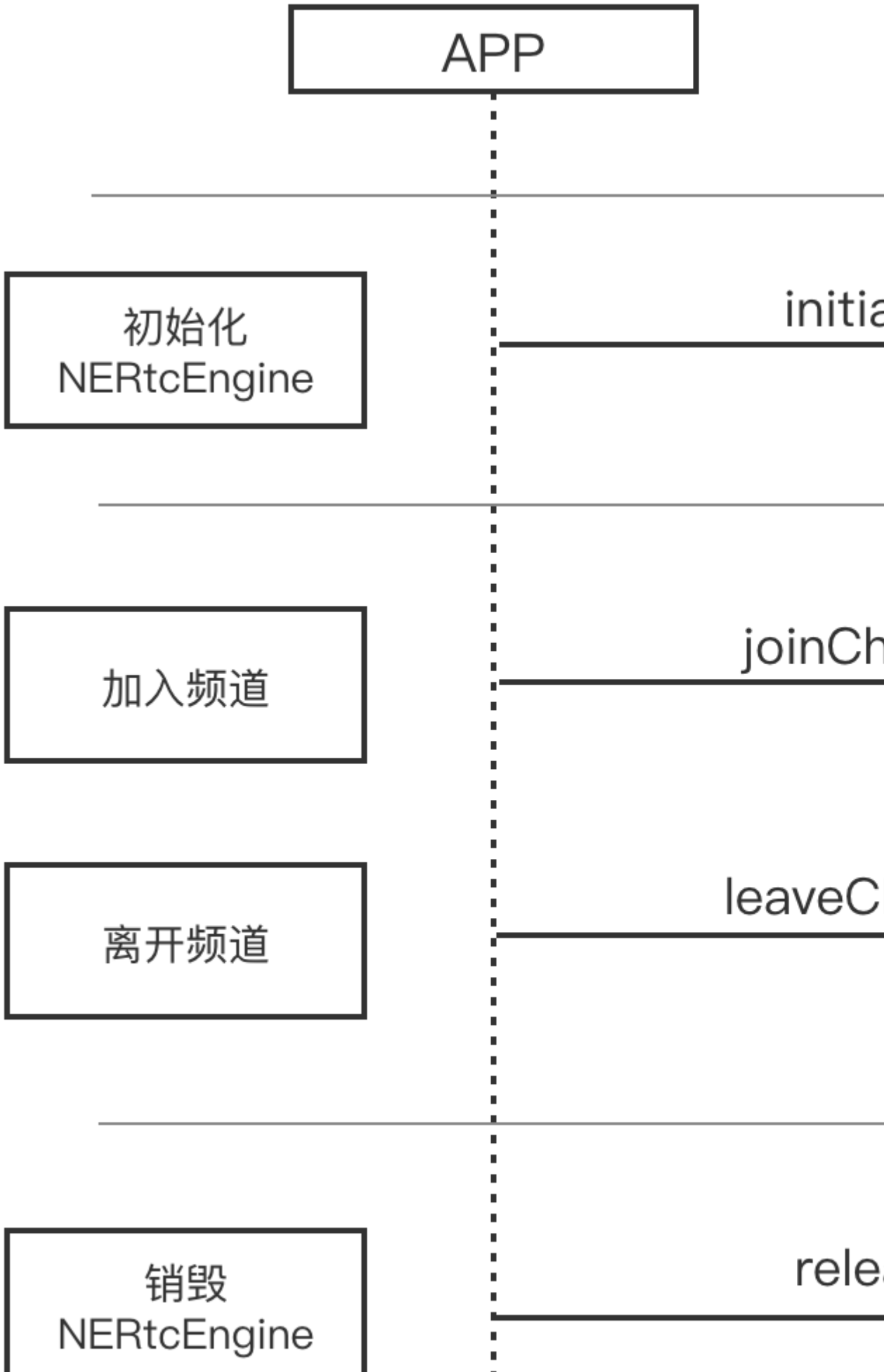
请确认您已完成以下操作：

- 创建应用并获取 App Key。
- 开通音视频通话 2.0 服务。

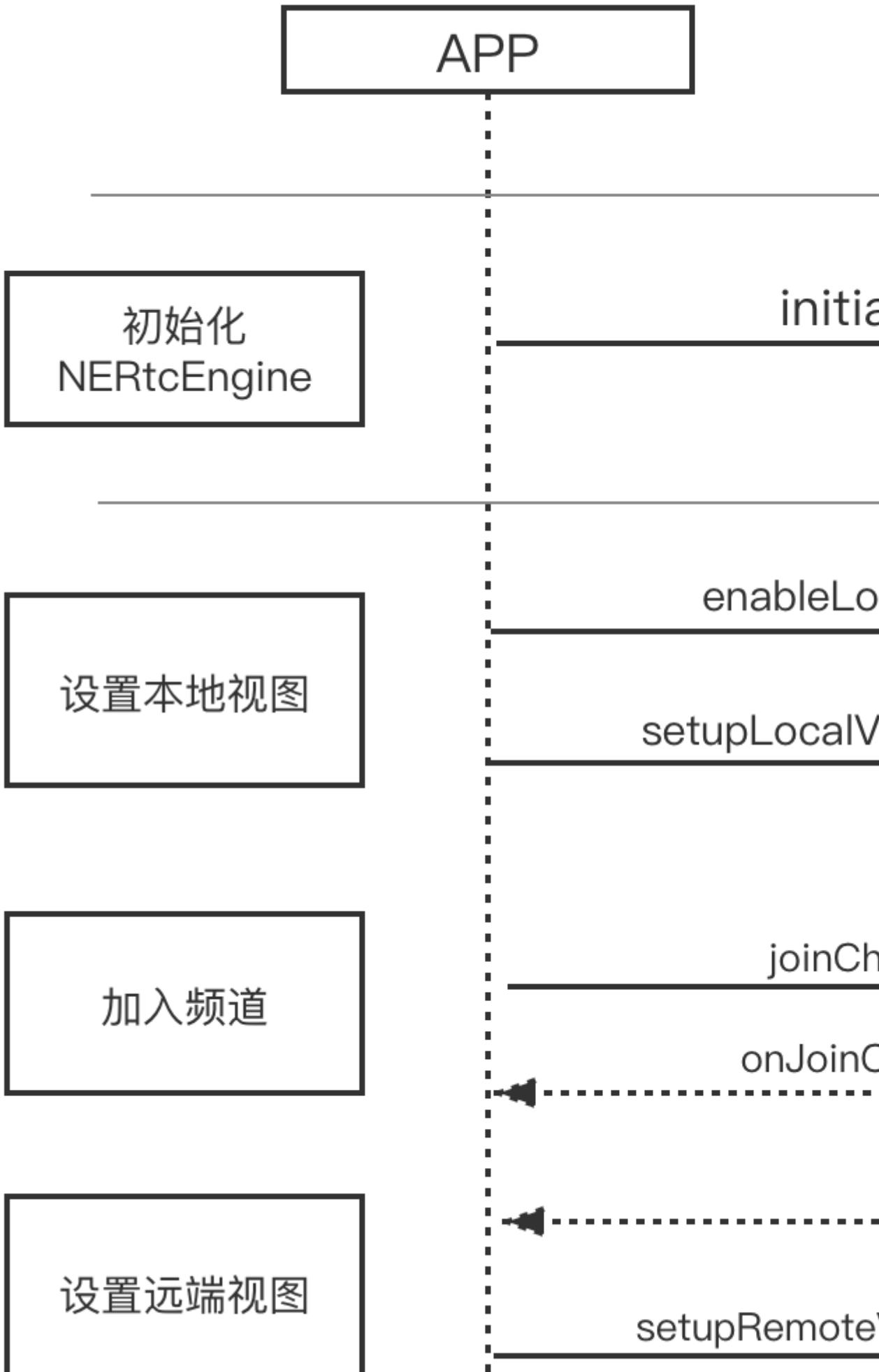
- 集成 SDK（iOS）。

4.7.2.2 API 时序图

NERTC SDK 实现音频通话的主要流程如下图所示。


APP
初始化
NERtcEngine
initia
加入频道
joinCh
离开频道
leaveC
销毁
NERtcEngine
rele

NERTC SDK 实现视频通话的主要流程如下图所示。


APP
初始化
NERtcEngine
initia
enableLo
设置本地视图
setupLocalV
joinCh
加入频道
onJoinC
设置远端视图
setupRemote

4.7.2.3 实现方法

步骤一 导入类

在项目中导入 NERtcSDK 类：

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

步骤二 初始化

调用 setupEngineWithContext: 方法完成初始化。

```
@interface Myapp ()<NERtcEngineDelegateEx>

...

NERtcEngine *coreEngine = [NERtcEngine sharedEngine];

NERtcEngineContext *context = [[NERtcEngineContext alloc] init];

// 设置通话相关信息的回调

context.engineDelegate = self;

// 设置当前应用的 appKey

context.appKey = AppKey;

[coreEngine setupEngineWithContext:context];
```

```
...
```


您需要将 AppKey 替换为您的应用对应的 App Key。

步骤三 设置本地视图

初始化成功后，可以设置本地视图，来预览本地图像。实现加入房间前预览和加入房间后预览的方法如下：

- 实现加入房间前预览。

1. 通过 startPreview 与 setupLocalVideoCanvas 方法，在加入房间前设置本地视图，预览本地图像。

```
2.  // 示例
3.  NERtcVideoCanvas *canvas = [[NERtcVideoCanvas alloc] init];
4.  canvas.container = self.localUserView;
5.  // 设置本地视频画布
6.  [NERtcEngine.sharedEngine setupLocalVideoCanvas:canvas];
7.  // 开启预览
8.  [NERtcEngine.sharedEngine startPreview];
```

9. 若要结束预览，或者准备加入房间时，调用 stopPreview 停止预览。

- 实现加入房间后预览。

  通过 enableLocalVideo 方法进行视频的采集发送与预览。

  您可以在加入房间前执行如下示例代码，加入房间后即可实现预览本地图像。

```
// 示例

[NERtcEngine.sharedEngine enableLocalVideo:YES];

NERtcVideoCanvas *canvas = [[NERtcVideoCanvas alloc] init];

canvas.container = self.localUserView;

[NERtcEngine.sharedEngine setupLocalVideoCanvas:canvas];
```


步骤四 加入房间

加入房间前，请确保已完成初始化相关事项。若您的业务中涉及呼叫邀请等机制，建议通过信令实现。

调用 joinChannelWithToken:channelName:myUid:completion: 方法加入房间。

```
// 示例

[NERtcEngine.sharedEngine joinChannelWithToken:@"Your Token"

                                   channelName: Your roomId
```

```
                                        myUid:Your userId

                                    completion:^(NSError * _Nullable error,
uint64_t channelId, uint64_t elapesd) {

                                                if (error) {

                                                    //加入失败

                                                } else {

                                                    //加入成功

                                                }

                                    }];
```


参数说明

参数	说明
Token	安全认证签名（NERTC Token）。

	调试模式下：可设置为 null。产品默认为安全模式，您可以在网易云信控制台将鉴权模式修改为调试模式，具体请参见 Token 鉴权。 **调试模式的安全性不高，请在产品正式上线前修改为安全模式。** 产品正式上线后：请设置为已获取的 NERTC Token。安全模式下必须设置为获取到的 Token 。若未传入正确的 Token 将无法进入房间。 **推荐使用安全模式。**
channelName	房间名称，长度为 1～64 字节。 设置相同房间名称的用户会进入同一个通话房间。 您也可以在加入通道前，通过创建房间接口创建房间。加入房间时，若传入的 {channelName} 未事先创建，则云信服务器内部将为其自动创建一个名为 {channelName} 的通话房间。
myUid	用户的唯一标识 id，为数字串，房间内每个用户的 uid 必须是唯一的。此 uid 为用户在您应用中的 ID，请在您的业务服务器上自行管理并维护。

步骤五 设置远端视图

音视频通话过程中，除了要显示本地的视频画面，通常也要显示参与互动的其他连麦者/主播的远端视频画面。

1. 监听远端用户进出频道。

NERtcEngineDelegate 通过以下回调获取相关信息：

- onNERtcEngineUserDidJoinWithUserID:userName：监听远端用户加入通话房间的事件，并抛出对方的 uid。当本端加入房间后，也会通过此回调抛出通话房间内已有的其他用户。
- onNERtcEngineUserVideoDidStartWithUserID:videoProfile：监听远端用户发布视频流的事件，回调中携带对方的 uid 与发布的视频分辨率。

2. 设置远端视频画布。

在监听到远端用户加入房间或发布视频流后，本端可以通过 setupRemoteVideoCanvas:forUserID: 方法设置远端用户视频画布，用于显示其视频画面。

示例代码

```
- (void)onNERtcEngineUserDidJoinWithUserID:(uint64_t)userID
userName:(NSString *)userName {




//如果已经 setup 了一个远端的 canvas，则不需要再建立了

...

if (_remoteCanvas != nil) {
```

```
        return;
    }



    //建立远端 canvas，用来渲染远端画面
    NERtcVideoCanvas *canvas = [[NERtcVideoCanvas alloc] init];
    canvas.container = _remoteCanvas;
    [NERtcEngine.sharedEngine setupRemoteVideoCanvas:canvas
forUserID:userID];



- (void)onNERtcEngineUserDidLeaveWithUserID:(uint64_t)userID
reason:(NERtcSessionLeaveReason)reason {



    //如果远端的人离开了，重置远端模型和 UI
    ...
```

```
    [NERtcEngine.sharedEngine setupRemoteVideoCanvas:nil
forUserID:userID];

    ...

    }
```

3. 监听远端视频流发布。

   当房间中的其他用户发出或关闭视频流时，分别会触发以下回调：

- onNERtcEngineUserDidLeaveWithUserID:reason：用户离开房间回调。
- onNERtcEngineUserVideoDidStop：远端用户关闭视频功能回调。

```
@protocol NERtcEngineDelegate <NSObject>

/**

 其他用户打开视频的回调

 @param userID  用户 id

 @param profile  用户发送视频的最大分辨率类型

 */
```

```
- (void)onNERtcEngineUserVideoDidStartWithUserID:(uint64_t)userID
videoProfile:(NERtcVideoProfileType)profile;

/**
 其他用户关闭视频的回调




 @param userID  用户 id
 */
- (void)onNERtcEngineUserVideoDidStop:(uint64_t)userID;@end
```


4. 订阅远端视频流。

在设置完远端视频画布后，且监听到远端用户有视频发布时，本端可以调用 subscribeRemoteVideo:forUserID:streamType: 订阅远端用户的视频流。

示例代码

```
// 监听到远端用户有视频流发布
- (void)onNERtcEngineUserVideoDidStartWithUserID:(uint64_t)userID
```

```
videoProfile:(NERtcVideoProfileType)profile {
    //如果已经订阅过远端视频流，则不需要再订阅了
    ...
    if (_remoteCanvas.subscribedVideo) {
        return;
    }


    //订阅远端视频流。
    _remoteCanvas.subscribedVideo = YES;
    [NERtcEngine.sharedEngine subscribeRemoteVideo:YES
                                        forUserID:userID

streamType:kNERtcRemoteVideoStreamTypeHigh];
    ...}
// 监听到远端用户停止视频流发布
```

```
- (void)onNERtcEngineUserVideoDidStop:(uint64_t)userID {

    if (userID == _remoteCanvas.uid) {

        // 收到此回调后，SDK 内部会取消对应的视频流订阅，无需开发者主动取消订
阅。

        ...

        _remoteStatLab.hidden = YES;

        ...

    }

}
```


步骤六 音频流

本地音频的采集发布和远端音频订阅播放是默认启动的，正常情况下无需开发者主动干预。

步骤七 退出通话房间

通过 leaveChannel 接口退出通话房间。

示例代码

```
//UI 挂断按钮事件- (IBAction)onHungupAction:(UIButton *)sender {
    [NERtcEngine.sharedEngine leaveChannel];
    [self dismiss];
}
```

执行完 leaveChannel 方法后，SDK 会触发离开房间回调 onNERtcEngineDidLeaveChannelWithResult，通知当前用户退出房间的结果。

示例代码

```
- (void)onNERtcEngineDidLeaveChannelWithResult:(NERtcError)result{
    // 进行业务数据清理
}
```

步骤八 销毁音视频实例

当确定 App 短期内不再使用音视频通话实例时，可以通过 destroyEngine 释放对应的对象资源。

释放资源需要在子线程中进行。

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY
_HIGH, 0), ^{
        //开始销毁
      [NERtcEngine destroyEngine];
      //到这里表示销毁完成
  });
```

4.8 Token 鉴权

更新时间： 2021/12/27 11:16:07

网易云信音视频通话和互动直播产品中，鉴权方式分为安全模式和调试模式。如果在控制台中为指定应用开启了安全模式，则对应 App 用户在加入房间时，需要通过 Token 进行身份校验。

本文档为您介绍鉴权方式和 Token 的生成方式。

4.8.1 鉴权方式

4.8.1.1 鉴权方式说明

鉴权方式分为安全模式和调试模式，其区别如下：

	安全模式	调试模式
鉴权方式	默认开启。加入房间时，必须设置正确、可用的 Token，以便网易云信服务器对登录用户进行身份和权限认证。安全性较高。	加入房间时，无需设置 Token，即不对登录用户进行鉴权，安全性较低，可能会有盗刷风险。
应用场景	适用于正式上线的应用。	适用于调试、测试阶段的应用。

4.8.1.2 修改鉴权方式

创建应用并开通音视频通话或互动直播服务后，应用默认的鉴权方式为安全模式。网易云信建议您在应用接入和测试期间使用调试模式，并在正式上线前改为安全模式。

请谨慎选择鉴权方式。如果鉴权方式为安全模式，但 App 用户加入房间时未设置 Token，会导致用户加入房间失败，同时报错 403。

操作步骤：

1. 登录网易云信控制台。
2. 在左侧导航栏单击指定应用名称。
3. 在 **功能管理** 区域中，**音视频通话 2.0** 或 **互动直播 2.0** 右侧单击 **功能配置**。

4. 在 **鉴权方式** 区域中，选择授权方式，并单击 **确定**。

5. 在弹出对话框中单击 **确定**。

4.8.2 申请 Token

安全模式下，需要使用 NERTC Token 才能加入房间。获取 NERTC Token 的基本流程如下：

1. 客户端向应用服务器发起安全认证签名的请求。

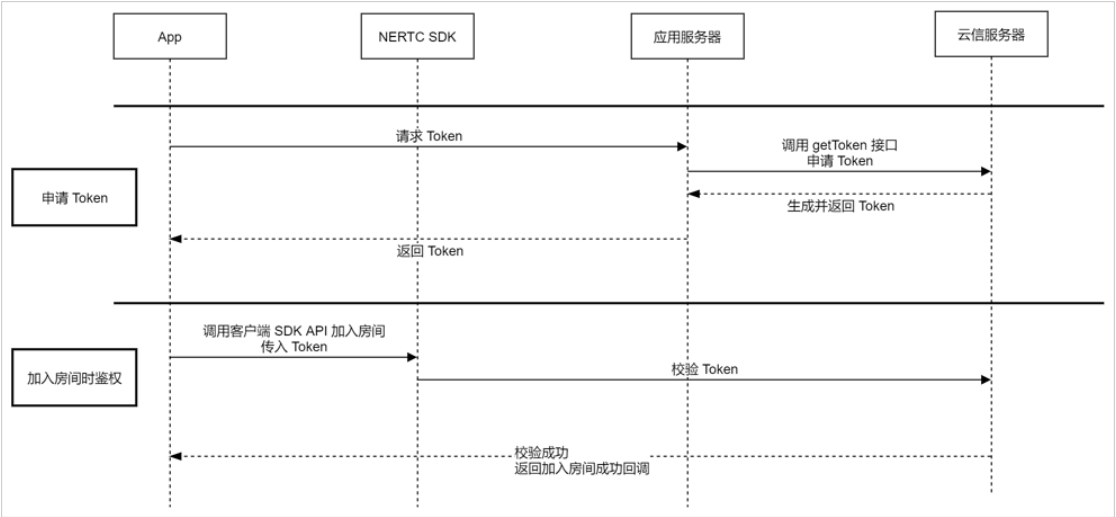
   该步骤交互由开发者自行完成。

2. 应用服务器调用接口 getToken，向云信服务器申请 NERTC Token。

   请求成功，云信服务器会将 NERTC Token 返回给应用服务器。

3. 应用服务器将获取到的 NERTC Token 返回给客户端。

   客户端通过以上步骤获取 NERTC Token 之后，可以携带 NERTC Token 加入房间。



4.8.3 API 参考

如果需要更新 NERTC Token 有效期，可以在该 Token 过期之前，重复使用同样的 uid 和 channelName 再次调用该接口，通过 expireAt 参数修改 Token 有效期。服务端不会返回新的 Token，但会更新此 Token 的有效期。

请求 Header 的相关说明请参考服务端 API 请求结构。

- getToken 接口的 Content-Type 为 application/x-www-form-urlencoded，服务端其他 API 接口的 Content-Type 为 application/json。
- getToken 接口的 channelName 为可选参数，如果首次获取 Token 时通过 channelName 设置了 Token 对应的房间名称，并使用该 Token 加入房间，之后该用户加入该房间的 Token 都应绑定此房间名称。

```
POST https://api.netease.im/nimserver/user/getToken.action HTTP/1.1

Content-Type:application/x-www-form-urlencoded;charset=utf-8
```

4.8.3.1 请求参数

参数名称	类型	是否必选	示例值	描述
uid	Long	必选	123456	用户 ID。
channelName	String	可选	abc	绑定的房间名称。未指定 channelName 时，获取的 Token 可以用于加入任何房间。
repeatUse	Boolean	可选	true	在 Token 有效期内该用户是否可以多次使用该 Token，默认为 true。 true：有效期内可多次使用。

				false：有效期内只能使用一次。
expireAt	int	可选	640	NERTC Token 的过期时间，过期后，该用户将无法通过此 Token 加入房间。 取值范围为 1~86400 秒，默认为 600 秒。

4.8.3.2 请求示例

curl 请求示例如下：

```
curl -X POST -H "AppKey: go9dnk********" -H "Nonce: 4tggger********" -H "CurTime: 1443592222" -H "CheckSum: 9e9db3b6c9ab**********" -H "Content-Type: application/x-www-form-urlencoded" -d 'uid=123456' 'https://api.netease.im/nimserver/user/getToken.action'
```

HttpClient 请求示例如下，此处以 JAVA 为例。

```
import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpPost;
```

```
import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.message.BasicNameValuePair;

import org.apache.http.util.EntityUtils;



import java.util.ArrayList;

import java.util.Date;

import java.util.List;



public class Test {

    public static void main(String[] args) throws Exception{

        DefaultHttpClient httpClient = new DefaultHttpClient();

        String url = "https://api.netease.im/nimserver/user/getToken.action";

        HttpPost httpPost = new HttpPost(url);



        String appKey = "94kid09c9ig9k1loimjg012345123456";

        String appSecret = "123456789012";
```

```
        String nonce =    "12345";

        String curTime = String.valueOf((new Date()).getTime() / 1000L);

        String checkSum = CheckSumBuilder.getCheckSum(appSecret,
nonce ,curTime);///checkSum 的计算方式请参考《 服务端 API-调用方式-请求结构 》



        // 设置请求的 header

        httpPost.addHeader("AppKey", appKey);

        httpPost.addHeader("Nonce", nonce);

        httpPost.addHeader("CurTime", curTime);

        httpPost.addHeader("CheckSum", checkSum);

        httpPost.addHeader("Content-Type", "application/x-www-form-
urlencoded;charset=utf-8");



        // 设置请求的参数

        List<NameValuePair> nvps = new ArrayList<NameValuePair>();

        nvps.add(new BasicNameValuePair("uid", "123456"));

        httpPost.setEntity(new UrlEncodedFormEntity(nvps, "utf-8"));
```

```
        // 执行请求
        HttpResponse response = httpClient.execute(httpPost);

        // 打印执行结果
        System.out.println(EntityUtils.toString(response.getEntity(), "utf-8"));
    }
}
```

4.8.3.3 返回参数

参数名称	类型	示例值	描述
code	Number	200	状态码。200 表示成功调用。
token	String	xxxxx	服务端返回的 Token。

4.8.3.4 返回示例

JSON 格式的返回示例如下：

```
"Content-Type": "application/json; charset=utf-8"

{

    "code":200,

    "token":"xxxxx"    //安全认证模式下的签名。

}
```

五、 基础功能

5.1 设置音频属性

5.1.1 Android

NERTC SDK 支持设置音频编码属性设置，本文档为您介绍如何通过 setAudioProfile 方法的 profile 和 scenario 参数实现不同的音质效果。

5.1.1.1 功能描述

在不同的场景中，用户对于音质、声道、噪声抑制等方面的要求各有不同，例如在音乐直播、在线 KTV 等场景中，需要设置适宜于音乐场景的采样率、码率、流畅度、噪声抑制等参数和能力。NERTC 支持多种音频属性档位设置，适用于不同场景的音质、声道需求，以便您根据实际场景方便快捷地调整音质属性、在常见场景中实现最优的音质效果。

NERTC SDK 通过 setAudioProfile 方法的 profile 和 scenario 参数提供多种音频 Profile 档位和音频场景设置，您可以根据实际场景需求灵活调整音质属性。

5.1.1.2 注意事项

SDK 支持在加入房间通话中后，您根据实际场景变化动态调用 setAudioProfile 方法。

5.1.1.3 实现方法

通过 setAudioProfile 设置音频属性。

设置音频模式

profile 参数代表不同的音频参数配置（音质），比如采样率、码率和编码模式，包括：

音频 profile 类型	描述
DEFAULT	默认设置。语音场景下为 STANDARD，音乐场景下为 HIGH_QUALITY。
STANDARD	标准音质模式。采样率为 16 kHz、语音编码、单声道、编码码率最大值为 20 Kbps。
MIDDLE_QUALITY	中等音质模式。采样率为 48 kHz、音乐编码、单声道、编码码率最大值为 64 Kbps。
MIDDLE_QUALITY_STEREO	中等音质模式（立体音）。采样率为 48 kHz、音乐编码、双声道、编码码率最大值为 80 Kbps。
HIGH_QUALITY	高音质模式。采样率为 48 kHz、音乐编码、

音频 profile 类型	描述
	单声道、编码码率最大值为 96 Kbps。
HIGH_QUALITY_STEREO	高音质模式（立体音）。采样率为 48 kHz、音乐编码、双声道、编码码率最大值为 128 Kbps。
STANDARD_EXTEND	标准扩展模式。采样率为 16 kHz、语音编码、单声道、编码码率最大值为 32 Kbps。

设置音频场景

scenario 参数用于设置音频的使用场景，在不同场景中，声音的流畅度、噪声抑制、音质等会根据不同的场景做出优化。目前支持以下类型的场景：

scenario 场景	描述	音量
DEFAULT	默认的音频应用场景（语音）。	通话音

scenario 场景	描述	音量
		量。
SPEECH	语音场景。推荐用于普通的音视频通话场景。	通话音量。
MUSIC	音乐场景。音乐级别音质设置，高保真、超流畅，推荐用于在线 KTV、在线演唱会、音乐直播、乐器教学等高音质要求的场景。	媒体音量。
CHATROOM	语音聊天室场景。优先保障音频连续性与稳定性，适用于用户需要频繁上下麦的场景。仅移动端支持。	通话音量。

不同的 scenario 下，设备的系统音量使用不同的音量类型。

参数搭配推荐

业务场景	profile 设置	scenario 设置	特性
游戏开黑	HIGH_QUALITY	CHATROOM	在保证高音质的同时节省流量，频繁上下麦时不卡顿。连麦时保留人声、减少杂音，适用于游戏语音场景。
语聊房（音乐场景）	HIGH_QUALITY 或 HIGH_QUALITY_STEREO	MUSIC	音乐级高清音质，人声和背景音乐声音都有丰富的呈现，可搭配音效功能实现更有趣味性的体验。
在线	HIGH_QUALITY_STEREO	MUSIC	高音质立体声，适用于音质要求较高

业务场景	profile 设置	scenario 设置	特性
KTV			的场景，还可以搭配丰富的音效增加娱乐效果。
互动直播（音乐场景）	HIGH_QUALITY 或 HIGH_QUALITY_STEREO	MUSIC 或 CHATROOM	高清音质。可根据实际直播场景选择单双声道，音乐场景推荐使用 MUSIC。
音乐教学	HIGH_QUALITY	MUSIC	优先保证高清音质，音乐级音频能力。适用于对音质要求较高的场景。

业务场景	profile 设置	scenario 设置	特性
一对一通话 语聊房（无音乐） 互动直播（无音乐） 多人通话/视	STANDARD	SPEECH	在保证传输流畅的同时，维持高清音质，在弱网环境下优先保证通话稳定、流畅。 SPEECH 模式下，回声、噪声等环境音过滤效果较好，可以提供更为清晰的人声语音效果。

业务场景	profile 设置	scenario 设置	特性
频会议 小班课/大班课 双师课堂			

5.1.1.4 示例代码

```
    NERtcEx.getInstance().setParameters(mRtcParameters); //先设置参数，后初始化
    try {

NERtcEx.getInstance().init(getApplicationContext(),Config.APP_KEY,this,null);

    } catch (Exception e) {
```

```
        showToast("SDK 初始化失败");
        finish();
        return;
    }
    NERtcEx.getInstance().setAudioProfile(mAudioProfile,mAudioScenario);//
初始化后设置音频场景
```


5.1.1.5 API 参考

方法	功能描述
setAudioProfile	设置音频场景与模式，可以在加入房间后设置。
enableLocalAudio	开启/关闭本地语音采集和发送。

方法	功能描述
muteLocalAudio	开关本地音频发送。
subscribeRemoteAudioStream	订阅 / 取消订阅指定音频流。
subscribeAllRemoteAudioStreams	订阅 / 取消订阅所有用户音频，对后续加入的用户也同样生效。

5.1.2 iOS

NERTC SDK 支持设置音频编码属性设置，本文档为您介绍如何通过 setAudioProfile 方法的 profile 和 scenario 参数实现不同的音质效果。

5.1.2.1 功能描述

在不同的场景中，用户对于音质、声道、噪声抑制等方面的要求各有不同，例如在音乐直播、在线 KTV 等场景中，需要设置适宜于音乐场景的采样率、码率、流畅度、噪声抑制等参数和能力。NERTC 支持多种音频属性档位设置，适用于不同场景的音质、声道需求，以便您根据实际场景方便快捷地调整音质属性、在常见场景中实现最优的音质效果。

NERTC SDK 通过 setAudioProfile 方法的 profile 和 scenario 参数提供多种音频 Profile 档位和音频场景设置，您可以根据实际场景需求灵活调整音质属性。

5.1.2.2 注意事项

SDK 支持在加入房间通话中后，您根据实际场景变化动态调用 setAudioProfile 方法。

5.1.2.3 实现方法

通过 setAudioProfile 设置音频属性。

设置音频模式

profile 参数代表不同的音频参数配置（音质），比如采样率、码率和编码模式，包括：

音频 profile 类型	描述
kNERtcAudioProfileDefault	默认设置。语音场景下为 kNERtcAudioProfileStandard，音乐场景下为 kNERtcAudioProfileHighQuality。
kNERtcAudioProfileStandard	标准音质模式。采样率为 16 kHz、语音编码、单声道、编码码率最大值为 20 Kbps。

音频 profile 类型	描述
kNERtcAudioProfileStandardExtend	标准扩展模式。采样率为 16 kHz、语音编码、单声道、编码码率最大值为 32 Kbps。
kNERtcAudioProfileMiddleQuality	中等音质模式。采样率为 48 kHz、音乐编码、单声道、编码码率最大值为 64 Kbps。
kNERtcAudioProfileMiddleQualityStereo	中等音质模式（立体音）。采样率为 48 kHz、音乐编码、双声道、编码码率最大值为 80 Kbps。
kNERtcAudioProfileHighQuality	高音质模式。采样率为 48 kHz、音乐编码、单声道、编码码率最大值为 96 Kbps。
kNERtcAudioProfileHighQualityStere	高音质模式（立体音）。采样率为 48 kHz、音乐编码、双声道、编码码率

音频 profile 类型	描述
o	最大值为 128 Kbps。

设置音频场景

scenario 参数用于设置音频的使用场景，在不同场景中，声音的流畅度、噪声抑制、音质等会根据不同的场景做出优化。目前支持以下类型的场景：

scenario 场景	描述	音量
kNERtcAudioScenarioDefault	默认的音频应用场景（语音）。	通话音量。
kNERtcAudioScenarioSpeech	语音场景。推荐用于普通的音视频通话场景。	通话音量。

scenario 场景	描述	音量
kNERtcAudioScenarioMusic	音乐场景。音乐级别音质设置，高保真、超流畅，推荐用于在线KTV、在线演唱会、音乐直播、乐器教学等高音质要求的场景。	媒体音量。
kNERtcAudioScenarioChatRoom	语音聊天室场景。优先保障音频连续性与稳定性，推荐用于在线语音聊天室场景。仅移动端支持。	通话音量。

不同的 scenario 下，设备的系统音量使用不同的音量类型。

参数搭配推荐

业务场景	profile 设置	scenario 设置	特性
游戏开黑	HighQuality	ChatRoom	在保证高音质的同时节省流量，

业务场景	profile 设置	scenario 设置	特性
			频繁上下麦时不卡顿。 连麦时保留人声、减少杂音，适用于游戏语音场景。
语聊房（音乐场景）	HighQuality 或 HighQualityStereo	Music	音乐级高清音质，人声和背景音乐声音都有丰富的呈现，可搭配音效功能实现更有趣味性的体验。
在线 KTV	HighQualityStereo	Music	高音质立体声，适用于音质要求

业务场景	profile 设置	scenario 设置	特性
			较高的场景，还可以搭配丰富的音效增加娱乐效果。
互动直播（音乐场景）	HighQuality 或 HighQualityStereo	Music 或 ChatRoom	高清音质。可根据实际直播场景选择单双声道，音乐场景推荐使用 Music。
音乐教学	HighQuality	Music	优先保证高清音质，音乐级音频能力。适用于对音质要求较高的场景。

业务场景	profile 设置	scenario 设置	特性
一对一通话 语聊房（无音乐） 互动直播（无音乐） 多人通话/视频会议 小班课/大班课 双师课堂	Standard	Speech	在保证传输流畅的同时，维持高清音质，在弱网环境下优先保证通话稳定、流畅。Speech 模式下，回声、噪声等环境音过滤效果较好，可以提供更为清晰的人声语音效果。

5.1.2.4 示例代码

```
// 在初始化后设置

[[NERtcEngine sharedEngine] setAudioProfile:profile scenario:scenario]
```

5.1.2.5 API 参考

方法	功能描述
setAudioProfile	设置音频编码配置，可以在加入房间后设置。
enableLocalAudio	开关本地音频采集
muteLocalAudio	开关本地音频发送
subscribeRemoteAudio	订阅 / 取消订阅指定音频流
subscribeAllRemoteAudio	订阅 / 取消订阅所有远端音频流

5.1.2.6 常见问题

- 为什么在我连接了蓝牙耳机的情况下，开启本地音频采集后，远端声音变外放?

5.2 设置视频属性

5.2.1 Android

在视频通话前，开发者可以根据不同场景下的用户喜好与需求，对视频属性进行设置，调整发送视频画面的清晰度以及流畅度，带给用户最优的通话体验。

视频属性通常包含分辨率、帧率、码率的档位，裁剪模式等参数。

NERTC SDK 通过 setLocalVideoConfig 设置视频属性，包括视频编码的分辨率、码率、帧率、适应性偏好等。

5.2.1.1 注意事项

- NERTC SDK 会根据实时网络环境，对设置的参数作自适应调整，通常会下调参数。
- 本文中各参数的设置可能会影响计费。
- setVideoConfig 为全量参数配置接口，重复调用此接口时，SDK 会刷新此前的所有参数配置，以最新的传参为准。所以每次修改配置时都需要设置所有参数，未设置的参数将取默认值。

5.2.1.2 设置视频编码分辨率、码率和帧率

您可以自定义视频编码的分辨率、码率或帧率，各参数的设置方法如下。

此处设置的数值均为理想情况下的最大值，实际场景中，如果视频引擎因网络环境或设备采集能力等原因无法达到设置的分辨率、帧率或码率的最大值时，会根据设备采集能力自动向下取值。

- **分辨率**
  - 视频编码的分辨率可以用来衡量编码质量，通常情况下分辨率越高，视频越清晰。
  - 通过 width 和 height 设置摄像头采集视频的编码分辨率（px）。默认为 640 x 360，取值范围为 64x64 ~ 1920 x 1080，如果设置取值范围以外的数值，则保持默认设置，即 640 x 360。
- **帧率**
  - 视频编码帧率（fps）指是每秒钟编码多少帧画面。
  - 通过 frameRate 可以设置视频编码帧率。
- **码率**
  - 码率（Bitrate）指每秒钟编码器输出多少 Kbit 的编码后的二进制数据。
  - 通过 bitrate 可以设置视频编码码率。

    视频的分辨率、码率和帧率并不是越高越好，三者之间需要维持一个平衡的映射关系关系，您可以参考分辨率、帧率和码率推荐表设置分辨率和码率。

5.2.1.3 设置适应性偏好

在弱网环境下，视频的清晰度和流畅度可能无法兼顾，为保证音视频通话或互动直播场景下的视频体验，您可以通过 degradationPrefer 参数设置带宽受限时视频编码的适应性偏好。

参数	描述
DEGRADATION_BALANCED	在编码帧率和视频质量之间保持平衡。
DEGRADATION_DEFAULT	根据场景模式调整适应性偏好。 通信场景中，选择 DEGRADATION_BALANCED 模式，在编码帧率和视频质量之间保持平衡。 直播场景中，选择 DEGRADATION_MAINTAIN_QUALITY 模式，降低编码帧率以保证视频质量。
DEGRADATION_MAINTAIN_FRAMERATE	流畅优先，降低视频质量以保证编码帧率。在弱网环境下，降低视频清晰度以保证视频流畅，此时画质降低，画面会变得模糊，但可以保持视频流畅。
DEGRADATION_MAINTAIN_QUALITY	清晰优先，降低编码帧率以保证视频质量。在弱网环境下，降低视频帧率以保证

参数	描述
	视频清晰，此时可能会出现一定卡顿。

5.2.1.4 设置旋转方向模式

旋转方向模式指移动端视频是横屏还是竖屏显示。采集端采集图像时，同时收集视频的旋转方向模式信息；播放端将接收到的视频图像进行渲染，并旋转方向模式信息对应地旋转视频，以达到和采集端表现一致。

NERTC SDK 支持根据 orientationMode 参数设置本地视频编码的旋转方向模式，使接收端的视频展示为指定的方向效果。其中，orientationMode 参数中提供了 VIDEO_OUTPUT_ORIENTATION_MODE_ADAPTATIVE（自适应）、VIDEO_OUTPUT_ORIENTATION_MODE_FIXED_LANDSCAPE（风景） 和 VIDEO_OUTPUT_ORIENTATION_MODE_FIXED_PORTRAIT（肖像） 三种方向模式。

- VIDEO_OUTPUT_ORIENTATION_MODE_ADAPTATIVE

  自适应模式下，SDK 输出的视频方向与采集到的视频方向一致。接收端会根据收到的视频旋转信息对视频进行旋转。该模式适用于接收端可以自主调整视频方向的场景。
- VIDEO_OUTPUT_ORIENTATION_MODE_FIXED_LANDSCAPE

  该模式下 SDK 固定输出横屏模式的视频。如果采集到的视频是竖屏模式，则视频编码器会对其进行裁剪。该模式适用于当接收端无法调整视频方向时，例如旁路推流场景。
- VIDEO_OUTPUT_ORIENTATION_MODE_FIXED_PORTRAIT

该模式下 SDK 固定输出竖屏模式的视频，如果采集到的视频是横屏模式，则视频编码器会对其进行裁剪。该模式适用于当接收端无法调整视频方向时，例如旁路推流场景。

若您集成的是 V4.6.0 之前的版本，设置旋转方向后，只会影响远端用户看到的视频画面；若您集成的是 V4.6.0 及之后的版本，设置旋转方向后，会同时影响本端用户的预览画面和远端用户看到的视频画面。

5.2.1.5 设置镜像模式

您可以通过 mirrorMode 参数来设置视频编码的镜像模式，影响远端用户看到的视频画面。

默认情况下，编码时由 SDK 决定镜像模式。移动端如果使用前置摄像头，则默认开启镜像模式；如果使用后置摄像头，则关闭镜像模式。

NERTC 同时支持设置视频画布的镜像模式，即画面是否左右翻转。设置画布时 NERtcVideoView.setMirror 设置为 true 表示开启镜像模式，默认情况下镜像模式为关闭状态。

5.2.1.6 示例代码

```
// 配置一个 NERtcVideoConfig 实例，参数可参考下文中的 API 参考链接
 NERtcVideoConfig videoConfig = new NERtcVideoConfig();
 videoConfig.frontCamera = mFrontCamera;
 videoConfig.videoProfile = mVideoProfile;
```

```
videoConfig.width = mVideoWidth;

videoConfig.height = mVideoHeight;

videoConfig.videoCropMode = mVideoCropMode;

videoConfig.colorFormat = mVideoColorFormat;

videoConfig.frameRate = mVideoFps;

videoConfig.minFramerate = mVideoMinFps;

videoConfig.degradationPrefer = mVideoDegradation;

videoConfig.bitrate = mVideoEncodeBitrate;

videoConfig.minBitrate = mVideoEncodeMinBitrate;

NERtcEx.getInstance().setLocalVideoConfig(videoConfig); //设置本地视频参数
```


5.2.1.7 参考信息

分辨率、帧率和码率推荐表

在大多数场景下，您可以根据实际业务场景中的视频窗口的大小去配置分辨率、帧率、码率，以控制带宽压力及编解码的资源消耗。 在自定义视频编码参数的过程中，您也可以参考下表对这些参数进行自定义配置。

注意：下表中的视频参数推荐仅适用于 Native 端，即 Android、iOS、Windows、macOS 平台。

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
90 x 90	30	49	73
90 x 90	15	32	48
120 x 90	30	61	91
120 x 90	15	40	60
120 x 120	30	75	113
120 x 120	15	50	75
160 x 90	30	75	113

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
160 x 90	15	50	75
160 x 120	30	94	141
160 x 120	15	62	93
180 x 180	30	139	208
180 x 180	15	91	137
240 x 180	30	172	259
240 x 180	15	113	170
240 x 240	30	214	321

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
240 x 240	15	141	212
320 x 180	30	214	321
320 x 180	15	141	212
320 x 240	30	259	389
320 x 240	15	175	263
360 x 360	30	393	590
360 x 360	15	259	389
424 x 240	15	217	325

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
480 x 360	30	488	732
480 x 360	15	322	483
480 x 480	30	606	909
480 x 480	15	400	600
640 x 360	30	606	909
640 x 360	15	400	600
640 x 480	30	752	1128
640 x 480	15	496	744

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
720 x 720	30	1113	1670
720 x 720	15	734	1102
848 x 480	30	929	1394
720 x 720	15	613	919
960 x 720	30	1382	2073
960 x 720	15	911	1367
1080 x 1080	30	2046	3069
1080 x	15	1350	2025

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
1080			
1280 x 720	30	1714	2572
1280 x 720	15	1131	1697
1440 x 1080	30	2538	3808
1440 x 1080	15	1675	2512
1920 x 1080	30	3150	4725
1920 x 1080	15	2078	3117

5.2.1.8 常见问题

- 为什么我关闭摄像头后重新打开，原来设置的后置摄像头变成了前置?

5.2.2 iOS

在视频通话前，开发者可以根据不同场景下的用户喜好与需求，对视频属性进行设置，调整发送视频画面的清晰度以及流畅度，带给用户最优的通话体验。 视频属性通常包含分辨率、帧率、码率的档位，裁剪模式等参数。

NERTC SDK 通过 setLocalVideoConfig 设置视频属性，包括视频编码的分辨率、码率、帧率、适应性偏好等。

5.2.2.1 注意事项

- NERTC SDK 会根据实时网络环境，对设置的参数作自适应调整，通常会下调参数。
- 本文中各参数的设置可能会影响计费。
- setVideoConfig 为全量参数配置接口，重复调用此接口时，SDK 会刷新此前的所有参数配置，以最新的传参为准。所以每次修改配置时都需要设置所有参数，未设置的参数将取默认值。

5.2.2.2 设置视频编码分辨率、码率和帧率

您可以自定义视频编码的分辨率、码率或帧率，各参数的设置方法如下。

此处设置的数值均为理想情况下的最大值，实际场景中，如果视频引擎因网络环境或设备采集能力等原因无法达到设置的分辨率、帧率或码率的最大值时，会根据设备采集能力自动向下取值。

- **分辨率**
  - 视频编码的分辨率可以用来衡量编码质量，通常情况下分辨率越高，视频越清晰。
  - 通过 width 和 height 设置摄像头采集视频的编码分辨率（px）。默认为 640 x 360，取值范围为 64x64 ~ 1920 x 1080，如果设置取值范围以外的数值，则保持默认设置，即 640 x 360。
- **帧率**
  - 视频编码帧率（fps）指是每秒钟编码多少帧画面。
  - 通过 frameRate 可以设置视频编码帧率。
- **码率**
  - 码率（Bitrate）指每秒钟编码器输出多少 Kbit 的编码后的二进制数据。
  - 通过 bitrate 可以设置视频编码码率。

视频的分辨率、码率和帧率并不是越高越好，三者之间需要维持一个平衡的映射关系关系，您可以参考分辨率、帧率和码率推荐表设置分辨率和码率。

5.2.2.3 设置适应性偏好

在弱网环境下，视频的清晰度和流畅度可能无法兼顾，为保证音视频通话或互动直播场景下的视频体验，您可以通过 degradationPreference 参数设置带宽受限时视频编码的适应性偏好。

参数	描述
kNERtcDegradationBalanced	在编码帧率和视频质量之间保持平衡。
kNERtcDegradationDefault	根据场景模式调整适应性偏好。 通信场景中，选择kNERtcDegradationBalanced模式，在编码帧率和视频质量之间保持平衡。 直播场景中，选择kNERtcDegradationMaintainQuality模式，降低编码帧率以保证视频质量。
kNERtcDegradationMaintainFramerate	流畅优先，降低视频质量以保证编码帧率。在弱网环境下，降低视频清晰度以保证视频流畅，此时画质降低，画面会变得模糊，但可以保持视频流畅。

参数	描述
kNERtcDegradationMaintainQuality	清晰优先，降低编码帧率以保证视频质量。在弱网环境下，降低视频帧率以保证视频清晰，此时可能会出现一定卡顿。

5.2.2.4 设置旋转方向模式

旋转方向模式指移动端视频是横屏还是竖屏显示。采集端采集图像时，同时收集视频的旋转方向模式信息；播放端将接收到的视频图像进行渲染，并旋转方向模式信息对应地旋转视频，以达到和采集端表现一致。

NERTC SDK 支持根据 orientationMode 参数设置本地视频编码的旋转方向模式，使接收端的视频展示为指定的方向效果。其中，orientationMode 参数中提供了 kNERtcVideoOutputOrientationModeAdaptative（自适应）、kNERtcVideoOutputOrientationModeFixedLandscape（风景） 和 kNERtcVideoOutputOrientationModeFixedPortrait （肖像） 三种方向模式。

- kNERtcVideoOutputOrientationModeAdaptative

  自适应模式下，SDK 输出的视频方向与采集到的视频方向一致。接收端会根据收到的视频旋转信息对视频进行旋转。该模式适用于接收端可以自主调整视频方向的场景。

- kNERtcVideoOutputOrientationModeFixedLandscape

该模式下 SDK 固定输出横屏模式的视频。如果采集到的视频是竖屏模式，则视频编码器会对其进行裁剪。该模式适用于当接收端无法调整视频方向时，例如旁路推流场景。

- kNERtcVideoOutputOrientationModeFixedPortrait

  该模式下 SDK 固定输出竖屏模式的视频，如果采集到的视频是横屏模式，则视频编码器会对其进行裁剪。该模式适用于当接收端无法调整视频方向时，例如旁路推流场景。

  若您集成的是 V4.6.0 之前的版本，设置旋转方向后，只会影响远端用户看到的视频画面；若您集成的是 V4.6.0 及之后的版本，设置旋转方向后，会同时影响本端用户的预览画面和远端用户看到的视频画面。

5.2.2.5 设置镜像模式

您可以通过 mirrorMode 参数来设置视频编码的镜像模式，影响远端用户看到的视频画面。

默认情况下，编码时由 SDK 决定镜像模式。移动端如果使用前置摄像头，则默认开启镜像模式；如果使用后置摄像头，则关闭镜像模式。

NERTC SDK 还支持在设置画布时调整本地和远端视频画面的镜像模式，即画面是否左右翻转。NERtcVideoCanvas 中的 mirrorMode 设置为 kNERtcVideoMirrorModeEnabled 表示开启镜像模式。

5.2.2.6 示例代码

请确保已在项目中实现了基本的音视频通信或直播功能。 使用 setLocalVideoConfig 方法来设置视频相关的属性，比如分辨率、码率、帧率等。配置中的具体含义请参考 NERtcVideoProfileType 定义。

```
NERtcEngine *coreEngine = [NERtcEngine sharedEngine];

NERtcVideoEncodeConfiguration *config = [[NERtcVideoEncodeConfiguration
alloc] init];

config.width = 640;//  设置分辨率宽

config.height = 360;//  设置分辨率高

config.cropMode = kNERtcVideoCropMode16_9;          //  设置裁剪格式为 16:9

config.frameRate = kNERtcVideoFrameRateFps30; //视频帧率

config.minFrameRate = mMinFrameRate; //视频最小帧率

config.bitrate = mBitrate; //视频编码码率

config.minBitrate = mMinBitrate; //视频编码最小码率

config.degradationPreference = kNERtcDegradationDefault; ;//带宽受限时的视频
编码降级偏好



[coreEngine setLocalVideoConfig:config];//设置本地视频参数



[coreEngine enableDualStreamMode:mVideoDual];//是否开启小流
```

5.2.2.7 参考信息

分辨率、帧率和码率推荐表

在大多数场景下，您可以根据实际业务场景中的视频窗口的大小去配置分辨率、帧率、码率，以控制带宽压力及编解码的资源消耗。 在自定义视频编码参数的过程中，您也可以参考下表对这些参数进行自定义配置。

注意：下表中的视频参数推荐仅适用于 Native 端，即 Android、iOS、Windows、macOS 平台。

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
90 x 90	30	49	73
90 x 90	15	32	48
120 x 90	30	61	91
120 x 90	15	40	60
120 x 120	30	75	113

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
120 x 120	15	50	75
160 x 90	30	75	113
160 x 90	15	50	75
160 x 120	30	94	141
160 x 120	15	62	93
180 x 180	30	139	208
180 x 180	15	91	137
240 x 180	30	172	259

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
240 x 180	15	113	170
240 x 240	30	214	321
240 x 240	15	141	212
320 x 180	30	214	321
320 x 180	15	141	212
320 x 240	30	259	389
320 x 240	15	175	263
360 x 360	30	393	590

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
360 x 360	15	259	389
424 x 240	15	217	325
480 x 360	30	488	732
480 x 360	15	322	483
480 x 480	30	606	909
480 x 480	15	400	600
640 x 360	30	606	909
640 x 360	15	400	600

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
640 x 480	30	752	1128
640 x 480	15	496	744
720 x 720	30	1113	1670
720 x 720	15	734	1102
848 x 480	30	929	1394
720 x 720	15	613	919
960 x 720	30	1382	2073
960 x 720	15	911	1367

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
1080 x 1080	30	2046	3069
1080 x 1080	15	1350	2025
1280 x 720	30	1714	2572
1280 x 720	15	1131	1697
1440 x 1080	30	2538	3808
1440 x 1080	15	1675	2512
1920 x	30	3150	4725

分辨率	帧率（fps）	通信场景码率(kbps)	直播场景码率(kbps)
1080			
1920 x 1080	15	2078	3117

5.2.2.8 常见问题

- 为什么我关闭摄像头后重新打开，原来设置的后置摄像头变成了前置?

5.3 设置通话音量

5.3.1 Android

在音视频通话和互动直播过程中，为了提高产品使用者的体验，NERTC SDK 支持调整各种声音的音量，例如调整 SDK 采集的声音、播放的声音等。音量调节功能适用于多种需要自定义调节音量的场景。

除此之外，NERTC SDK 也支持通过监听回调的方式获取房间内成员的音量。在一些语音连麦场景或者视频会议场景中，产品应用层常常需要获取发言者的音量，并通过 UI 进行音量展示，或者根据发言者的音量大小进行视图布局的动态调整。详细信息请参考监测发言者音量。

5.3.1.1 设置采集音量

NERTC SDK 通过录音设备采集房间内成员的音频信号，将其录制下来以后播放给远端用户收听。您可以通过 adjustRecordingSignalVolume 设置录制声音的信号幅度，从而达到调节采集音量的目的。

该方法中通过 volume 参数设置录音信号的音量，取值范围为 0～400：

- 0：静音。
- 100：（默认值）原始音量，即不对信号做缩放。
- 400：原始音量的 4 倍，即把信号放大到原始信号的 4 倍。

建议设置采集音量时使用默认值（100）。

5.3.1.2 设置播放音量

在音视频通话过程中，音频信号从发送端进入到接收端，然后使用播放设备进行播放。您可以通过 adjustPlaybackSignalVolume 方法或 adjustUserPlaybackSignalVolume 方法调节播放声音的信号幅度，从而达到调节播放音量的目的。

- 通过 adjustPlaybackSignalVolume 可以调节所有远端用户混音后的音频流在本地播放的音量，通过 volume 参数设置播放信号的音量，取值范围为 0～400：
  - 0：静音。
  - 100：（默认值）原始音量，即不对信号做缩放。
  - 400：原始音量的 4 倍，即把信号放大到原始信号的 4 倍。

建议设置本地播放音量时使用默认值（100）或小于该值，否则可能会导致音质问题。

- 通过 adjustUserPlaybackSignalVolume 可以调节指定远端用户混音后的音频流在本地播放的音量。
  - uid 用于指定远端用户，volume 参数用于设置播放信号的音量，取值范围为 0～100。
  - 多次调用该方法，可以设置不同远端用户在本地播放的音量；也可以反复调节某个远端用户在本地播放的音量。

5.3.1.3 设置耳返音量

通过 enableEarback 开启耳返功能之后，您可以通过 setEarbackVolume 设置耳返音量。

该方法中通过 volume 参数设置播放信号的音量，取值范围为 0～100，默认为 100。

5.3.2 iOS

在音视频通话和互动直播过程中，为了提高产品使用者的体验，NERTC SDK 支持调整各种声音的音量，例如调整 SDK 采集的声音、播放的声音等。音量调节功能适用于多种需要自定义调节音量的场景。

除此之外，NERTC SDK 也支持通过监听回调的方式获取房间内成员的音量。在一些语音连麦场景或者视频会议场景中，产品应用层常常需要获取发言者的音量，并通过 UI 进行音量展示，或者根据发言者的音量大小进行视图布局的动态调整。详细信息请参考监测发言者音量。

5.3.2.1 设置采集音量

NERTC SDK 通过录音设备采集房间内成员的音频信号，将其录制下来以后播放给远端用户收听。您可以通过 adjustRecordingSignalVolume 设置录制声音的信号幅度，从而达到调节采集音量的目的。

该方法中 volume 参数表示录音信号的音量，取值范围为 0～400：

- 0：静音。
- 100：（默认值）原始音量，即不对信号做缩放。
- 400：原始音量的 4 倍（把信号放大到原始信号的 4 倍）。

建议设置采集音量时使用默认值（100）。

5.3.2.2 设置播放音量

在音视频通话过程中，音频信号从发送端进入到接收端，然后使用播放设备进行播放。您可以通过 adjustPlaybackSignalVolume 方法

或 adjustUserPlaybackSignalVolume 方法调节播放声音的信号幅度，从而达到调节播放音量的目的。

- 通过 adjustPlaybackSignalVolume 可以调节所有远端用户混音后的音频流在本地播放的音量，通过 volume 参数设置播放信号的音量，取值范围为 0～400：
  - 0：静音。
  - 100：（默认值）原始音量，即不对信号做缩放。
  - 400：原始音量的 4 倍，即把信号放大到原始信号的 4 倍。

建议设置本地播放音量时使用默认值（100）或小于该值，否则可能会导致音质问题。

- 通过 adjustUserPlaybackSignalVolume 可以调节指定远端用户混音后的音频流在本地播放的音量。
  - userID 用于指定远端用户，volume 参数用于设置播放信号的音量，取值范围为 0～100。
  - 多次调用该方法，可以设置不同远端用户在本地播放的音量；也可以反复调节某个远端用户在本地播放的音量。

5.3.2.3 设置耳返音量

通过 enableEarback 开启耳返功能之后，您可以通过 setEarbackVolume 设置耳返音量。耳返音量支持设置为 0~100，默认为 100，表示使用原始音量。

该方法中 volume 参数表示录音信号的音量，取值范围为 0～400：

- 0：静音。
- 100：（默认值）原始音量，即不对信号做缩放。
- 400：原始音量的 4 倍（把信号放大到原始信号的 4 倍）。

5.4 音频共享

5.4.1 Android

在屏幕分享或共享本地播放的音乐文件等场景中，用户常常需要将本地系统音频发送至远端。NERTC 提供了音频共享功能，帮助您在共享屏幕的同时也能播放本地背景音，或者共享本地视频文件或音乐文件的声音，为您规避播放在线音乐文件可能会遇到的版权问题。

5.4.1.1 功能介绍

通过 NERTC SDK 可以在视频通话或互动直播的过程中实现音频共享，主持人、主播或连麦者可以将本地播放的音频流分享给远端参会者或在线观众收听，从而提升视频通话体验。

音频共享适用于在线会议、在线教育以及互动直播等场景，具体场景说明如下。

- 视频会议场景中，参会者可以在会议中将本地的音频文件、PPT 背景音等所有系统声音分享给其他与会者，让其他与会者更加沉浸式地了解讨论的内容和主题。
- 在线课堂场景中，老师可以线上教学过程中在将课件、教学视频的背景音等所有系统声音分享给远端的学生，提升教育场景的用户体验。
- 互动直播场景中，主播可以在直播过程中将本地播放的音乐、视频背景音等所有系统声音分享给远端的观众，避免了因需要播放在线音乐而遇到的版权问题，同时丰富了场景体验。

5.4.1.2 注意事项

- Android 10 及以后的版本音频共享系统要求开启一个前台服务，因此您需要在 AndroidManifest.xml 中添加 service，同时将 compileSdkVersion 设置为 29。请根据您的业务需求添加 service。
- MediaProjection 等 API 需要 Android API level 21+，使用方法请参考 Google MediaProjection API 文档。
- 您不能同时使用音频自播放和音频共享功能。否则会导致加入音视频通话房间后，无法听到对端用户的音频和本地共享音频的声音。

5.4.1.3 配置步骤

1. 添加 ScreenShareService，具体请参考 Demo 或联系技术支持。
2. 通过 MediaProjection 创建 ScreenCaptureIntent 请求系统权限，并将 intent 传递给 startActivityForResult() 。
3. 在加入房间后调用 enableLoopbackRecording 开启音频共享。调用此方法时，您需要设置 enable 参数为 true 开启音频共享，传入请求权限后返回的 intent data，并设置 MediaProjection.Callback() 以接收音频共享状态回调。
4. 调用 adjustLoopBackRecordingSignalVolume 调整共享音频音量。调用该方法时，您需要设置 volume 参数指定采集信号量。该参数的取值范围为 0～100。
5. 需要结束音频共享时，调用 enableLoopBackRecording 并设置 enable 参数为 false 停止音频共享。

5.4.1.4 示例代码

```
if(open){

    //ScreenShareService 部分参考 demo 实现

    if (ScreenShareService.mediaProjectionIntent == null) {

        MediaProjectionManager mediaProjectionManager = null;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {

            mediaProjectionManager = (MediaProjectionManager)
mActivity.getSystemService(Context.MEDIA_PROJECTION_SERVICE);

            Intent permissionIntent =
mediaProjectionManager.createScreenCaptureIntent();

            startActivityForResult(permissionIntent,
LOOP_BACK_START_CODE);

        }

    }else{

        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {

            int nRet = NERtcEx.getInstance().enableLoopbackRecording(true,
ScreenShareService.mediaProjectionIntent, new MediaProjection.Callback() {
```

```
                @Override
                public void onStop() {
                    mUIHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        Log.w(TAG, "系统录音已停止");
                    }});
                    super.onStop();
            }});
            }
        }
}else {
    NERtcEx.getInstance().enableLoopbackRecording(false, null, null);
}
```

5.4.1.5 API 参考

方法	功能描述
enableLoopbackRecording	开启或关闭音频共享
adjustLoopBackRecordingSignalVolume	调整音频共享音量

5.5 屏幕共享

5.5.1 Android

在大型会议或在线教育等场景中，为了满足提升沟通效率的需求，主讲人或老师需要将本端的屏幕内容分享给远端参会者或在线学生观看。NERTC 支持屏幕共享功能，帮助您实时分享本端设备的屏幕内容。

5.5.1.1 功能介绍

通过 NERTC SDK 可以在视频通话或互动直播过程中实现屏幕共享，主播或连麦者可以将自己的屏幕内容，以视频的方式分享给远端参会者或在线观众观看，从而提升沟通效率，一般适用于多人视频聊天、在线会议以及在线教育场景。

- 视频会议场景中，参会者可以在会议中将本地的文件、数据、网页、PPT 等画面分享给其他与会者，让其他与会者更加直观的了解讨论的内容和主题。

- 在线课堂场景中，老师可以通过屏幕共享将课件、笔记、教学内容等画面展示给远端的其他学生观看，降低传统教学模式下的沟通成本，提升教育场景的用户体验。

  NERTC SDK 以辅流的形式实现屏幕共享，即单独为屏幕共享开启一路上行的视频流，摄像头的视频流作为主流，屏幕共享的视频流作为辅流，两路视频流并行，主播同时上行摄像头画面和屏幕画面两路画面。

5.5.1.2 注意事项

- NERTC Android、iOS、Windows 和 macOS SDK V3.9.0 及以上版本，Web SDK V4.1.0 及以上版本支持通过辅流实现屏幕共享。如果使用辅流的屏幕共享方案，请保证房间内所有成员均升级到支持版本以上，否则互相通信时会因同时发送主流和辅流造成通话异常等问题。
- 如果您的 App 无法针对所有端进行强制升级，屏幕共享场景中仅部分端使用 V3.9.0 及以上版本，为避免上述通话异常问题，必须保证通话过程中单人同时只有一路上行视频流。当需要将视频流切换为屏幕共享流时，请先通过 enableLocalVideo 关闭视频流，再通过 startScreenCapture 启动屏幕共享流。反向切换同理。
- 在开始屏幕共享前，请确保已在你的项目中实现基本的实时音视频功能。
- MediaProjection 等 API 需要 Android API level 21+，使用方法请参考 Google MediaProjection API 文档。
- **Android 10** 及以后的版本屏幕共享系统要求开启一个前台服务，因此需要在 `AndroidManifest.xml` 中添加 service，同时将 **compileSdkVersion** 设置为 29。请根据您的业务需求添加 service。

5.5.1.3 本端共享屏幕

配置步骤

1. 添加 ScreenShareService，具体请参考 Demo。
2. 通过 MediaProjection 创建 ScreenCaptureIntent 请求屏幕共享权限，并将 intent 传递给 startActivityForResult() 。
3. 在加入房间之后调用 startScreenCapture 方法开启屏幕共享，以辅流形式发送屏幕共享内容。调用此方法时，您需要设置 screenConfig 配置本地辅流的编码参数，传入请求权限后返回的 intent data，并设置 MediaProjection.Callback() 以接收屏幕共享状态回调。
4. 调用 setupLocalSubStreamVideoCanvas 方法设置本端的辅流视频画布。调用此接口时，您需要设置 render 参数配置视频画布。
5. 若您要结束屏幕共享，调用 stopScreenCapture 方法关闭辅流形式的屏幕共享。

示例代码

```
//先请求屏幕共享权限

@TargetApi(Build.VERSION_CODES.LOLLIPOP)

private void startScreenCapture() {

    MediaProjectionManager mediaProjectionManager =
```

```
                (MediaProjectionManager)
getApplication().getSystemService(
                        Context.MEDIA_PROJECTION_SERVICE);
        startActivityForResult(
                mediaProjectionManager.createScreenCaptureIntent(),
CAPTURE_PERMISSION_REQUEST_CODE);
    }

    //在权限请求返回中打开屏幕共享接口
    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode != CAPTURE_PERMISSION_REQUEST_CODE)
            return;
        if(resultCode != Activity.RESULT_OK) {
            showToast("你拒绝了录屏请求！");
```

```
                getUiKitButtons().find("screen_cast",
Boolean.class).setState(false);

                return;

            }

            NERtcScreenConfig screenProfile = new NERtcScreenConfig();

            screenProfile.videoProfile = mScreenProfile;

            screenProfile.contentPrefer = mScreenContent;

            screenProfile.frameRate = mScreenFps;

            screenProfile.minFramerate = mScreenMinFps;

            screenProfile.bitrate = mScreenEncodeBitrate;

            screenProfile.minBitrate = mScreenEncodeMinBitrate;

            mScreenService.startScreenCapture(screenProfile, data, new
MediaProjection.Callback() {

                @Override

                public void onStop() {

                    super.onStop();

                    showToast("录屏已停止");
```

```
            }

        });


NERtcEx.getInstance().setupLocalSubStreamVideoCanvas(mScreenView);

    }


    // 停止桌面共享

    NERtcEx.getInstance().stopScreenCapture();
```


5.5.1.4 观看远端屏幕共享

配置步骤

1. 远端用户加入房间。
2. 通过 setupRemoteSubStreamVideoCanvas 设置远端的辅流视频回放画布。
3. 收到 onUserSubStreamVideoStart 其他用户开启屏幕共享辅流通道的回调。
4. 通过 subscribeRemoteSubStreamVideo 订阅远端的屏幕共享辅流视频，订阅之后才能接收远端的辅流视频数据。
5. 收到 onUserSubStreamVideoStop 其他用户关闭辅流的回调，结束屏幕共享。

示例代码

```
public void onUserSubStreamVideoStart(long uid,int maxProfile) {
    Log.i(TAG, "onUserSubStreamVideoStart uid: " + uid);
    NERtcEx.getInstance().subscribeRemoteSubStreamVideo(
        user.userId, true);
    NERtcEx.getInstance().setupRemoteSubStreamVideoCanvas(view, uid);
}
```

5.5.2 iOS

通过 NERTC SDK 可以在视频通话或互动直播过程中实现屏幕共享，主播或连麦者可以将自己的屏幕内容，以视频的方式分享给远端参会者或在线观众观看，从而提升沟通效率，一般适用于多人视频聊天、在线会议以及在线教育场景。

- 视频会议场景中，参会者可以在会议中将本地的文件、数据、网页、PPT 等画面分享给其他与会者，让其他与会者更加直观的了解讨论的内容和主题。
- 在线课堂场景中，老师可以通过屏幕共享将课件、笔记、教学内容等画面展示给远端的其他学生观看，降低传统教学模式下的沟通成本，提升教育场景的用户体验。

NERTC SDK 以辅流的形式实现屏幕共享，即单独为屏幕共享开启一路上行的视频流，摄像头的视频流作为主流，屏幕共享的视频流作为辅流，两路视频流并行，主播同时上行摄像头画面和屏幕画面两路画面。

音频共享功能基于原先的 NERTC SDK 可以实现，需要您在应用开发中自行编码支持。如有相关需要，您可以联系技术支持，索取 iOS 相关 Demo 的 SampleCode，参考 Demo 在您的开发环境中完成编码。

5.5.2.1 实现流程

基于 iOS 系统的屏幕共享功能，需要在 App Extension 中通过 iOS 原生的 ReplayKit 特性实现录屏进程，并配合主 App 进程进行推流。需要进行屏幕共享的时候，使用 Apple ReplayKit 框架进行屏幕录制，接收系统采集的屏幕图像，并将其发送给 SDK 以传输视频流数据。

屏幕共享的主要流程包括：

1. （可选）创建 App Group。 App Group 用于在主 App 进程和扩展程序之间之间进行视频数据和控制指令的传输。
2. 通过 Xcode 在工程中创建一个 Target，类型为 Broadcast Upload Extension， 用于开启屏幕共享的进程。
3. 添加 ReplayKit 扩展，并使用 Apple ReplayKit 框架进行屏幕录制。
4. 将录屏数据作为自定义视频源发送给 SDK，并使用 SDK 进行视频流的传输。

5.5.2.2 注意事项

- NERTC Android、iOS、Windows 和 macOS SDK V3.9.0 及以上版本，Web SDK V4.1.0 及以上版本支持通过辅流实现屏幕共享。如果使用辅流的屏幕共享方案，请

保证房间内所有成员均升级到支持版本以上，否则互相通信时会因同时发送主流和辅流造成通话异常等问题。

- 如果您的 App 无法针对所有端进行强制升级，屏幕共享场景中仅部分端使用 V3.9.0 及以上版本，为避免上述通话异常问题，必须保证通话过程中单人同时只有一路上行视频流。当需要将视频流切换为屏幕共享流时，请先通过 enableLocalVideo 关闭视频流，再通过 startScreenCapture 启动屏幕共享流。反向切换同理。
- 在开始屏幕共享前，请确保已在你的项目中实现基本的实时音视频功能。
- ReplayKit 仅支持 iOS 11.0 以上共享系统屏幕。Sample 工程支持 iOS 12 及以上唤起系统录屏能力，若系统版本低于 iOS 12，需手动唤起系统录屏。
- 主 App 和系统录屏需使用相同的 App Group 名。

5.5.2.3 添加 ReplayKit

步骤一（可选）创建 App Group

1. 在 Certificates, Identifiers & Profiles 页面中注册 App Group。

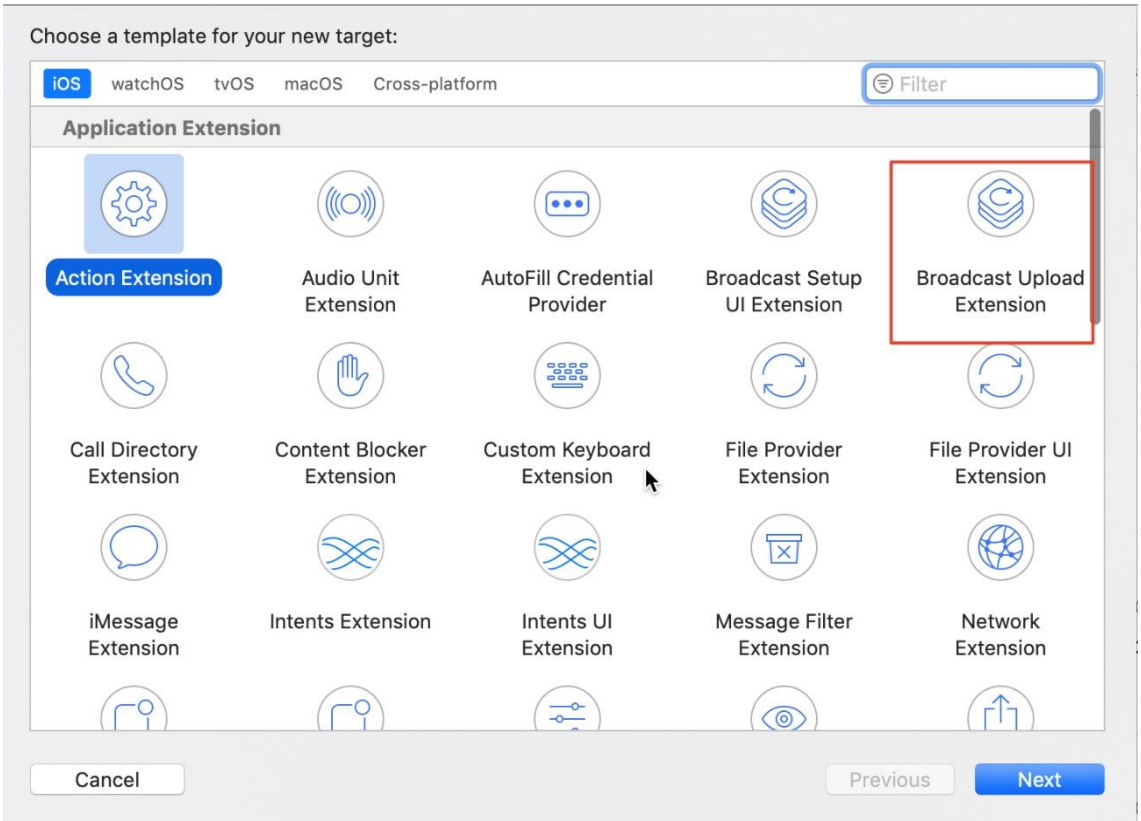
   操作步骤请参考注册 App Group。
2. 为您的 App ID 启用 App Group 功能。

   操作步骤请参考启用 App Group。
3. 重新下载 Provisioning Profile 并配置到 XCode 中。

步骤二 创建 Extension 录屏进程

创建一个类型为 Broadcast Upload Extension 的 Target，用于存放屏幕共享功能的实现代码。

1. 在 Xcode 中打开项目的工程文件。
2. 在菜单中选择 **Editor > Add Target...**。
3. 在 **iOS** 页签中 选择 **Broadcast Upload Extension**，并单击 **Next**。



4. 在 **Product Name** 中为 Extension 命名，单后单击 **Finish**。

步骤三 创建 App Group 数据池

数据池用于扩展 ReplayKit 程序和主工程之间通信。

```
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject
*> *)setupInfo {

    self.userDefaults = [[NSUserDefaults alloc]
initWithSuiteName:<#kAppGroupName#>];

}
```


步骤四 通过 ReplayKit 实现屏幕共享

压缩裁剪采集图片，发送到宿主 App，并通过 ReplayKit 实现屏幕共享。

1. 采集到的屏幕视频数据通过 processSampleBuffer:withType: 给用户。使用云信 SDK 音频采集，忽略音频数据回调。
2. 将视频数据压缩后存入共享内存。
3. 主程序监测到视频数据变更后，通过 SDK 自定义视频数据进行发送。

```
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer
withType:(RPSampleBufferType)sampleBufferType {



    switch (sampleBufferType) {

        case RPSampleBufferTypeVideo: {

            @autoreleasepool {
```

```
                CVImageBufferRef pixelBuffer =
CMSampleBufferGetImageBuffer(sampleBuffer);

                NSDictionary *frame = [self
createI420VideoFrameFromPixelBuffer:pixelBuffer];

                [self.userDefautls setObject:frame forKey:<#KeyPath#>];

                [self.userDefautls synchronize];

            }

            break;

        }

        case RPSampleBufferTypeAudioApp:

            // Handle audio sample buffer for app audiobreak;

        case RPSampleBufferTypeAudioMic:

            // Handle audio sample buffer for mic audiobreak;



        default:

            break;

    }
```

```
}
```


数据压缩采用的是 libyuv 第三方工具。

```
- (NSDictionary
*)createI420VideoFrameFromPixelBuffer:(CVPixelBufferRef)pixelBuffer
{
    CVPixelBufferLockBaseAddress(pixelBuffer, 0);

    // 转 I420
    int psrc_w = (int)CVPixelBufferGetWidth(pixelBuffer);
    int psrc_h = (int)CVPixelBufferGetHeight(pixelBuffer);
    uint8 *src_y = (uint8 *)CVPixelBufferGetBaseAddressOfPlane(pixelBuffer,
0);
    uint8 *src_uv = (uint8
*)CVPixelBufferGetBaseAddressOfPlane(pixelBuffer, 1);
    int y_stride = (int)CVPixelBufferGetBytesPerRowOfPlane(pixelBuffer, 0);
    int uv_stride = (int)CVPixelBufferGetBytesPerRowOfPlane(pixelBuffer, 1);
```

```
uint8 *i420_buf = (uint8 *)malloc((psrc_w * psrc_h * 3) >> 1);



libyuv::NV12ToI420(&src_y[0],                          y_stride,
                   &src_uv[0],                         uv_stride,
                   &i420_buf[0],                       psrc_w,
                   &i420_buf[psrc_w * psrc_h],         psrc_w >> 1,
                   &i420_buf[(psrc_w * psrc_h * 5) >> 2],  psrc_w >> 1,
                   psrc_w, psrc_h);



// 缩放至 720
int pdst_w = 720;
int pdst_h = psrc_h * (pdst_w/(double)psrc_w);
libyuv::FilterMode filter = libyuv::kFilterNone;
uint8 *pdst_buf = (uint8 *)malloc((pdst_w * pdst_h * 3) >> 1);
libyuv::I420Scale(&i420_buf[0],                        psrc_w,
                  &i420_buf[psrc_w * psrc_h],          psrc_w >> 1,
```

```
                    &i420_buf[(psrc_w * psrc_h * 5) >> 2], psrc_w >> 1,
                    psrc_w, psrc_h,
                    &pdst_buf[0],                          pdst_w,
                    &pdst_buf[pdst_w * pdst_h],            pdst_w >> 1,
                    &pdst_buf[(pdst_w * pdst_h * 5) >> 2], pdst_w >> 1,
                    pdst_w, pdst_h,
                    filter);


    free(i420_buf);


    CVPixelBufferUnlockBaseAddress(pixelBuffer, 0);


    NSUInteger dataLength = pdst_w * pdst_h * 3 >> 1;
    NSData *data = [NSData dataWithBytesNoCopy:pdst_buf
length:dataLength];
```

```
    NSDictionary *frame = @{
        @"width": @(pdst_w),
        @"height": @(pdst_h),
        @"data": data,
        @"timestamp": @(CACurrentMediaTime() * 1000)
    };
    return frame;
}
```


5.5.2.4 屏幕分享主程序

1. 初始化 SDK，配置允许使用外部视频源，确保视频通话功能正常。

```
//开启外部视频源，并将外部视频源配置为屏幕共享
NERtcEngine *coreEngine = [NERtcEngine sharedEngine];
[coreEngine enableLocalAudio:YES];
[[[NTESDemoLogic sharedLogic] getCoreEngine]
setExternalVideoSource:YES isScreen:YES];
```

```
NERtcEngineContext *context = [[NERtcEngineContext alloc] init];

context.engineDelegate = self;

context.appKey = <#请输入您的 AppKey#>;

[coreEngine setupEngineWithContext:context];
```


2. 在 RPSystemBroadcastPickerView 中添加扩展程序。

```
- (void)addSystemBroadcastPickerIfPossible

{

    if (@available(iOS 12.0, *)) {

        // Not recommend

        RPSystemBroadcastPickerView *picker =
[[RPSystemBroadcastPickerView alloc] initWithFrame:CGRectMake(0, 0, 120,
64)];

        picker.showsMicrophoneButton = NO;

        picker.preferredExtension = <#扩展程序的 BundleId#>;

        [self.view addSubview:picker];

        picker.center = self.view.center;
```

```
        UIButton *button = [picker.subviews
filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id
_Nullable evaluatedObject, NSDictionary<NSString *,id> * _Nullable bindings) {
            return [evaluatedObject isKindOfClass:UIButton.class];
        }]].firstObject;
        [button setImage:nil forState:UIControlStateNormal];
        [button setTitle:@"Start Share" forState:UIControlStateNormal];
        [button setTitleColor:self.navigationController.navigationBar.tintColor
forState:UIControlStateNormal];

        UIBarButtonItem *leftItem = [[UIBarButtonItem alloc]
initWithCustomView:picker];
        self.navigationItem.leftBarButtonItem = leftItem;
    }
}
```

3. 添加监听事件。

```
- (void)setupUserDefaults

{

    // 通过 UserDefaults 建立数据通道，接收 Extension 传递来的视频帧

self.userDefaults = [[NSUserDefaults alloc]

initWithSuiteName:<#AppGroupName#>];

    [self.userDefaults addObserver:self forKeyPath:<#KeyPath#>

options:NSKeyValueObservingOptionNew context:KVOContext];

}
```

4. 监听到数据帧变化，校验后推送外部视频帧到 SDK。

```
- (void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object

change:(NSDictionary<NSKeyValueChangeKey,id> *)change context:(void

*)context

{

    if ([keyPath isEqualToString:<#KeyPath#>]) {

        if (self.currentUserID) {
```

```
            NSDictionary *i420Frame = change[NSKeyValueChangeNewKey];

            NERtcVideoFrame *frame = [[NERtcVideoFrame alloc] init];

            frame.format = kNERtcVideoFormatI420;

            frame.width = [i420Frame[@"width"] unsignedIntValue];

            frame.height = [i420Frame[@"height"] unsignedIntValue];

            frame.buffer = (void *)[i420Frame[@"data"] bytes];

            frame.timestamp = [i420Frame[@"timestamp"]
unsignedLongLongValue];

            int ret = [NERtcEngine.sharedEngine
pushExternalVideoFrame:frame]; //  推送外部视频帧到 SDKif (ret != 0) {

                NSLog(@"发送视频流失败:%d", ret);

                return;

            }

        }

    }

}
```

5. 设置视频回放画布，并开启屏幕共享。屏幕共享内容以辅流形式发送。

i.通过 setupLocalSubStreamVideoCanvas 设置本端的辅流视频画布。

ii. 加入房间后，通过 startScreenCapture 开启屏幕共享，屏幕共享内容以辅流形式发送。

iii. 若有需要，可以通过 setLocalRenderSubStreamScaleMode 设置本端的辅流渲染缩放模式。

```
//设置本端的辅流视频画布

NERtcVideoCanvas *subStreamCanvas = nil;

if([NTESDemoSettings
boolForKey:keyNRTCDemoLocalSubStreamExternalRender])

{

    NTESExternalRenderView *externalview = [[NTESExternalRenderView
alloc] initWithFrame:CGRectZero format:SDL_FCC_I420];

    subStreamCanvas = [NERtcVideoCanvas
localCanvasWithExternalRender:externalview];

    [NTESDemoLogic sharedLogic].userManager.me.screenRenderView =
externalview;

 }else{

    UIView *view = [[UIView alloc] initWithFrame:CGRectZero];
```

```
    subStreamCanvas = [NERtcVideoCanvas
localSubStreamCanvasWithView:view];

    [NTESDemoLogic sharedLogic].userManager.me.screenRenderView =
(NTESExternalRenderView *)view;

 }

 [[[NTESDemoLogic sharedLogic] getCoreEngine]
setupLocalSubStreamVideoCanvas:subStreamCanvas];




//设置本端的辅流渲染缩放模式

NSString *key = keyNRTCDemoLocalSubStreamRenderScaleMode;

if (settings[key]) {

    NERtcVideoRenderScaleMode renderMode =
(NERtcVideoRenderScaleMode)[settings jsonInteger:key];

    [[[NTESDemoLogic sharedLogic] getCoreEngine]
setLocalRenderSubStreamScaleMode:renderMode];

}
```

```
//屏幕共享开启和关闭

- (void)onMenuMySubStream:(id)sender {

    NTESUser *me = findMe();

    int result = 0;

    BOOL toStart = !me.screenConnected;

    if (toStart) {

        NERtcVideoSubStreamEncodeConfiguration *config =
[[NERtcVideoSubStreamEncodeConfiguration alloc] init];

        if([NTESDemoSettings
objectForKey:keyNRTCDemoLocalVideoSubStreamProfileType]) {

            NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalVideoSubStreamProfileType];

            config.maxProfile = (NERtcVideoProfileType)value;

        }
```

```
        if ([NTESDemoSettings
objectForKey:keyNRTCDemoLocalSubStreamEncodeFrameRate]) {
            NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalSubStreamEncodeFrameRate];
            config.frameRate = value;
        }

        if ([NTESDemoSettings
objectForKey:keyNRTCDemoLocalSubStreamEncodeMinFrameRate]) {
            NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalSubStreamEncodeMinFrameRate];
            config.minFrameRate = value;
        }

        if ([NTESDemoSettings
objectForKey:keyNRTCDemoLocalSubStreamEncodeBitrate]) {
            NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalSubStreamEncodeBitrate];
```

```
        config.bitrate = value;
    }

    if ([NTESDemoSettings
objectForKey:keyNRTCDemoLocalSubStreamEncodeMinBitrate]) {
        NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalSubStreamEncodeMinBitrate];
        config.minBitrate = value;
    }

    if ([NTESDemoSettings
objectForKey:keyNRTCDemoLocalSubStreamEncodeContentPrefer]) {
        NSInteger value = [NTESDemoSettings
integerForKey:keyNRTCDemoLocalSubStreamEncodeContentPrefer];
        config.contentPrefer = value;
    }
```

```
        //屏幕共享开启

        result = [[[NTESDemoLogic sharedLogic] getCoreEngine]
startScreenCapture:config];

    }else{

        //屏幕共享关闭

        result = [[[NTESDemoLogic sharedLogic] getCoreEngine]
stopScreenCapture];

    }

    NTESCheckResultAndReturn(result, nil);

    me.screenConnected = toStart;

    if (self.eventDelegate && [self.eventDelegate
respondsToSelector:@selector(handlerEventSubStreamStart:)]) {

        [self.eventDelegate
handlerEventSubStreamStart:me.screenConnected];

    }

}
```

6. 不使用该功能时，需要移除观察者，并关闭屏幕共享。

```
[self.userDefaults removeObserver:self forKeyPath:<#KeyPath#>];

[[[NTESDemoLogic sharedLogic] getCoreEngine] stopScreenCapture];
```


5.5.2.5 观看远端屏幕共享

1. 远端用户加入房间。
2. 通过 setupRemoteSubStreamVideoCanvas|设置远端的辅流视频回放画布 设置远端的辅流视频回放画布。
3. 收到 onNERtcEngineUserSubStreamDidStartWithUserID 其他用户开启屏幕共享辅流通道的回调。
4. 通过 subscribeRemoteSubStreamVideo 订阅或取消订阅远端的屏幕共享辅流视频，订阅之后才能接收远端的辅流视频数据。
5. 管理屏幕共享任务。

- 通过 setRemoteRenderSubStreamVideoScaleMode 设置远端的屏幕共享辅流视频渲染缩放模式。
- 通过 subscribeRemoteSubStreamVideo 取消订阅远端的屏幕共享辅流视频。

6. 收到 onNERtcEngineUserSubStreamDidStop 其他用户关闭辅流的回调，结束屏幕共享。

示例代码：

```
//其他用户开启屏幕共享辅流通道的回调

- (void)onNERtcEngineUserSubStreamDidStartWithUserID:(uint64_t)userID
subStreamProfile:(NERtcVideoProfileType)profile {

    NTESUser *user = [[NTESDemoLogic sharedLogic].userManager
userWithID:userID];

    if (!user || user.isMe) {

        return;

    }



    //设置远端的辅流视频回放画布

    NERtcVideoCanvas *subCanvas = nil;

    if([NTESDemoSettings
boolForKey:keyNRTCDemoRemoteSubStreamExternalRender]){

        NTESExternalRenderView *externalview =
[[NTESExternalRenderView alloc] initWithFrame:CGRectZero
format:SDL_FCC_I420];

        subCanvas = [NERtcVideoCanvas
remoteCanvasWithExternalRender:externalview];
```

```
        user.screenRenderView = externalview;

    }

    else{

        VIEW_CLASS *view = [[VIEW_CLASS alloc]
initWithFrame:CGRectZero];

        subCanvas = [NERtcVideoCanvas
remoteSubStreamCanvasWithView:view];

        user.screenRenderView = (NTESExternalRenderView *)view;

    }

    [[[NTESDemoLogic sharedLogic] getCoreEngine]
setupRemoteSubStreamVideoCanvas:subCanvas forUserID:userID];




    VideoUserCell *cell = [self.mainView cellForUserID:userID];

    cell.screenRenderView = (UIView *)user.screenRenderView;
```

```
    if ([NTESDemoSettings
boolForKey:keyNRTCDemoChanelEnableMeetingScene defaultVal:NO] &&
[self.mainView cellForUserID:userID]) {
        [self subscribeSubStreamWithUserID:userID];
        return;
    }

    // screen start 之前就操作过订阅了
    if (user.isScreenSubscribed) {
        return;
    }

    BOOL autoSubscribe = [NTESDemoSettings
boolForKey:keyNRTCDemoAutoSubscribeRemoteSubStream
defaultVal:YES];
    if (autoSubscribe && [self.mainView cellForUserID:userID]) {
        //订阅远端的屏幕共享辅流视频
```

```
        [self subscribeSubStreamWithUserID:userID];

    }

}



//其他用户关闭辅流的回调

- (void)onNERtcEngineUserSubStreamDidStop:(uint64_t)userID {

    NTESUser *user = [[NTESDemoLogic sharedLogic].userManager
userWithID:userID];

    if (!user || user.isMe) {

        return;

    }



    VideoUserCell *cell = [self.mainView cellForUserID:userID];

    cell.screenRenderView = nil;
```

```
    //SubStream

    NERtcVideoCanvas *subCanvas = [NERtcVideoCanvas
remoteSubStreamCanvasWithView:nil];

    [[[NTESDemoLogic sharedLogic] getCoreEngine]
setupRemoteSubStreamVideoCanvas:subCanvas forUserID:userID];

}




//设置远端的屏幕共享辅流视频渲染缩放模式

NSString *key = keyNRTCDemoRemoteSubStreamRenderScaleMode;

if (settings[key]) {

    NERtcVideoRenderScaleMode renderMode =
(NERtcVideoRenderScaleMode)[settings jsonInteger:key];

    NSArray *users = [NTESDemoLogic sharedLogic].userManager.users;

    for (NTESUser *user in users) {

        if (user.userID != [NTESUser selfID]) {
```

```
            [[[NTESDemoLogic sharedLogic] getCoreEngine]
setRemoteRenderSubStreamVideoScaleMode:renderMode
forUserID:user.userID];

        }

    }

}
```

六、 进阶功能

更多功能的操作指引请参见音视频通话 2.0 开发指南。