ALIBABA CLOUD

阿里云

视频直播

推流SDK

文档版本：20211214

阿里云

法律声明

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

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

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

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

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

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

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

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

目录

1.产品介绍 05
2.基础概念 09
3.SDK下载与历史记录 10
4.Android推流SDK 13
4.1. V4.0.2升级至V4.1.0及以上迁移说明 13
4.2. Demo编译 16
4.3. SDK集成 23
4.4. 功能使用 25
4.5. 异常及特殊场景处理 43
5.iOS推流SDK 46
5.1. V4.0.2升级至V4.1.0及以上迁移说明 46
5.2. Demo编译 49
5.3. SDK集成 52
5.4. 功能使用 58
5.5. 异常及特殊场景处理 74
6.RTS推流（WebRTC推流）使用流程 79

1.产品介绍

本文介绍推流SDK产品简介、功能特性、核心优势、使用场景和工作流程。

产品简介

阿里云推流SDK是基于阿里云强大内容分发网络和音视频实时通讯技术的直播客户端推流开发工具，为您提供简单易用的开放接口、网络自适应的流畅体验、多节点的低延迟优化、功能强大的实时美颜等音视频直播技术服务。SDK为免费提供，让您告别复杂的架构设计，降低维护成本，专注于自身业务逻辑实现和用户体验的提升。

阿里云推流SDK集成了Queen智能美化特效，开放含人脸识别的高级美颜功能，可实现瘦脸、小脸、大眼以及基于人脸识别的美白功能。

> 说明
>
> 集成Queen智能美化特效功能，需要申请开通License。详细产品介绍与开通流程，请参见智能美化特效。

功能特性

功能	描述
RTMP推流	支持RTMP协议直播推流，并支持RTMP、FLV、HLS、ARTC直播拉流协议。分辨率支持180P~720P，建议使用540P。
WebRTC推流	支持基于UDP的ARTC协议推流。
录屏直播	iOS支持replayKit录屏直播，Android支持摄像头混流录屏直播。 支持窄带高清，同等码率下画质更佳。
插入SEI信息	支持在直播流中插入SEI信息，通过播放器解析SEI配合您的业务实现多种功能。
动态水印	支持在直播中实时插入或移除带动画效果的水印。
外部音视频推流	支持输入外部音视频数据流进行直播。
后台推图片	支持在切后台时设置图片进行推流，同时也支持在网络非常差的情况下替换为图片推流。
音视频编码	支持H264视频编码（软编和硬编）和支持AAC音频编码（软编和硬编）。
实时美颜	支持人脸识别高级美颜，包含磨皮、美白、瘦脸、小脸、大眼等功能。

功能	描述
动态码率	支持根据网络情况自动调整推流码率，支持多种模式设置，使直播更加流畅。
动态分辨率	支持根据网络情况自动调整推流分辨率（限清晰度和流畅度模式下使用）。
后台推流	支持退到后台后视频流不断，回到前台后继续推流。
立体声推流	支持立体声推流，可设置单声道和双声道推流。
多水印	支持添加多个水印效果（最多3个），水印支持位置和大小设置。
横屏推流	支持竖向（portrait）、左侧横向（landscape left）和右侧横向（landscape right）三个方向发起推流。
采集参数	支持分辨率、帧率、音频采样率、GOP、码率等多种采集参数设置，满足不同场景下画面采集的需求。
镜像推流	支持摄像头采集镜像和推流镜像分别设置，前置摄像头需默认开启镜像功能。
纯音频推流	支持仅采集音频流并发起推流功能，在纯音频场景下节约带宽流量。
静音推流	支持推流时关闭麦克风，仅推送视频画面的功能。
自动聚焦	支持开启或关闭自动对焦功能，也可以使用手动对焦。
镜头缩放	支持摄像头支持的最大缩放比例进行采集画面的缩放。
摄像头切换和闪光灯	支持前置和后置摄像头切换和开启或关闭闪关灯功能（仅后置）。
背景音乐	支持背景音乐播放，包含开始、停止、暂停、继续、循环播放等功能。
混音	支持音乐和人声混音，分别调整音乐和人声的音量。
耳返	支持耳返功能，例如主播带上耳机唱歌时，从耳机中可以实时听到自己的声音，满足KTV的场景。

功能	描述
降噪	支持环境音、手机干扰等引起的噪音降噪处理。

核心优势

- **简单、易集成**

  Android和iOS提供统一接口和错误码，提供同步和异步接口，满足不同开发架构的接入需求，完善接口文档和Demo方便您参考。

- **一体化解决方案**

  提供从视频采集、渲染、推流、转码、分发到播放的一体化视频直播解决方案，端上的自适应码率推流、云端的窄带高清转码到播放端的首屏秒开完美配合，让您享受一站式优质服务。

- **高性能、低延时**

  在推流的卡顿率、CPU和内存消耗、耗电量、发热量等方面都处于业内领先水平，全球2500+的直播节点为各区域的低延时提供了有效保障。

- **WebRTC推流**

  提供基于UDP协议的WebRTC推流，且控制台支持自助开通WebRTC推流域名。WebRTC推流在上行网络质量较差时有更优秀的抗卡顿能力。

使用场景

场景一：教育直播

- **场景描述**：教育类直播通常对师生的互动比较注重，在接入直播课堂时可以配合阿里云的消息服务来完成师生之间的文字实时互动。推流SDK可以让教师随时随地为学生解惑答疑，让问题及时有效地解决。同时，配合云端的录制、转码等功能，学生可以随时回看课程，温习知识点，增强学习效果。
- **使用说明**：开通阿里云直播服务并启用录制和转码功能，接入阿里云推流SDK和消息服务SDK完成直播课堂或答疑服务，在播放端使用阿里云播放器SDK进行观看直播或回放视频课程，即可享受低延时、高互动的教育直播场景。

场景二：娱乐直播

- **场景描述**：娱乐直播借助手机的便利性形成了全民直播的风暴，主播对美颜、滤镜依赖成了绝对刚需，通过实时聊天、点赞和打赏等行为完成主播与观众的互动，提高人气值和可玩性。另外，手机端的娱乐直播门槛相对较低，对应内容的安全性（如涉黄、暴恐等）需要严格把关，这里可以借助直播鉴黄功能来降低审核成本。
- **使用说明**：开通阿里云直播服务并开启录制、鉴黄功能，接入阿里云推流SDK并开启美颜功能完成视频推流，集成阿里云消息服务于您的互动聊天场景，用户观看直播时可以在聊天面板里发送文字、图片等信息，也可以用来搭建自己的礼物系统（IM配合支付），在播放端使用阿里云播放器SDK进行观看直播或回放。

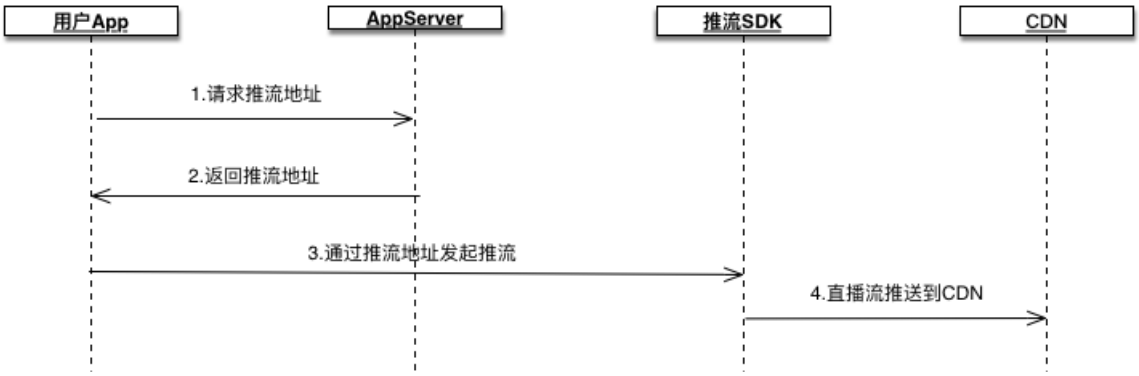
场景三：游戏直播

- **场景描述**：手机端游戏直播主要通过录屏技术将当前游戏画面和摄像头采集画面合并后一并通过推流SDK发起推流，因此推流SDK需要支持录屏功能。主播与观众的互动基本与娱乐直播类似，可以通过阿里云消息服务SDK实现聊天、点赞和打赏等交互行为。对于游戏中的精彩内容回放，可以使用直播回放录制服务。

- **使用说明**：开通阿里云直播服务并开启直播录制服务，接入阿里云推流SDK并使用录屏直播功能，集成阿里云消息服务于您的互动聊天场景，用户观看直播时可以在聊天面板里发送文字、图片等信息，也可以用来搭建自己的礼物系统（IM配合支付），集成阿里云播放器SDK完成极速秒开和动态追帧，观看直播或精彩回放。

SDK工作流程

1. 用户App向AppServer发起请求，获取推流URL。
2. AppServer根据规则拼接推流URL返回给App。
3. App赋值推流URL到推流SDK，使用推流SDK发起推流。
4. 推流SDK将直播流推送到CDN。



开发支持

如果您在使用推流SDK有任何问题或建议，欢迎通过钉钉搜索群号32825314加入推流SDK开发者生态群，或扫描以下二维码加入。

2.基础概念

通过阅读本文，您可以了解推流SDK中一些专业术语的基本概念。

- 码率控制：一种编码的优化算法，用于控制视频流码流的大小。同样的视频编码格式，码流越大，包含的信息越多，对应的图像也就越清晰，反之亦然。
- 视频丢帧：发送视频帧时，如果网络非常差，导致视频帧堆积严重，可以通过丢弃视频帧来缩短推流的延时。
- 耳返：指主播可以通过耳机实时听到自己的声音。例如，当主播带上耳机唱歌时，需要把握音调，这时就需要开启耳返功能。因为声音通过网络传入耳朵和通过空气传入耳朵差异很大，而主播需要直接听到观众端的效果。
- 混音：把多种来源的声音整合至一个立体音轨或单音音轨中，推流SDK支持音乐和人声的混音。
- 合流：把多种来源的视频图像数据根据位置叠加到同一个视频画面中。目前仅Android推流SDK支持。
- 动态库：即动态链接库，与常用的静态库相反。动态库在编译时并不会被拷贝到目标程序中，目标程序中只会存储指向动态库的引用。在程序运行时，动态库才会被真正加载进来。

> **说明** Xcode加载动态库需要加载到Embedded Binaries中，而不是加载到Linked Frameworks and Libraries中。

3.SDK下载与历史记录

本文提供推流SDK和Demo的下载链接，并介绍了推流SDK的发布历史。

Demo安装包

请扫描下方二维码下载最新版SDK的Demo。

发布历史

> **说明**
>
> - 自2021年7月起，阿里云视频直播服务将逐步停止对推流SDK V4.0.2的支持与维护，请使用最新版本的SDK。
> - 如需从推流SDK V4.0.2升级至最新版本，请参考Android端迁移说明或iOS端迁移说明，完成升级。

日期	版本	发布说明	下载地址
2021-12-02	V4.3.1	推流SDK更新，如下所示： iOS和Android端： • RTMP和RTS硬件编码支持H265。 • RTMP硬件编码支持B帧。 • 分辨率支持1080P。 • 硬件编码重置。 • BugFix：推流中偶现无音频修复。 iOS端：录屏推流支持暂停。	• Android SDK • iOS SDK • Android Demo源码 • iOS Demo源码

日期	版本	发布说明	下载地址

2021-11-01	V4.3.0	优化推流SDK，如下所示：
		• iOS和Android端：
		  ○ 接入音频3A算法。
		  ○ 音频外置输入推流模块重构。
		  ○ 音频播放与采集重构优化。
		  ○ 音频处理Audio_Process模块重构。
		• iOS端：
		  ○ 音乐场景音质提升。
		  ○ 录屏推流重构。
			• Android SDK
			• iOS SDK
			• Android Demo源码
			• iOS Demo源码

2021-09-10	V4.2.1	推流SDK更新，如下所示： iOS和Android端： • 底层架构重构：性能优化和包大小优化。 • 修复部分Bug。	• Android SDK • iOS SDK • Android Demo源码 • iOS Demo源码

2021-08-18	V4.2.0	优化推流SDK，如下所示：
		• iOS和Android端：
		  ○ 录屏推流支持窄带高清功能，同等码率下画质更佳。
		  ○ 增加实时状态数据回调，以便数据统计时监测直播质量。
		  ○ 支持自定义音视频采集。
		  ○ 去除部分依赖，缩减包大小：包括去除原有内置美颜依赖、播放器依赖和ARTP模块。
		  ○ 完善埋点，提供日志接口。
		• iOS端：增加Pod依赖。
		• Android端：升级OpenH264软编库，解决与短视频SDK冲突问题。
			• Android SDK
			• iOS SDK
			• Android Demo源码
			• iOS Demo源码

2021-06-25	V4.1.0	推流SDK全新优化，主要包括： • 支持RTS推流。 • 集成智能美化特效Queen SDK。 • Demo UI全新设计。 • 升级播放器SDK。	• Android SDK • iOS SDK • Android Demo源码 • iOS Demo源码

日期	版本	发布说明	下载地址
2021-01-13	V4.0.2	主要功能如下所示： • 摄像头推流：支持RTMP协议推流和WebRTC直播连麦推流。 • 直播观看：支持拉取RTMP、FLV、HLS和超低延时RTS的直播流进行观看。 • 支持采集、编码和推流参数设置。 • 支持添加背景音乐和音效。 • 支持美颜。 • 支持连麦、PK。（连麦、PK已停止开通）	• Android SDK • iOS SDK • Android Demo源码 • iOS Demo源码
2020-12-01	V3.6.1	修复弱网时偶现的发送音视频时间戳相差较大问题 。	• Android SDK和Demo • iOS SDK和Demo

V4.3.1版本推流SDK包的大小说明如下所示：

版本	平台	包大小	安装包增量
V4.3.1	Android	4.8MB	4.6MB
	iOS	• 7.05MB（arm64） • 6.32MB（armv7）	2.5（arm64）

4.Android推流SDK

4.1. V4.0.2升级至V4.1.0及以上迁移说明

如果您现在使用的是V4.0.2版本的Android推流SDK，想要升级到*V4.1.0*及以上版本时，可参考本文步骤进行升级。

前提条件

请下载最新版本的推流SDK。

新版推流SDK，请参见SDK下载与历史记录。

升级步骤

从项目中移除SDK V4.0.2相关类库和资源文件，添加*V4.1.0*及以上版本相关类库和资源文件，排查并更新相关API和推流主流程接口。

1. 在工程libs目录下：用*V4.1.0*及以上版本项目中的*AlivcLivePusher.aar*替换V4.0.2项目中的*AliLiveSdk.aar*。
2. 在*build.gradle*项目文件下，将播放器SDK升级到最新版本。

   > **说明** 播放器SDK发布记录，请参见Android播放器SDK。

   以2021年8月发布的播放器版本为例，修改播放器相关配置如下：

   ```
   implementation 'com.aliyun.sdk.android:AliyunPlayer:5.4.2.0-full'
   implementation 'com.aliyun.sdk.android:AlivcArtc:5.4.2.0'
   ```

3. 如果接入Queen智能美化特效，可以工程增加Demo中的beauty、beautyui、imageutil、queenbeauty model，方便美颜SDK及UI的接入。
4. 相关API会有部分调整，请根据核心接口对比排查并更新当前代码中的API。
5. 推流主流程接口有部分变更，请根据推流主流程接口变更修改当前代码。

核心接口对比

- 基础接口

V4.0.2	*V4.1.0*及以上版本	描述
getSdkVersion	getSdkVersion	获取版本号。
create	init	创建推流实例。
destroy	destroy	销毁推流。
setStatusCallback	setLivePushInfoListener	设置推流状态相关回调。
setNetworkCallback	setLivePushNetworkListener	设置推流网络状态相关回调。

V4.0.2	*V4.1.0*及以上版本	描述
setLogDirPath	setLogDirPath	设置SDK日志文件保存路径。 如需调用，请在调用所有API之前先调用此接口，避免日志出现丢失，同时保证指定的目录已存在且可写入。
setLogLevel	setLogLevel	设置日志输出级别。

- 推流基础接口

V4.0.2	*V4.1.0*及以上版本	描述
startPreview	startPreview	开始预览（主播端接口）。
stopPreview	stopPreview	停止预览（主播端接口）。
pausePush	pause	暂停摄像头采集并进入垫片推流状态（仅支持RTMP模式推流）。需要先调用startPush后才可以调用pausePush，否则调用顺序会出错。
resumePush	resume()	恢复摄像头采集并结束垫片推流状态（仅支持RTMP模式推流）。需要先调用pausePush后才可以调用resumePush，否则调用顺序会出错。
startPush	startPush	开始推流。
stopPush	stopPush	停止推流。
isPublishing	isPublishing	查询是否正在推流。
getPublishUrl	getPushUrl	获取当前推流的地址。

- 视频相关接口

V4.0.2	*V4.1.0*及以上版本	描述
setPreviewMode	setPreviewMode	设置预览模式。
isAudioOnly	isAudioOnly	查询是否纯音频推流。
switchCamera	switchCamera	切换前后摄像头。
setCameraZoom	setCameraZoom	设置摄像头缩放及是否允许闪光灯。
isCameraExposurePointSupported	setExposureCompensation	摄像头是否支持设置曝光区域。
setCameraFocusPoint	setLiveCameraFocus	设置摄像头聚焦。

- 音频相关接口

V4.0.2	*V4.1.0*及以上版本	描述
setMute	setMute	设置本地音频采集是否为静音帧。
isAudioOnly	isAudioOnly	查询是否纯音频推流。
enableEarBack	setBGMEarsBack	启用耳返。建议在插入耳机后开启耳返，否则可能会引入回声。
playBGM	startBGMAsync	播放背景音乐。
stopBGM	stopBGM	停止播放背景音乐。
pauseBGM	pauseBGM	暂停播放背景音乐。
resumeBGM	resumeBGM	恢复播放背景音乐。
setBGMVolume	setBGMVolume	设置背景音乐音量。

推流主流程接口变更

1. 创建Engine。

创建AliLiveEngine（V4.0.2）	*V4.1.0*及以上版本改为：创建AlivcLivePusher

//创建RTMP相关配置对象
AliLiveRTMPConfig rtmpConfig = new AliLiveRTMPConfig();
//初始化码率配置
rtmpConfig.videoInitBitrate = 1000;
rtmpConfig.videoTargetBitrate = 1500;
rtmpConfig.videoMinBitrate = 600;
//创建直播推流配置
AliLiveConfig mAliLiveConfig = new AliLiveConfig(rtmpConfig);
//初始化分辨率、帧率、是否开启高清预览、暂停后默认显示图片
mAliLiveConfig.videoFPS = 20;
mAliLiveConfig.videoPushProfile = AliLiveConstants.AliLiveVideoPushProfile.AliLiveVideoProfile_540P;
mAliLiveConfig.enableHighDefPreview = false;
mAliLiveConfig.pauseImage = bitmap;
mAliLiveConfig.accountId = "";
AliLiveEngine mAliLiveEngine = AliLiveEngine.create(PushActivity.this, mAliLiveConfig);
	//初始化推流配置类
	AlivcLivePushConfig mAlivcLivePushConfig = new AlivcLivePushConfig();
	//初始化分辨率，分辨率540P，最大支持720P
	mAlivcLivePushConfig.setResolution(AlivcResolutionEnum.RESOLUTION_540P);
	//初始化帧率，建议用户使用20fps
	mAlivcLivePushConfig.setFps(AlivcFpsEnum.FPS_20);
	//打开码率自适应，默认为true
	mAlivcLivePushConfig.setEnableBitrateControl(true);
	//默认为竖屏，可设置home键向左或向右横屏。
	mAlivcLivePushConfig.setPreviewOrientation(AlivcPreviewOrientationEnum.ORIENTATION_PORTRAIT);
	//设置音频编码模式
	mAlivcLivePushConfig.setAudioProfile(AlivcAudioAACProfileEnum.AlivcAudioAACProfileEnum.AAC_LC);
	AlivcLivePusher mAlivcLivePusher = new AlivcLivePusher();mAlivcLivePusher.init(mContext, mAlivcLivePushConfig);

2. 创建预览。

V4.0.2	*V4.1.0*及以上版本

//创建预览显示窗口
AliLiveRenderView mAliLiveRenderView = mAliLiveEngine.createRenderView(false);
//添加预览显示窗口到布局中
addSubView(mAliLiveRenderView);
//设置预览显示模式
mAliLiveEngine.setPreviewMode(AliLiveRenderModeAuto, AliLiveRenderMirrorModeOnlyFront);
//开始预览
mAliLiveEngine.startPreview(mAliLiveRenderView);
	livePusher对象初始化完成之后，可以进行开始预览操作。预览时需要传入摄像头预览的显示SurfaceView，示例代码如下：
	//开始预览，也可根据需求调用异步接口startPreviewAysnc来实现
	mAlivcLivePusher.startPreview(mSurfaceView)

3. 开始推流。

V4.0.2	*V4.1.0*及以上版本
mAliLiveEngine.startPush(mPushUrl);	mAlivcLivePusher.startPush(mPushUrl);

4. 停止推流。

V4.0.2	*V4.1.0*及以上版本

//停止预览
mAliLiveEngine.stopPreview();
//停止推流
mAliLiveEngine.stopPush();
//销毁liveEngine
mAliLiveEngine.destroy();
mAliLiveEngine = null;
	//停止预览
	mAliLivePusher.stopPreview();
	//停止推流
	mAliLivePusher.stopPush();
	//销毁AliLivePushe
	mAliLivePusher.destroy();
	mAliLivePusher = null;

4.2. Demo编译

本文介绍Android端Demo的编译环境要求和编译方法，并提供了Demo目录结构。

环境要求

- 硬性要求

类别	要求
系统版本	支持Android 5.0及以上。
Android API版本	最低为API 21。

类别	要求
CPU架构	ARM64、ARMv7。
集成工具	建议使用Android Studio。下载Android Studio。

- 非硬性要求（开发此Demo的环境，仅供编译运行源码的人员参考）

类别	要求
Android Studio版本	4.1.3
JRE	1.8.0_152-release-1136-b06 amd64
JVM	OpenJDK 64-Bit
compileSdkVersion	30
buildToolsVersion	30.0.3
minSdkVersion	18
targetSdkVersion	30
gradle version	gradle-5.6.4-all
gradle plugin version	com.android.tools.build:gradle:3.6.2
NDK插件版本	20.0.5594570

运行推流Demo

1. 请在SDK下载与历史记录中，下载对应版本的Demo压缩包。
2. 打开**Android Studio**，单击**Open an Existing Project**并选择Demo目录下的*AlivcLivePusherDemo*工程，即可将AlivcLivePusherDemo工程导入到Android Studio中。
3. 编译成功后，单击**运行**，安装Demo到Android终端上。
4. 体验功能。

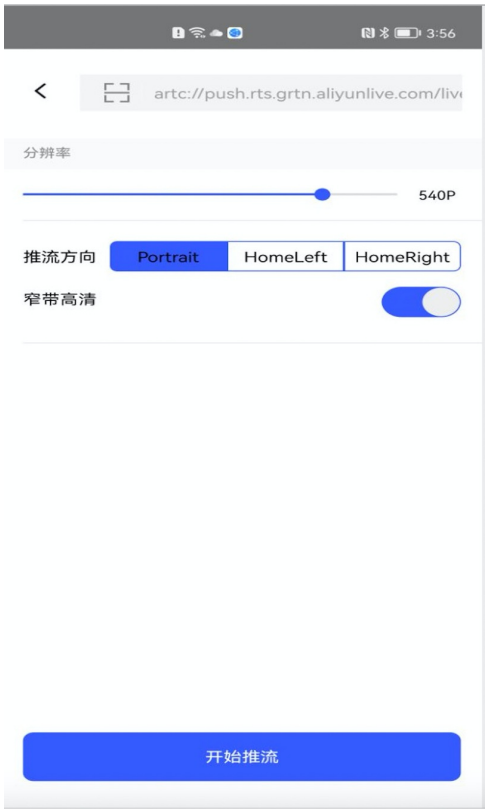
推流体验	详细内容

推流体验	详细内容
直播推流首页	
	选择**摄像头推流**即可进行如下配置： ○ 推流参数：开启**码控**、**高级设置**，同时将**显示模式**设置为**清晰度优先**。其余参数根据您的实际需要设置。 ○ 推流功能：**推流方向**设置为Portrait，**显示模式**设置为**剪裁**，其余参数保持默认状态即可。

推流体验	详细内容
直播推流配置	

推流体验	详细内容

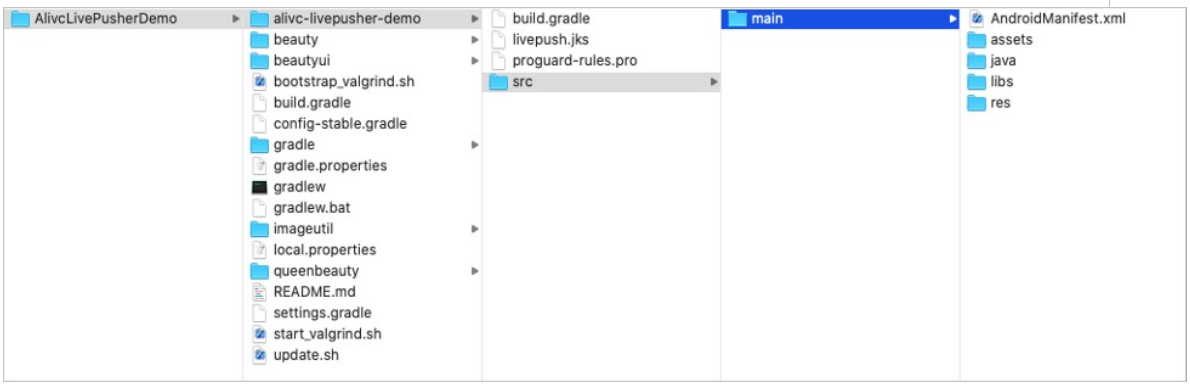

3:36
请输入推流url
推流参数
推流功能
推流方向
Portrait
HomeLeft
HomeRight
显示模式
拉伸
适合
剪裁
自动重连
重连时长
请输入/ms
重连次数
请输入/次
水印
水印位置
水印位置
镜像
推流镜像
预览镜像
进入直播

推流体验	详细内容
直播推流及美颜	输入推流URL后，即可选择**进入直播**。 **说明** 推流URL中填入有效的推流RTMP地址，推流成功后，观看的效果可以使用阿里云播放器SDK、FFplay、VLC等工具查看。
录屏推流	

推流体验	详细内容

Android Demo目录结构

- V4.1.0及以上版本Demo源码目录结构



其中main文件夹内：

- AndroidManifest.xml：Android Demo配置文件
- assets：资源文件存放位置

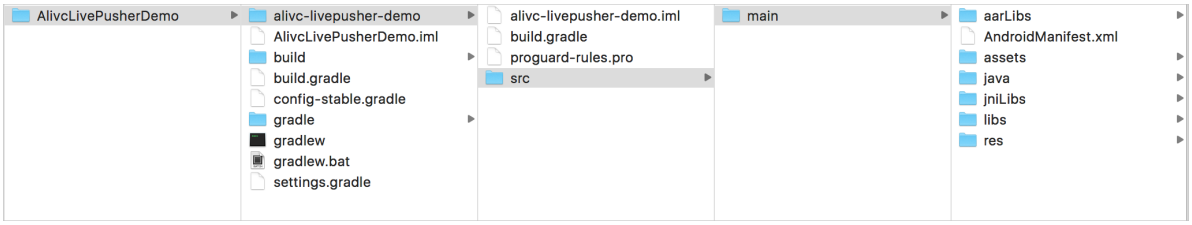
- java：Demo代码位置

  java源码目录结构如下。

- libs：依赖的JAR包位置
- res：Demo资源布局文件位置

- V3.6.1版本Demo源码目录结构

  其中main文件夹内：

  - aarLibs：依赖aar包存放路径
  - AndroidManifest.xml：Android Demo配置文件
  - assets：资源文件存放位置
  - java：Demo代码位置
  - jniLibs：依赖的so库位置
  - libs：依赖的JAR包位置
  - res：Demo资源布局文件位置

4.3. SDK集成

通过阅读本文，您可以快速了解如何集成Android端推流SDK。

集成环境

类别	要求
系统版本	支持Android 5.0及以上。
Android API版本	最低为API 21。
CPU架构	ARM64、ARMv7。
集成工具	建议使用Android Studio。下载Android Studio。

推流SDK下载

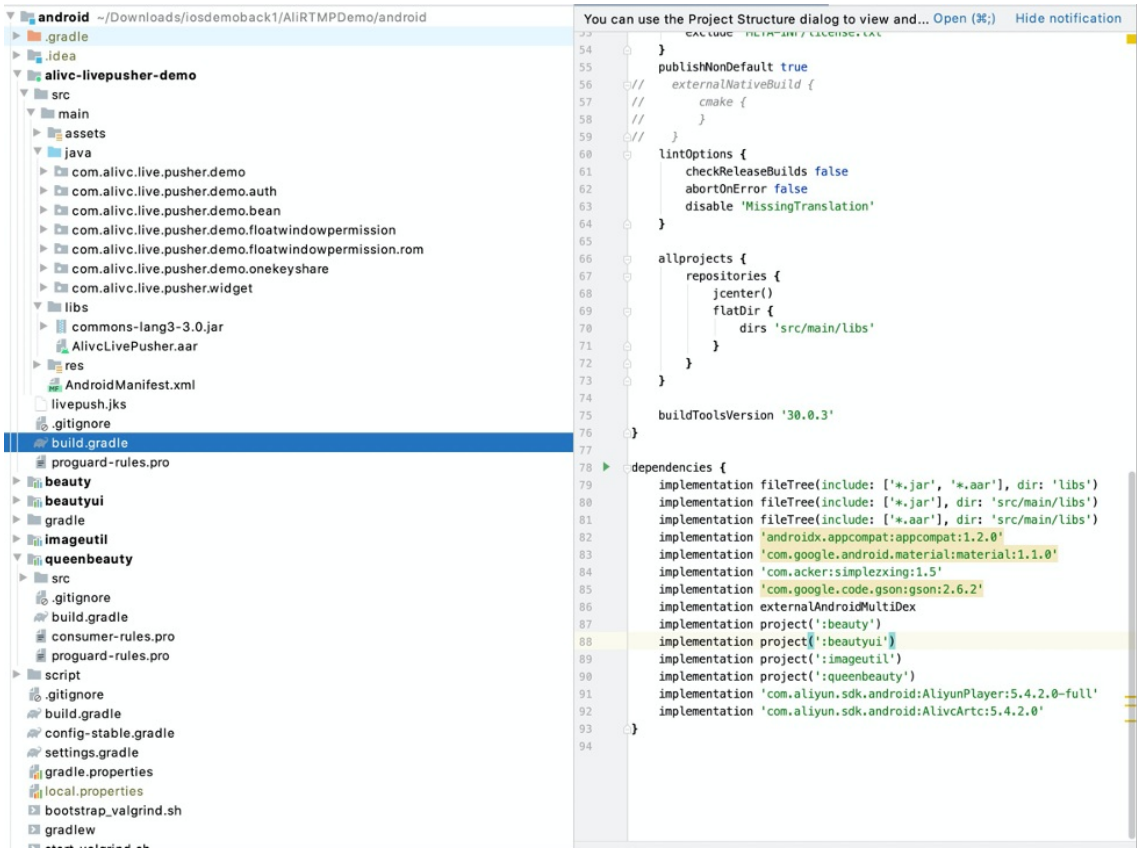
请在SDK下载与历史记录中下载最新版Android端推流SDK包。

推流SDK集成

1. 使用aar方式或maven方式集成。
   - aar方式

     将所有SDK目录下文件拷贝到自己工程对应libs目录下，并修改主模块（一般是app）的build.gradle中的dependencies，然后同步工程，代码如下：

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

     > **说明** 使用背景音乐功能时，必须集成播放器SDK（AliyunPlayer.aar）。

   - maven方式

     在工程build.gradle配置脚本中的dependencies段中添加如下代码：

     ```
     implementation 'com.alivc.pusher:AlivcLivePusher:4.3.1'
     ```

2. 添加请求权限。

   > **说明** 请务必添加录音权限和相机权限。

   在*alivc-livepusher-demo/src/main*路径下的*AndroidManifest*文件下添加如下代码：

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.REORDER_TASKS" />
<uses-permission android:name="android.permission.VIBRATE" />
//添加录音权限
<uses-permission android:name="android.permission.RECORD_AUDIO" />
//添加相机权限
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

3. 添加混淆规则。

   在alivc-livepusher-demo工程下面的*proguard-rules.pro*文件下添加如下代码：

   ```
   -keep class com.alivc.** { *;}
   ```

4. 具体使用说明。
   - API的详细注释说明，请参见V4.3.1接口说明（Android），或查看SDK包里的**android文档**文件夹。
   - 具体SDK的API使用示例代码，请参见功能使用。

4.4. 功能使用

本文详细说明Android端推流SDK主要接口、SDK的基本使用流程，以及相关功能的使用示例。

> **说明** 如果您需要使用移动端进行推流，详细操作请参见推流、拉流与播流。

Android推流SDK特性

- 支持RTMP推流协议。
- 使用视频H.264编码以及音频AAC编码。
- 支持码控、分辨率、显示模式等自定义配置。
- 支持多种摄像头相关操作。
- 支持实时美颜和自定义美颜效果调节。
- 支持增、删动态贴纸实现动态水印效果。
- 支持录屏直播。
- 支持自定义YUV、PCM等外部音视频输入。
- 支持多路混流功能。
- 支持纯音视频推流以及后台推流。

- 支持背景音乐及其相关操作。
- 支持视频截图功能。
- 支持自动重连、异常处理。
- 支持音频3A算法。

推流主要接口类列表

类	说明
AlivcLivePushConfig	推流初始配置
AlivcLivePusher	推流功能类
AlivcLivePusherErrorListener	错误回调
AlivcLivePusherNetworkListener	网络相关通知回调
AlivcLivePusherInfoListener	推流相关信息回调
AlivcLivePusherBGMListener	背景音乐回调
AlivcLivePushCustomFilter	自定义滤镜回调
AlivcLivePushCustomDetect	自定义人脸检测回调
AlivcSnapshotListener	截图回调

功能限制

使用Android推流SDK需注意以下限制：

- 您只能在推流之前设置横竖屏模式，不支持在直播的过程中实时切换。
- 在推流设定为横屏模式时，需设定界面为不允许自动旋转。
- 在硬编模式下，考虑编码器兼容问题分辨率会使用16的倍数，如设定为540P，则输出的分辨率为544*960，在设置播放器视图大小时需按输出分辨率等比缩放，避免黑边等问题。

推流SDK使用流程

SDK的基本使用流程如下：

步骤	描述	操作指引及代码示例
一、配置推流参数	完成推流基本配置、码率控制配置、分辨率自适应配置、美颜功能配置等。	配置推流参数

步骤	描述	操作指引及代码示例
二、使用推流SDK推流	初始化SDK、注册推流回调、创建预览视图后可以开始推流。用户可以根据业务需求添加推流控制、设置背景音乐、摄像头、外部音频、动态贴纸等。 注意 开始推流前必须初始化SDK及创建预览视图。	使用推流SDK推流
三、设置录屏推流（按需）	如需使用录屏推流，可以实现录屏推流相关的配置。	设置录屏推流

配置推流参数

您可以使用AlivcLivePushConfig配置推流参数，每个参数有一个对应的默认值。您可以根据需求修改对应的属性值。关于默认值和参数范围，请参见V4.3.0接口说明（Android）或注释。

> **说明** 如需在推流过程中实时修改参数，请参见AlivcLivePusher提供的属性和方法。

1. 基本推流配置。

   基本推流配置对应参数都有默认值，建议采用默认值，即您可以进行简单初始化，不做配置。 示例代码如下：

   ```
   AlivcLivePushConfig mAlivcLivePushConfig = new
   AlivcLivePushConfig();//初始化推流配置类
   mAlivcLivePushConfig.setResolution(AlivcResolutionEnum.RESOLUTION_540P);//分辨率540P，最大支持
   720P
   mAlivcLivePushConfig.setFps(AlivcFpsEnum.FPS_20); //建议用户使用20fps
   mAlivcLivePushConfig.setEnableBitrateControl(true); // 打开码率自适应，默认为true
   mAlivcLivePushConfig.setPreviewOrientation(AlivcPreviewOrientationEnum.ORIENTATION_PORTRAIT
   );
   // 默认为竖屏，可设置home键向左或向右横屏
   mAlivcLivePushConfig.setAudioProfile(AlivcAudioAACProfileEnum.AlivcAudioAACProfileEnum.AAC_LC);
   //设置音频编码模式
   ```

   注意

   - 综合手机性能和网络带宽要求，建议您将分辨率设置为540P（主流移动直播App基本都采用540P）。
   - 关闭自适应码率后，码率将固定在初始码率，不会在设定的目标码率和最小码率之间自适应调整。如果网络情况不稳定，可能造成播放卡顿，请慎用。

2. 配置码率控制。

   码率控制通过AlivcQualityModeEnum参数配置。推流SDK提供以下码率控制模式，请根据实际需求修改参数值。

码率控制模式	描述	示例代码
QM_RESOLUTION_FIRST	清晰度优先模式。SDK内部会对码率参数进行配置，优先保障推流视频的清晰度。	mAlivcLivePug.setQualityMode(AshConfilivcQualityModeEnum.QM_RESOLUTION_FIRST);//清晰度优先
QM_FLUENCY_FIRST	流畅度优先模式。SDK内部会对码率参数进行配置，优先保障推流视频的流畅度。	mAlivcLivePushConfig.setQualityMode(AlivcQualityModeEnum.QM_FLUENCY_FIRS);//流畅度优先

QM_CUSTOM	自定义模式。SDK会根据开发者设置的码率进行配置。设置为自定义模式时，您可以选择配置画质优先或者流畅度优先，并自行设定初始码率、最小码率和目标码率。
	○ 初始码率：开始直播时的码率。
	○ 最小码率：当网络较差时，码率会逐步减低到最小码率，以减少视频的卡顿。
	○ 目标码率：当网络较好时，码率会逐步提高到目标码率，以提高视频清晰度。
		mAlivcLivePushConfig.setQualityMode(AlivcQualityModeEnum.QM_CUSTOM);
		mAlivcLivePushConfig.setTargetVideoBitrate(1000); //目标码率1000Kbps
		mAlivcLivePushConfig.setMinVideoBitrate(300); //最小码率300Kbps
		mAlivcLivePushConfig.setInitialVideoBitrate(800); //初始码率800Kbps

说明

- 选择清晰度优先或流畅度优先模式时，不需设置初始码率、最小码率和目标码率（initialVideoBitrate、minVideoBitrate、targetVideoBitrate）。推流SDK内部策略会自动保障在网络抖动情况下优先考虑视频清晰度或流畅度。
- 选择自定义码率时，请参考阿里云推荐设置配置对应码率。推荐设置请参考下表内容。

自定义码率控制推荐设置（画质优先）

分辨率	初始码率 initialVideoBitrate	最小码率 minVideoBitrate	目标码率 targetVideoBitrate
360P	600	300	1000
480P	800	300	1200
540P	1000	600	1400
720P	1500	600	2000
1080P	1800	1200	2500

自定义码率控制推荐设置（流畅度优先）

分辨率	初始码率 initialVideoBitrate	最小码率 minVideoBitrate	目标码率 targetVideoBitrate
360P	400	200	600
480P	600	300	800
540P	800	300	1000
720P	1000	300	1200
1080P	1500	1200	2200

3. 配置分辨率自适应。

   分辨率自适应即动态调整推流分辨率。开启功能后，当网络较差时会自动降低分辨率以提高视频的流畅度和清晰度。示例代码如下：

   ```
   mAlivcLivePushConfig.setEnableAutoResolution(true); // 打开分辨率自适应，默认为false
   ```

   > 注意
   > - 某些播放器可能不支持动态分辨率，如果您需要使用分辨率自适应功能，建议使用阿里云播放器。
   > - 分辨率自适应只有在清晰度优先或流畅度优先时才会生效（AlivcQualityModeEnum参数配置），自定义模式时无效。

4. 配置美颜功能。

   如需在推流SDK中使用美颜功能，需要引入美颜库并配置对应回调。

   i. 通过aar方式引入美颜库。在工程的build.gradle文件中添加如下代码：

   ```
   com.aliyun.maliang.android:queen:1.3.1-official-full
   ```

   还可以集成Demo中的如下模块：

文件或文件夹	功能描述
beauty	美颜基础抽象类
beautyui	美颜基础UI控件
queenbeauty	Queen美颜组件
Imageutil	美颜Imageutil

   ii. 设置人脸识别及美颜回调。

   如果您有接入第三方美颜库的需求，可设置setCustomDetect和setCustomFilter回调。

- AlivcLivePushCustomDetect回调函数customDetectProcess（long data、int width、int height、int rotation、int format、long extra参数）中返回的参数data是采集数据的指针，第三方美颜库可对数据指针中的数据进行识别或者处理。
- AlivcLivePushCustomFilter回调函数customFilterProcess（int inputTexture、inttextureWidth、int textureHeight、long extra参数）中返回的参数inputTexture是图像的纹理texture，第三方美颜库可对纹理进行处理。如果需要返回一个处理过的纹理texture，则返回texture id。否则，返回原来的inputTexture即可。

示例代码如下：

```
/**
* 人脸识别回调
**/
mAlivcLivePusher.setCustomDetect(new AlivcLivePushCustomDetect() {
  @Override
  public void customDetectCreate() {
    Log.d(TAG, "customDetectCreate start");
    initBeautyManager();
    Log.d(TAG, "customDetectCreate end");
  }
  @Override
  public long customDetectProcess(long data, int width, int height, int rotation, int format, long ex
tra) {
    Log.d(TAG, "customDetectProcess start: data ptr:" + data + ",width:" + width + ",height:" + heig
ht + "," + format + "," + rotation);
    if (mBeautyManager != null) {
      mBeautyManager.onDrawFrame(data, BeautyImageFormat.kNV21, width, height, 0, mCamer
aId);
      Log.d(TAG, "keria: " + mCameraId);
    }
    Log.d(TAG, "customDetectProcess end");
    return 0;
  }
  @Override
  public void customDetectDestroy() {
    Log.d(TAG, "customDetectDestroy start");
    destroyBeautyManager();
    Log.d(TAG, "customDetectDestroy end");
  }
});
/**
* 美颜回调
**/
mAlivcLivePusher.setCustomFilter(new AlivcLivePushCustomFilter() {
  @Override
  public void customFilterCreate() {
    Log.d(TAG, "customFilterCreate start");
    initBeautyManager();
    Log.d(TAG, "customFilterCreate end");
  }
  @Override
  public void customFilterUpdateParam(float fSkinSmooth, float fWhiten, float fWholeFacePink, fl
oat fThinFaceHorizontal, float fCheekPink, float fShortenFaceVertical, float fBigEye) {
  }
```

```
    @Override
    public void customFilterSwitch(boolean on) {
    }
    @Override
    public int customFilterProcess(int inputTexture, int textureWidth, int textureHeight, long extra) {
        Log.d(TAG, "customFilterProcess start: textureId" + inputTexture + ",width:" + textureWidth + "
,height:" + textureHeight);
        int ret = mBeautyManager != null ? mBeautyManager.onTextureInput(inputTexture, textureWid
th, textureHeight) : inputTexture;
        Log.d(TAG, "customFilterProcess end, textureId:" + ret);
        Log.d(TAG, "keria1: " + mCameraId);
        return ret;
    }
    @Override
    public void customFilterDestroy() {
        destroyBeautyManager();
    }
});
```

5. 配置图片推流。

   为了更好的用户体验，推流SDK提供了后台图片推流和码率过低时进行图片推流的设置。当SDK退至后台时默认暂停推流视频，只推流音频，此时可以设置图片来进行图片推流和音频推流。例如，在图片上提醒用户*主播离开片刻，稍后回来*。示例代码如下：

   ```
   mAlivcLivePushConfig.setPausePushImage("退后台png图片路径");//设置用户后台推流的图片
   ```

   另外，当网络较差时您可以根据自己的需求设置推流一张静态图片。设置图片后，SDK检测到当前码率较低时，会推流此图片，避免视频流卡顿。示例代码如下所示：

   ```
   mAlivcLivePushConfig.setNetworkPoorPushImage("网络差png图片路径");//设置网络较差时推流的图片
   ```

6. 配置水印。

   推流SDK提供了添加水印功能，并且支持添加多个水印。水印图片必须为PNG格式图片。示例代码如下：

   ```
   mAlivcLivePushConfig.addWaterMark(waterPath,0.1,0.2,0.3);//添加水印
   ```

   > **说明**
   > - x、y、width为相对值，例如x为0.1表示水印的x值为推流画面x轴的10%位置，如果推流分辨率为540*960，则水印x值为54。
   > - 水印图片的高度，按照水印图片的真实宽高与输入的width值等比缩放。
   > - 要实现文字水印，可以先将文字转换为图片，再使用此接口添加水印。

7. 配置预览显示模式。

   推流SDK支持三种预览模式，预览显示模式不影响推流。

   - AlivcPreviewDisplayMode.ALIVC_LIVE_PUSHER_PREVIEW_SCALE_FILL：预览显示时，铺满窗口。当视频比例和窗口比例不一致时，预览会有变形。
   - AlivcPreviewDisplayMode.ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FIT：预览显示时，保持视频比例。当视频比例与窗口比例不一致时，预览会有黑边。

- AlivcPreviewDisplayMode.ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FILL：预览显示时，剪切视频以适配窗口比例。当视频比例和窗口比例不一致时，预览会裁剪视频。

示例代码如下：

```
mAlivcLivePushConfig.setPreviewDisplayMode(AlivcPreviewDisplayMode.ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FIT);
```

使用推流SDK推流

AlivcLivePusher为使用推流SDK推流的核心类，提供初始化操作、推流回调、摄像头预览、推流控制、推流过程中的参数调节等功能。

> **说明**
> - 接口的调用需要对接口抛出的异常进行处理，添加try catch处理操作。
> - 接口调用的顺序必须按照说明的顺序调用，否则会因调用顺序不正确而出现异常。

1. 初始化。

   在配置好推流参数后，可以使用推流SDK的init方法进行初始化。示例代码如下：

   ```
   AlivcLivePusher mAlivcLivePusher = new AlivcLivePusher();
   mAlivcLivePusher.init(mContext, mAlivcLivePushConfig);
   ```

   > **说明** AlivcLivePusher目前不支持多实例，所以一个init必须对应有一个destory

2. 注册推流回调。

   推流回调分为三种：

   - Info：主要做提示和状态检测使用。
   - Error：错误回调。
   - Network：主要为网络相关。

   用户通过对应的回调通知，当发生对应的类型的事件时，相应的回调函数被触发运行。示例代码如下：

   ```
   /**
    * 设置推流错误事件
    *
    * @param errorListener 错误监听器
    */
   mAlivcLivePusher.setLivePushErrorListener(new AlivcLivePushErrorListener() {
    @Override
    public void
   onSystemError(AlivcLivePusher livePusher, AlivcLivePushError error) {
       if(error != null) {
           //添加UI提示或者用户自定义的错误处理
       }
    }
    @Override
    public void onSDKError(AlivcLivePusher
   livePusher, AlivcLivePushError error) {
       if(error != null) {
           //添加UI提示或者用户自定义的错误处理
   ```

```
        //添加UI提示或者用户自定义的错误处理
    }
  }
});
/**
* 设置推流通知事件
*
* @param infoListener 通知监听器
*/
mAlivcLivePusher.setLivePushInfoListener(new AlivcLivePushInfoListener() {
 @Override
 public void
onPreviewStarted(AlivcLivePusher pusher) {
    //预览开始通知
 }
 @Override
 public void
onPreviewStoped(AlivcLivePusher pusher) {
    //预览结束通知
 }
 @Override
 public void
onPushStarted(AlivcLivePusher pusher) {
    //推流开始通知
 }
 @Override
 public void onPushPauesed(AlivcLivePusher
pusher) {
    //推流暂停通知
 }
 @Override
 public void
onPushResumed(AlivcLivePusher pusher) {
    //推流恢复通知
 }
 @Override
 public void
onPushStoped(AlivcLivePusher pusher) {
    //推流停止通知
 }
 @Override
 public void onPushRestarted(AlivcLivePusher
pusher) {
    //推流重启通知
 }
 @Override
 public void
onFirstFramePreviewed(AlivcLivePusher pusher) {
    //首帧渲染通知
 }
 @Override
 public void onDropFrame(AlivcLivePusher
pusher, int countBef, int countAft) {
    //丢帧通知
 }
```

```
 @Override
 public void
onAdjustBitRate(AlivcLivePusher pusher, int curBr, int targetBr) {
    //调整码率通知
 }
 @Override
 public void onAdjustFps(AlivcLivePusher
pusher, int curFps, int targetFps) {
    //调整帧率通知
 }
});
/**
* 设置网络通知事件
*
* @param infoListener 通知监听器
*/
mAlivcLivePusher.setLivePushNetworkListener(new AlivcLivePushNetworkListener()
{
 @Override
 public void
onNetworkPoor(AlivcLivePusher pusher) {
    //网络差通知
 }
 @Override
 public void
onNetworkRecovery(AlivcLivePusher pusher) {
    //网络恢复通知
 }
 @Override
 public void
onReconnectStart(AlivcLivePusher pusher) {
    //重连开始通知
 }
 @Override
 public void
onReconnectFail(AlivcLivePusher pusher) {
    //重连失败通知
 }
 @Override
 public void
onReconnectSucceed(AlivcLivePusher pusher) {
    //重连成功通知
 }
 @Override
 public void
onSendDataTimeout(AlivcLivePusher pusher) {
    //发送数据超时通知
 }
 @Override
 public void
onConnectFail(AlivcLivePusher pusher) {
    //连接失败通知
 }
});
```

```
/**
* 设置背景音乐播放通知事件
*
* @param pushBGMListener 通知监听器
*/
mAlivcLivePusher.setLivePushBGMListener(new AlivcLivePushBGMListener() {
 @Override
 public void onStarted() {
    //播放开始通知
 }
 @Override
 public void onStoped() {
    //播放停止通知
 }
 @Override
 public void onPaused() {
    //播放暂停通知
 }
 @Override
 public void onResumed() {
    //播放恢复通知
 }
 /**
  * 播放进度事件
  *
  * @param progress 播放进度
  */
 @Override
 public void onProgress(final long
progress, final long duration) {
 @Override
 public void onCompleted() {
    //播放结束通知
 }
 @Override
 public void onDownloadTimeout() {
    //播放器超时事件，在这里处理播放器重连并且seek到播放位置
 }
 @Override
 public void onOpenFailed() {
    //流无效通知，在这里提示流不可访问
 }
});
```

3. 开始预览。

   livePusher对象初始化及回调配置完成之后，可以进行开始预览操作。预览时需要传入摄像头预览的显示SurfaceView。示例代码如下：

```
mAlivcLivePusher.startPreview(mSurfaceView)//开始预览，也可根据需求调用异步接口startPreviewAysnc来实现
```

> **注意** 部分接口（如背景音乐、摄像头相关操作）必须在设置预览之后才能调用。因此，建议开始推流前先进行预览。

4. 开始推流。

   预览成功后才可以开始推流，因此需监听onPreviewStarted回调，在回调里面添加如下代码。

```
mAlivcLivePusher.startPush(mPushUrl);
```

> **说明**
>
> - 推流SDK同时提供了异步方法，可调用startPushAysnc来实现。
> - 推流SDK支持RTMP的推流地址，阿里云推流地址获取请参见推流地址和播放地址。
> - 使用正确的推流地址开始推流后，可用播放器（阿里云播放器、FFplay、VLC等）进行拉流测试，拉流地址获取请参见推流地址和播放地址。

5. 设置其他推流控制。

   推流控制主要包括开始推流、停止推流、停止预览、重新推流、暂停推流、恢复推流、销毁推流等操作，用户可以根据业务需求添加按钮进行操作。示例代码如下：

```
/*正在推流状态下可调用暂停推流。暂停推流后，视频预览和视频推流保留在最后一帧，音频推流继续。*/
mAlivcLivePusher.pause();
/*暂停状态下可调用恢复推流。恢复推流后，音视频预览与推流恢复正常。*/
mAlivcLivePusher.resume();
/*推流状态下可调用停止推流，完成后推流停止。*/
mAlivcLivePusher.stopPush();
/*在预览状态下才可以调用停止预览，正在推流状态下，调用停止预览无效。预览停止后，预览画面定格在最后一帧。*/
mAlivcLivePusher.stopPreview();
/*推流状态下或者接收到所有Error相关回调状态下可调用重新推流,且Error状态下只可以调用此接口(或者reconnectPushAsync重连)或者调用destory销毁推流。完成后重新开始推流，重启ALivcLivePusher内部的一切资源，包括预览、推流等等restart。*/
mAlivcLivePusher.restartPush();
/*推流状态下或者接收到AlivcLivePusherNetworkDelegate相关的Error回调状态下可调用此接口,且Error状态下只可以调用此接口(或者restartPush重新推流)或者调用destory销毁推流。完成后推流重连，重新链接推流RTMP。*/
mAlivcLivePusher.reconnectPushAsync();
/*销毁推流后，推流停止，预览停止，预览画面移除。AlivcLivePusher相关的一切资源销毁。*/
mAlivcLivePusher.destroy();
```

6. 设置背景音乐。

   推流SDK提供了背景音乐播放、混音、降噪、耳返、静音等功能。示例代码如下：

```
/*开始播放背景音乐。*/
mAlivcLivePusher.startBGMAsync(mPath);
/*停止播放背景音乐。若当前正在播放BGM，并且需要切换歌曲，只需要调用开始播放背景音乐接口即可，无需停止当前正在播放的背景音乐。*/
mAlivcLivePusher.stopBGMAsync();
/*暂停播放背景音乐，背景音乐开始播放后才可调用此接口。*/
mAlivcLivePusher.pauseBGM();
/*恢复播放背景音乐，背景音乐暂停状态下才可调用此接口。*/
mAlivcLivePusher.resumeBGM();
/*开启循环播放音乐*/
mAlivcLivePusher.setBGMLoop(true);
/*设置降噪开关。打开降噪后，将对采集到的声音中非人声的部分进行过滤处理。可能存在对人声稍微抑制作用，建议让用户自由选择是否开启降噪功能，默认不使用*/
mAlivcLivePusher.setAudioDenoise(true);
/*设置耳返开关。耳返功能主要应用于KTV场景。打开耳返后，插入耳机将在耳机中听到主播说话声音。关闭后，插入耳机无法听到人声。未插入耳机的情况下，耳返不起作用。*/
mAlivcLivePusher.setBGMEarsBack(true);
/*混音设置，提供背景音乐和人声采集音量调整。*/
mAlivcLivePusher.setBGMVolume(50);//设置背景音乐音量
mAlivcLivePusher.setCaptureVolume(50);//设置人声采集音量
/*设置静音。静音后音乐声音和人声输入都会静音。要单独设置音乐或人声静音可以通过混音音 量设置接口来调整。*/
mAlivcLivePusher.setMute(true);
```

> 注意 背景音乐相关接口在开始预览之后才可调用。

7. 摄像头相关操作。

   摄像头相关操作包括推流状态、暂停状态、重连状态等，可操作摄像头切换、闪关灯、焦距、变焦和镜像设置等。示例代码如下：

```
/*切换前后摄像头*/
mAlivcLivePusher.switchCamera();
/*开启/关闭闪光灯，在前置摄像头时开启闪关灯无效*/
mAlivcLivePusher.setFlash(true);
/*焦距调整，即可实现采集画面的缩放功能。缩放范围为[0,getMaxZoom()]。*/
mAlivcLivePusher.setZoom(5);
/*手动对焦。手动聚焦需要传入两个参数:1.point 对焦的点（需要对焦的点的坐标);2.autoFocus 是否需要自动对焦，只有本次对焦操作调用该接口时，该参数才生效。后续是否自动对焦沿用上述自动聚焦接口设置值。*/
mAlivcLivePusher.focusCameraAtAdjustedPoint(x, y, true);
/*设置是否自动对焦*/
mAlivcLivePusher.setAutoFocus(true);
/*镜像设置。镜像相关接口有两个，PushMirror推流镜像和PreviewMirror预览镜像。PushMirror设置仅对播放画面生效，PreviewMirror仅对预览画面生效，两者互不影响。*/
mAlivcLivePusher.setPreviewMirror(false);
mAlivcLivePusher.setPushMirror(false);
```

> 注意 您摄像头相关接口只能在开始预览之后调用。

8. 配置外部音视频输入。

   推流SDK支持将外部的音视频源输入进行推流，比如推送一个音视频文件。

i. 配置外部音视频输入。

示例代码如下：

```
/**
* 输入自定义音频数据
* @param data 音频数据byte array
* @param size
* @param sampleRate
* @param channels
* @param pts 音频数据pts(μs)
* 此接口不控制时序，需要调用方控制输入音频帧的时序
*/
mAlivcLivePusher. inputStreamAudioData(byte[] data, int size, int sampleRate, int channels, long pts);
/**
* 输入自定义音频数据
* @param dataPtr 音频数据native内存指针
* @param size
* @param sampleRate
* @param channels
* @param pts 音频数据pts(μs)
* 此接口不控制时序，需要调用方控制输入音频帧的时序
*/
mAlivcLivePusher. inputStreamAudioPtr(long dataPtr, int size, int sampleRate, int channels, long pts);
```

ii. 插入外部视频数据。

示例代码如下：

```
/**
* 输入自定义视频流
*
* @param data 视频图像byte array
* @param width 视频图像宽度
* @param height 视频图像高度
* @param size 视频图像size
* @param size 视频图像stride
* @param pts 视频图像pts（μs）
* @param rotation 视频图像旋转角度
* 此接口不控制时序，需要调用方控制输入视频帧的时序
* 附：调用此接口时需要在config中设置setExternMainStream(true,***)
*/
mAlivcLivePusher. inputStreamVideoData(byte[] data, int width, int height, int stride, int size, long pts, int rotation);
/**
* 输入自定义视频流
*
* @param dataptr 视频图像native内存指针
* @param width 视频图像宽度
* @param height 视频图像高度
* @param size 视频图像stride
* @param size 视频图像size
* @param pts 视频图像pts（μs）
* @param rotation 视频图像旋转角度
* 此接口不控制时序，需要调用方控制输入视频帧的时序
* 附：调用此接口时需要在config中设置setExternMainStream(true,***)
*/
mAlivcLivePusher. inputStreamVideoPtr(long dataptr, int width, int height, int stride, int size, long pts, int rotation);
```

iii. 插入音频数据。

示例代码如下：

```
/**
* AlivcImageFormat输入的视频图像格式
* AlivcSoundFormat输入的音频帧格式
* 其他参数：如输出分辨率，音频采样率，通道数等在config里
setResolution，setAudioSamepleRate，setAudioChannels里设置
* 附：输入自定义视频和音频流时，调用inputStreamVideoData，inputStreamAudioData等接口。
*/
mAlivcLivePushConfig.setExternMainStream(true,AlivcImageFormat.IMAGE_FORMAT_YUVNV12,
AlivcSoundFormat.SOUND_FORMAT_S16);
```

9. 动态贴纸。

推流SDK实现了在直播流中添加动态贴纸效果，使用此功能可实现动态水印效果。

i. 动态贴纸的制作可参考Demo提供的素材进行简单修改。自己制作动图贴纸的序列帧图片，并打开config.json文件自定义以下参数：

```
"du": 2.04,//播放一遍动画持续的时间
"n": "qizi",//动图的名称，制作动图时文件夹以动图名称命名，每一张图片以动图名称+序号命名，比如qizi0
"c": 68.0,//动画帧数，即一个完整动画的图片数量
"kerneframe": 51,//关键帧，即指定哪一张图片为关键帧，比如demo中指定第51帧为关键帧（需确保51帧是存在的）
"frameArry": [
  {"time":0,"pic":0},
  {"time":0.03,"pic":1},
  {"time":0.06,"pic":2},
  ],
//动画参数，上述参数即表示第0秒显示第一帧（qizi0），第0.03秒显示第二帧（qizi1）...以此规则填写所有帧的动画
```

> **说明** 其他字段可以直接使用demo提供的json文件中的内容，无需修改。

ii. 添加动态贴纸。

示例代码如下：

```
/**
* 添加动态贴纸
* @param path 贴纸文件路径，必须含config.json
* @param x 显示起始x位置(0~1.0f)
* @param y 显示起始y位置(0~1.0f)
* @param w 显示宽度(0~1.0f)
* @param h 显示高度(0~1.0f)
* @return id 贴纸id，删除贴纸时需设置id
*/
mAlivcLivePusher.addDynamicsAddons("贴纸路径", 0.2f, 0.2f, 0.2f, 0.2f);
```

iii. 删除动态贴纸。

示例代码如下：

```
mAlivcLivePusher.removeDynamicsAddons(int id);
```

10. 其他接口的使用。

```
/*在自定义模式下，用户可以实时调整最小码率和目标码率。*/
mAlivcLivePusher.setTargetVideoBitrate(800);
mAlivcLivePusher.setMinVideoBitrate(400);
/*是否支持自动对焦*/
mAlivcLivePusher.isCameraSupportAutoFocus();
/*是否支持闪光灯*/
mAlivcLivePusher.isCameraSupportFlash();
/*获取是否正在推流的状态*/
mAlivcLivePusher.isPushing();
/*获取推流地址*/
mAlivcLivePusher.getPushUrl();
/*获取推流性能调试信息。推流性能参数具体参数和描述参考API文档或者接口注释。*/
mAlivcLivePusher.getLivePushStatsInfo();
/*获取版本号。*/
 mAlivcLivePusher.getSDKVersion();
/*设置log级别，根据需求过滤想要的调试信息*/
mAlivcLivePusher.setLogLevel(AlivcLivePushLogLevelAll);
/*获取当前sdk状态*/
AlivcLivePushStats getCurrentStatus();
/*获取上一个错误码，如无错误返回：ALIVC_COMMON_RETURN_SUCCESS*／
AlivcLivePushError getLastError();
```

设置录屏推流

推流SDK支持录屏推流。使用录屏推流需在初始化操作、设置预览及开始推流之后进行相关配置。具体操作如下：

1. 配置录屏模式。

   Android推流SDK支持三种录屏模式，请按需设置。

录屏模式	所需配置
录屏时不开启摄像头	i. 在config中设置权限请求的返回数据即可。
录屏时开启摄像头 **说明** 主播端有摄像头预览，同样观众端也有摄像头画面（通过录屏录制进去）	i. 在config中设置权限请求的返回数据。 ii. 调用StartCamera传入surfaceView。
录屏时开启摄像头 **说明** 主播端无摄像头预览，观众端有摄像头画面叠加	i. 在config中设置权限请求的返回数据。 ii. 调用StartCamera无需传入surfaceView。 iii. 调用startCameraMix传入观众端摄像头画面显示位置。

2. 设置开启录屏权。

   录屏采用MediaProjection，需要用户请求权限，将权限请求返回的数据通过此接口设置，即开启录屏模式。录屏情况下，默认不开启摄像头。请在推流配置里进行配置，示例代码如下：

```
mAlivcLivePushConfig.setMediaProjectionPermissionResultData(resultData)
```

3. 设置摄像头预览。

   在录屏开启成功后，调用开启或关闭摄像头预览接口，示例代码如下：

```
mAlivcLivePusher.startCamera(surfaceView);//开启摄像头预览
mAlivcLivePusher.stopCamera();//关闭摄像头预览
```

> **说明**
> - 录屏模式下摄像头预览surfaceView的长宽建议设置成1：1，这样在屏幕旋转时无需调整surfaceview。
> - 若设置的长宽不为1：1，则需要在屏幕旋转时，调整surfaceView的比例后，先stopCamera再startCamera。
> - 如果主播端不需要预览，则surfaceview填为null。

4. 设置摄像头混流。

   当主播端不需要摄像头预览，观众端需要的情况可开启混流，主要应用于游戏直播，主播不想玩游戏的时候摄像头画面挡住游戏画面，示例代码如下：

```
/**
* @param x 混流显示x初始位置(0~1.0f)
* @param y 混流显示y初始位置(0~1.0f)
* @param w 混流显示宽度(0~1.0f)
* @param h 混流显示高度(0~1.0f)
* @return
*/
mAlivcLivePusher.startCameraMix(x, y, w, h);//开启摄像头混流
mAlivcLivePusher.stopCameraMix();//停止摄像头混流
```

5. 设置屏幕旋转。

   录屏模式下，可设置感应的屏幕旋转角度，支持横屏和竖屏录制，示例代码如下：

```
mAlivcLivePusher.setScreenOrientation(0);
```

> **说明** 在横竖屏切换时，需要在应用层监听OrientationEventListener事件，并将旋转角度设置到此接口。

6. 设置隐私。

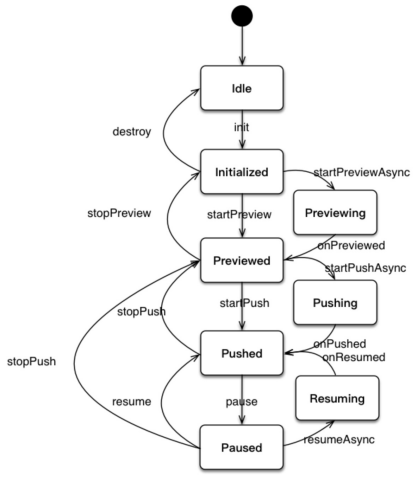
   当主播在录屏时要进行密码输入等操作时，主播可以开启隐私保护功能，结束操作后可以关闭隐私，示例代码如下：

```
mAlivcLivePusher.pauseScreenCapture();//开启隐私保护
mAlivcLivePusher.resumeScreenCapture();//关闭隐私保护
```

> **说明** 暂停录屏，如果在config中设置了setPausePushImage则观众端会在此接口后显示图片。如果没有，则观众端停留在最后一帧。

注意事项

使用Android推流SDK需要注意以下事项：

事项	说明
混淆规则	检查混淆，确认已将SDK相关包名加入至不混淆名单中。 `-keep class com.alivc.** { *;}`
接口调用	• 同步和异步接口都可以正常调用，尽量使用异步接口调用，可以避免对主线程的资源消耗。 • SDK接口会在发生错误或者调用顺序不对时抛出异常（throws），调用时注意添加try catch处理，否则会造成程序的crash。 • 接口调用顺序，如下图所示： 
历史版本升级说明	由于旧版本已暂停维护，如您使用的Android推流SDK版本低于4.1.0，请将推流SDK升级至最新版，版本升级详情请参见V4.0.2升级至V4.1.0及以上迁移说明。 **注意** 升级时需注意集成最新的播放器SDK。

4.5. 异常及特殊场景处理

本文介绍使用Android推流SDK可能出现的异常情况、特殊情况，及其处理办法。

当您收到AlivcLivePushErrorListener时

- 当出现onSystemError系统级错误时，您需要退出直播。
- 当出现onSDKError错误（SDK错误）时，有两种处理方式，选择其一即可：销毁当前直播并重新创建，或调用restartPush或restartPushAsync重启AlivcLivePusher。
- 您需要特别处理App没有麦克风权限和没有摄像头权限的回调，App没有麦克风权限错误码为

ALIVC_PUSHER_ERROR_SDK_CAPTURE_CAMERA_OPEN_FAILED，App没有摄像头权限错误码为ALIVC_PUSHER_ERROR_SDK_CAPTURE_MIC_OPEN_FAILED。

当您收到AlivcLivePushNetworkListener时

- 网速慢时，回调onNetworkPoor，当您收到此回调说明当前网络对于推流的支撑度不足，此时推流仍在继续、没有中断。网络恢复时，回调onNetworkRecovery，您可以在此处理自己的业务逻辑，比如UI提醒用户。
- 网络出现相关错误时，回调onConnectFail、onReconnectError或onSendDataTimeout。有两种处理方式，您只需选择其一：销毁当前推流并重新创建，或调用reconnectAsync进行重连，建议您重连之前先进行网络检测和推流地址检测。
- SDK内部每次自动重连或者开发者主动调用reconnectAsync重连接口的情况下，会回调onReconnectStart重连开始。每次重连都会对RTMP进行重连链接。

  RTMP链接建立成功之后会回调onReconnectSuccess，此时只是链接建立成功，并不意味着可以推流数据成功。如果链接成功之后，由于网络等原因导致推流数据发送失败，SDK会继续重连。
- 推流地址鉴权即将过期会回调onPushURLAuthenticationOverdue。如果您的推流开启了推流鉴权功能（推流URL中带有auth_key），我们会对推流URL做出校验。在推流URL过期前约1min，您会收到此回调，实现该回调后，您需要回传一个新的推流URL，以此保证不会因为推流地址过期而导致推流中断。示例代码如下：

```
String onPushURLAuthenticationOverdue(AlivcLivePusher pusher) {
  return "新的推流地址 rtmp://";
}
```

当您收到AlivcLivePusherBGMListener背景音乐错误回调时

- 背景音乐开启失败时会回调onOpenFailed，检查背景音乐开始播放接口所传入的音乐路径与该音乐文件是否正确，可调用startBGMAsync重新播放。
- 背景音乐播放超时会回调onDownloadTimeout，多出现于播放网络URL的背景音乐，提示主播检查当前网络状态，可调用startBGMAsync重新播放。

当网络中断时

- 短时间断网和网络切换：即短时间的网络波动或者网络切换。一般情况下，中途断网时长在AlivcLivePushConfig设置的重连超时时长和次数范围之内，SDK会进行自动重连，重连成功之后将继续推流。若您使用阿里云播放器，建议播放器收到超时通知之后短暂延时5s后再做重连操作。
- 长时间断网：断网时长超出AlivcLivePushConfig设置的重连超时时长和次数范围时，SDK自动重连失败，此时会回调onReconnectError，在等到网络恢复之后调用reconnectAsync接口进行重连。同时播放器也要配合做重连操作。
  - 建议您在SDK外部做网络监测。
  - 主播端和播放端在客户端无法进行直接通信，需要配合服务端使用。比如主播端断网，服务端会收到CDN的推流中断回调，此时可以推送给播放端，主播推流中断，播放端再做出相应处理。恢复推流同理。
  - 阿里云播放器重连需要先停止播放再开始播放。调用接口顺序stop>prepareAndPlay。

```
mPlayer.stop();
mPlayer.prepareAndPlay(mUrl);
```

> **说明** 关于播放器请参见阿里云播放器SDK使用说明。

后台运行和锁屏

- 当App退至后台或锁屏时，您可调用AlivcLivePusher的pause()或resume()接口，暂停或恢复推流。
- 对于非系统的音视频通话，SDK会采集声音并推送出去，您可以根据业务需求在退后台或锁屏时调用静音接口mAlivcLivePusher.setMute（true或false）来决定后台时是否采集音频。

码率设置

SDK内部有动态变化码率策略，您可以在AlivcLivePushConfig中修改码率预设值。根据产品需求对于视频分辨率、视频流畅度、视频清晰度的要求不同，对应设置的码率范围也不同，具体参考如下：

- 视频清晰度：推流码率越高，则视频清晰度越高。所以推流分辨率越大，所需要设置的码率也就越大，以保证视频清晰度。
- 视频流畅度：推流码率越高，所需要的网络带宽越大。所以在较差的网络环境下，设置较高的码率有可能影响视频流畅度。

5.iOS推流SDK

5.1. V4.0.2升级至V4.1.0及以上迁移说明

如果您现在使用的是V4.0.2版本的iOS推流SDK，想要升级到*V4.1.0*及以上版本时，可参考本文步骤进行升级。

前提条件

请下载最新版本的推流SDK。

新版推流SDK，请参见SDK下载与历史记录。

升级步骤

从项目中移除SDK V4.0.2相关类库和资源文件，添加*V4.1.0*及以上版本相关类库和资源文件，排查并更新相关API和推流主流程接口。

1. 在工程libs目录下：用*V4.1.0*及以上版本中需要加入*AlivcLivePusher.framework*和*AlivcLibRtmp.framework*替换V4.0.2项目中的*AliLivesdk.framework*。
2. 播放器相关SDK需要升级到AliyunPlayer，请参考SDK及Demo中使用的SDK。
3. 如果接入Queen智能美化特效，请在Demo中查看使用方式，方便美颜SDK及UI的接入。
4. 相关API会有部分调整，请根据核心接口对比排查并更新当前代码中的API。
5. 推流主流程接口有部分变更，请根据推流主流程接口变更修改当前代码。

核心接口对比

- 基础接口

V4.0.2	*V4.1.0*及以上版本	描述
getSdkVersion	getSdkVersion	获取版本号。
initWithConfig	initWithConfig	创建推流实例。
destorySdk	destroy	销毁推流。
◦ setStatusDelegate ◦ setRtsDelegate ◦ setVidePreProcessDelegate ◦ setDataStatsDelegate	◦ AlivcPublisherViewDelegate ◦ AlivcLivePusherInfoDelegate ◦ AlivcLivePusherErrorDelegate	RTC推流模式下订阅某个粉丝媒体相关回调，详情请参见AliLiveRtsDelegate设置视频前处理回调。设置直播媒体参数回调。
setNetworkDelegate	AlivcLivePusherNetworkDelegate	设置推流网络状态相关回调。
setLogDirPath	无：自定义Log写入	设置SDK日志文件保存路径。如需调用，请在调用所有API之前先调用此接口，避免日志出现丢失，同时保证指定的目录已存在且可写入。

V4.0.2	*V4.1.0*及以上版本	描述
setLogLevel	无	设置日志输出级别。

- 推流基础接口

V4.0.2	*V4.1.0*及以上版本	描述
startPreview	startPreview	开始预览（主播端接口）。
stopPreview	stopPreview	停止预览（主播端接口）。
pausePush	pause	暂停摄像头采集并进入垫片推流状态（仅支持RTMP模式推流）。需要先调用startPush后才可以调用pausePush，否则调用顺序会出错。
resumePush	resume()	恢复摄像头采集并结束垫片推流状态（仅支持RTMP模式推流）。需要先调用pausePush后才可以调用resumePush，否则调用顺序会出错。
startPush	startPushWithURL	开始推流。
stopPush	stopPush	停止推流。
isPublishing	isPushing	查询是否正在推流。
getPublishUrl	getPushURL	获取当前推流的地址。

- 视频相关接口

V4.0.2	*V4.1.0*及以上版本	描述
setPreviewMode	setpreviewDisplayMode	设置预览模式。
switchCamera	switchCamera	切换前后摄像头。
setCameraZoom	setZoom	设置摄像头缩放及是否允许闪光灯。
isCameraExposurePointSupported	setExposure	摄像头是否支持设置曝光区域。
setCameraFocusPoint	setAutoFocus	设置摄像头聚焦。

- 音频相关接口

V4.0.2	*V4.1.0*及以上版本	描述
setMute	setMute	设置本地音频采集是否为静音帧。
isAudioOnly	isAudioOnly	查询是否纯音频推流。

V4.0.2	*V4.1.0*及以上版本	描述
enableEarBack	setBGMEarsBack	启用耳返。建议在插入耳机后开启耳返，否则可能会引入回声。
playBGM	startBGMAsync	播放背景音乐。
stopBGM	stopBGM	停止播放背景音乐。
pauseBGM	pauseBGM	暂停播放背景音乐。
resumeBGM	resumeBGM	恢复播放背景音乐。
setBGMVolume	setBGMVolume	设置背景音乐音量。

推流主流程接口变更

1. 创建Engine。

创建AliLiveEngine（V4.0.2）	*V4.1.0*及以上版本改为：创建AlivcLivePusher

//创建RTMP相关配置对象
头文件引用。#import <AliLiveSdk/AliLiveSdk.h>
创建AliLiveEngine。AliLiveConfig *config = [[AliLiveConfig alloc] init];
config.videoProfile = AliLiveVideoProfile_540P;
config.videoFPS = 20;
myConfig.pauseImage = [UIImage imageNamed:@"background_img.png"];
myConfig.accountID = @"";
AliLiveEngine *engine = [[AliLiveEngine alloc] initWithConfig:myConfig];
[engine setAudioSessionOperationRestriction:AliLiveAudioSessionOperationRestrictionDeactivateSession];
[engine setRtsDelegate:self];
[engine setStatusDelegate:self];
	在需要使用推流器的ViewController中引用头文件 #import <AlivcLivePusher/AlivcLivePusherHeader.h> ，示例代码如下：
	//初始化推流配置类，也可使用initWithResolution来初始化
	AlivcLivePushConfig *config = [[AlivcLivePushConfig alloc] init];
	//默认为540P，最大支持720P
	config.resolution = AlivcLivePushResolution540P;
	//建议用户使用20fps
	config.fps = AlivcLivePushFPS20;
	// 打开码率自适应，默认为true
	config.enableAutoBitrate = true;
	//默认值为2，关键帧间隔越大，延时越高。建议设置为1-2
	config.videoEncodeGop = AlivcLivePushVideoEncodeGOP_2;
	// 单位为毫秒，重连时长2s，重连间隔设置不小于1秒，建议使用默认值即可
	config.connectRetryInterval = 2000;
	// 默认为false，正常情况下都选择false即可
	config.previewMirror = false;
	// 默认为竖屏，可设置home键向左或向右横屏
	config.orientation = AlivcLivePushOrientationPortrait;

2. 创建预览。

V4.0.2	*V4.1.0*及以上版本
开始预览。[self.engine startPreview:self.renderView];	推流预览显示支持以下三种模式： ○ ALIVC_LIVE_PUSHER_PREVIEW_SCALE_FILL：铺满窗口，视频比例和窗口比例不一致时预览会有变形。 ○ ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FIT：保持视频比例，视频比例和窗口比例不一致时有黑边（默认）。 ○ ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FILL：剪切视频以适配窗口比例，视频比例和窗口比例不一致时会裁剪视频。 这三种模式可以在AlivcLivePushConfig中设置，也可以在预览中和推流中通过API setpreviewDisplayMode进行动态设置。 [self.livePusher startPreview:self.view];

3. 开始推流。

V4.0.2	*V4.1.0*及以上版本
[self.engine startPushWithURL:self.pushUrl];	[self.livePusher startPushWithURL:@"推流测试地址(rtmp://.....)"];

4. 停止推流。

V4.0.2	*V4.1.0*及以上版本
[self.engine stopPush]; [self.engine stopPreview]; [self.engine destorySdk]; self.engine = nil;	[self.livePusher destory]; self.livePusher = nil; /*获取推流状态。*/ AlivcLivePushStatus status = [self.livePusher getLiveStatus];

5.2. Demo编译

本文介绍iOS端Demo的编译环境要求和编译方法。

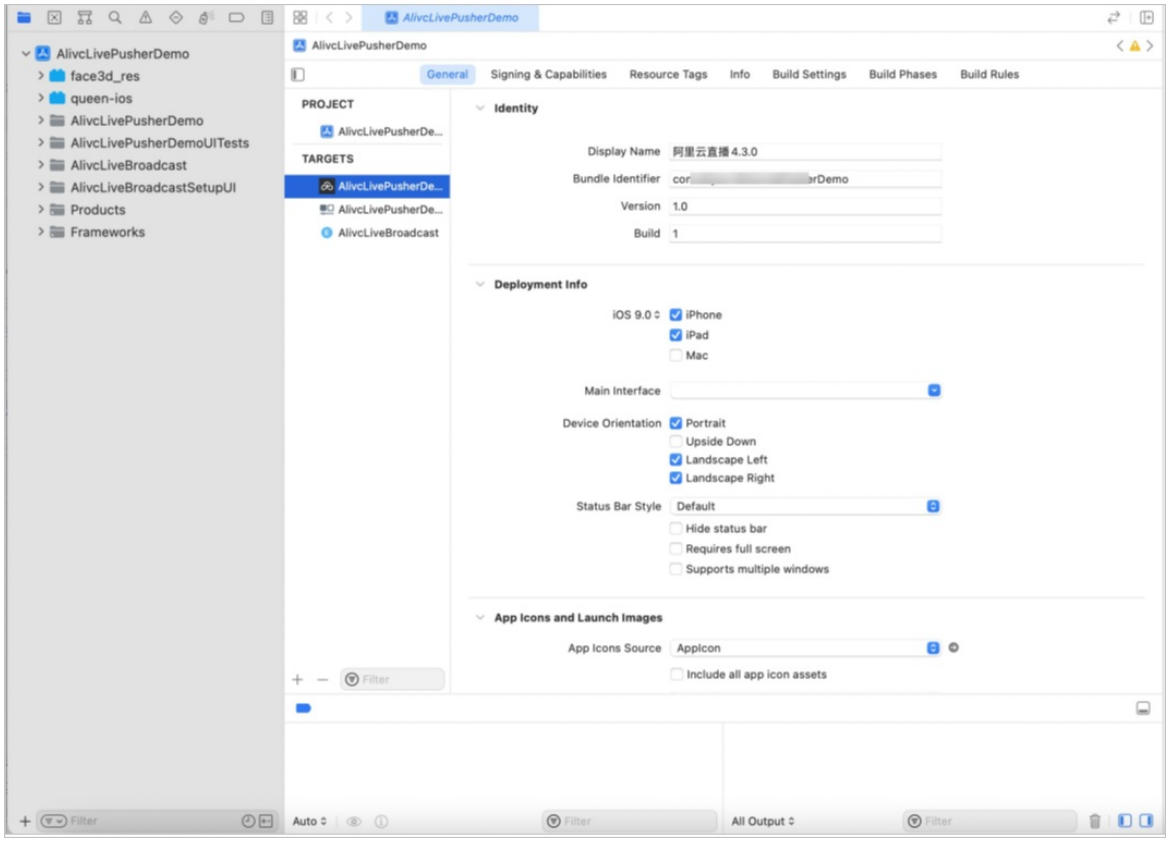
环境要求

名称	要求
系统版本	支持iOS 8.0及以上。
机器型号	支持iPhone 5s及以上。

名称	要求
CPU架构支持	ARMv7、ARM64。
集成工具	Xcode 8.0及以上版本。
bitcode	关闭。

运行推流Demo

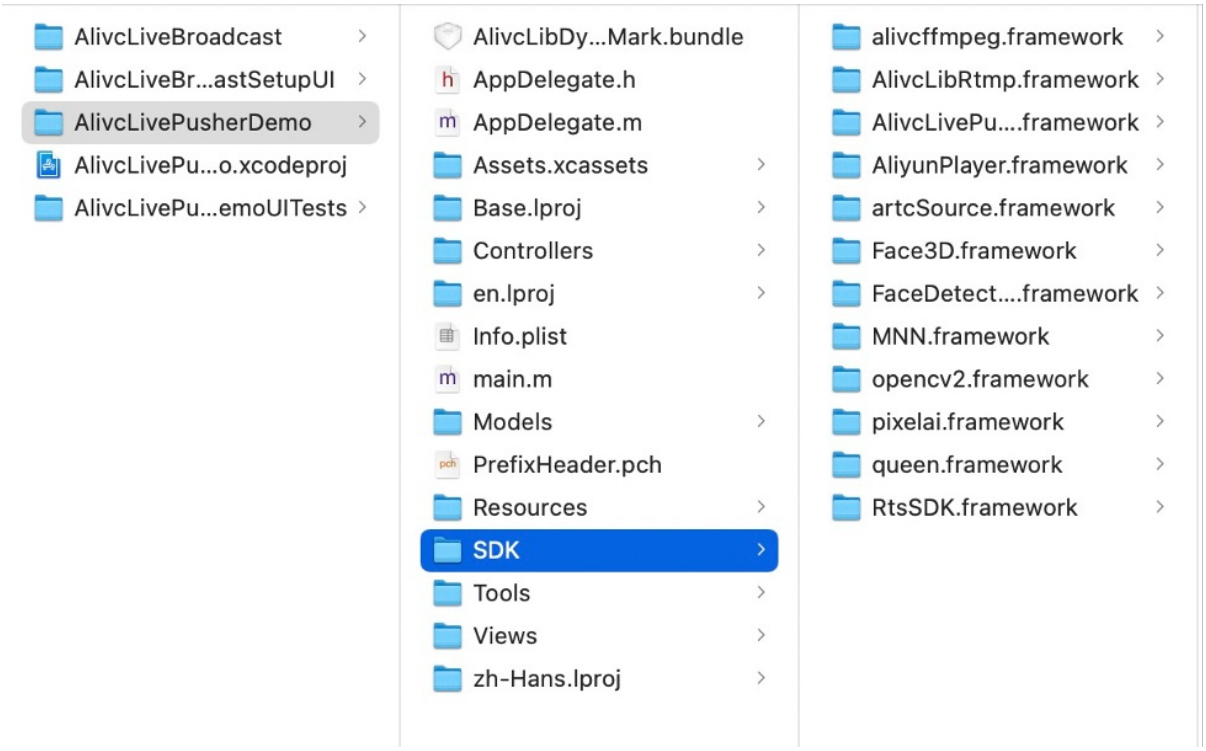
1. 请在SDK下载与历史记录中，下载最新版Demo压缩包并解压。
2. 使用xcode打开AlivcLivePusherDemo.xcodeproj工程。



3. 直接运行工程查看Demo效果。

推流URL（图示中①）中填入有效的推流RTMP地址。推流成功后，可以使用阿里云播放器SDK、FFplay、VLC等工具查看播放效果。

Demo目录结构



其中SDK文件夹内：

库文件	文件说明
• AlivcLivePusher.framework • AlivcLibRtmp.framework • RtsSDK.framework	推流SDK
• queen.framework • FaceDetection.framework • MNN.framework • Face3D.framework • opencv2.framework • pixelai.framework	美颜SDK
• AliyunPlayer.framework • alivcffmpeg.framework • artcSource.framework	播放器SDK

5.3. SDK集成

通过阅读本文，您可以快速了解如何集成iOS端推流SDK。

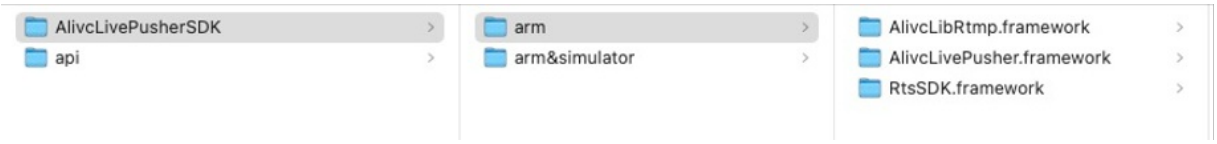
集成环境

名称	要求
系统版本	支持iOS 8.0及以上。
机器型号	支持iPhone 5s及以上。
CPU架构支持	ARMv7、ARM64。
集成工具	Xcode 8.0及以上版本。
bitcode	关闭。

推流SDK下载

说明 每个版本均包含arm和arm&simulator两套SDK，arm仅支持真机调试。arm和simulator支持真机+模拟器调试。项目在release上线的时候必须使用arm版本。

在SDK下载与历史记录下载对应版本的iOS端推流SDK，推流SDK包含在解压包的AlivcLivePusher文件夹中，如下图所示：



上图中的文件内容区别如下：

文件名称	文件说明
AlivcLivePusherSDK/arm	推流SDK，纯arm版本。
AlivcLivePusherSDK/arm&simulator	推流SDK，arm+模拟器版本。

推流SDK集成

通过手动或Pod方式集成推流SDK后，您还需要添加请求权限、关闭Bitcode，以及查看具体使用说明。

1. 请选择手动集成或Pod集成推流SDK。如果选择手动集成，则按照步骤2进行操作；如果选择Pod集成，则按照步骤3进行操作。
2. 手动集成：
    i. 新建SDK测试工程，App > DemoPush。

ii. 分别将以下文件拖入您的Xcode工程中：

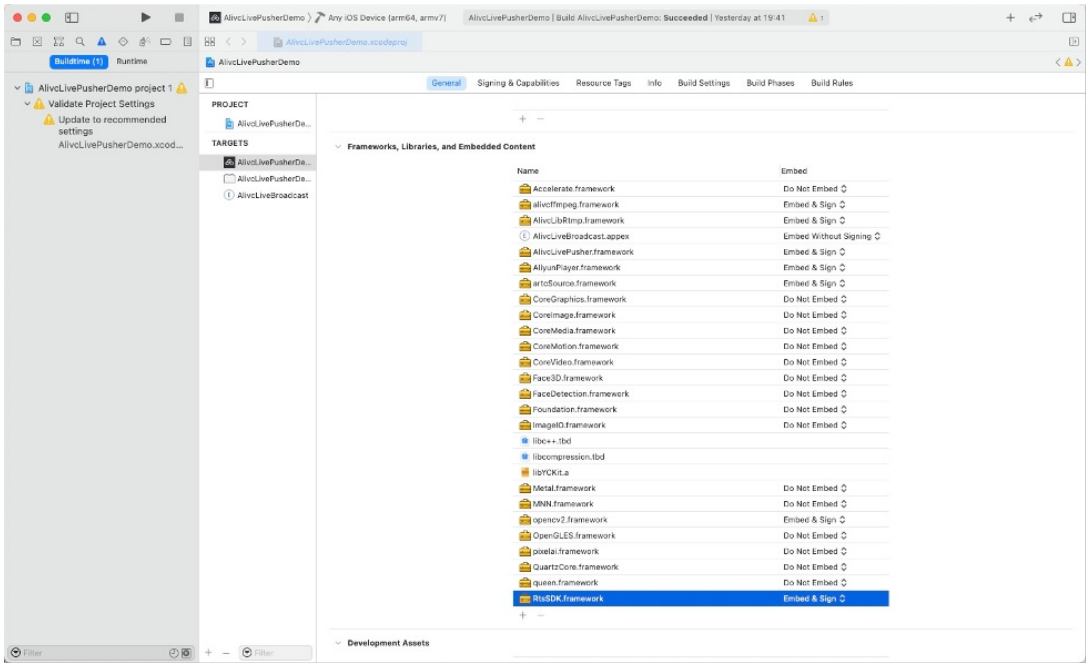
- AlivcLibRtmp.framework
- AlivcLivePusher.framework
- RtsSDK.framework

如需使用美颜和贴纸特效，还需拖入以下文件：

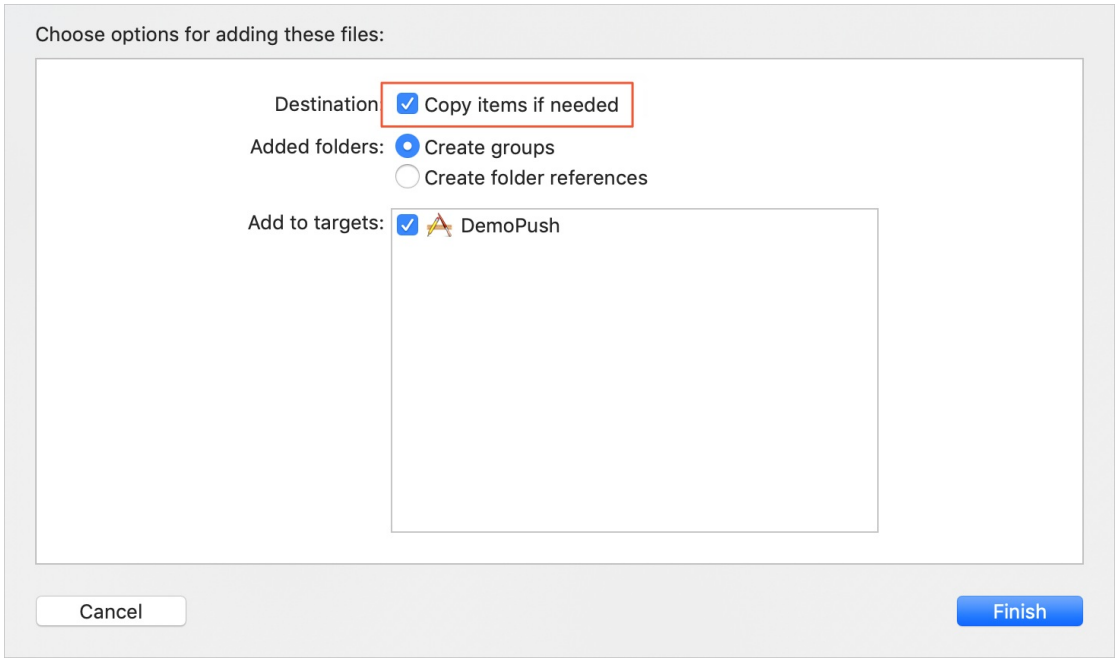
- queen.framework
- openvc2.framework
- Face3D.framework
- FaceDetection.framework
- MNN.framework
- pixelai.framework

如需依赖播放器SDK的版本，还需拖入以下文件：

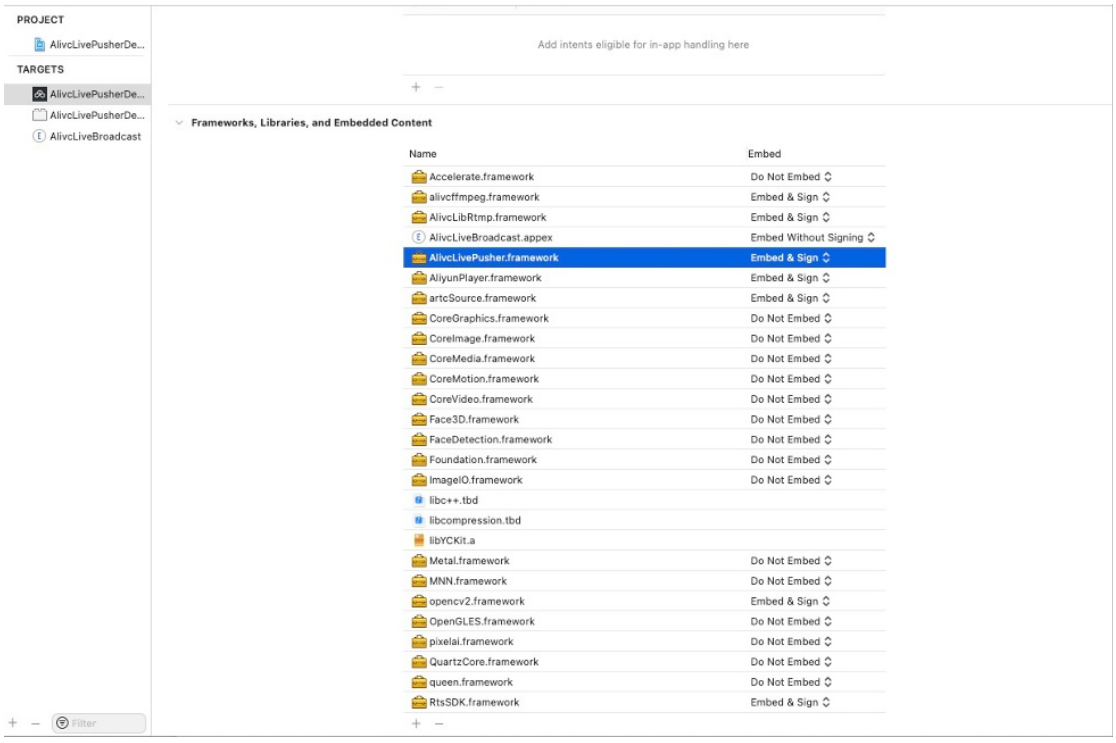
- AliyunPlayer.framework
- artcSource.framework
- alivcffmpeg.framework



iii. 勾选Copy items if needed，并单击Finish。



iv. 成功导入SDK后，在Xcode > General > Embedded Binaries中添加SDK依赖。



3. Pod集成：

i. 在终端窗口中输入以下命令，安装CocoaPods。注：请提前在Mac中安装Ruby环境。

```
sudo gem install cocoapods
```

ii. 进入项目所在路径，在终端窗口中输入以下命令，创建Podfile文件。

```
pod init
```

iii. 编辑Podfile文件，添加直播SDK依赖。如需直播播放，可以添加播放器SDK依赖。

```
# 推流SDK Pod (live pusher pod)
pod 'AlivcLivePusher', '~> 4.3.1'
pod 'RtsSDK', '~> 1.8'
# 播放器SDK Pod (live player pod)
pod 'AliPlayerSDK_iOS', '~> 5.4.2'
pod 'AliPlayerSDK_iOS_ARTC', '~> 5.4.2'
```

iv. 在终端窗口中输入以下命令。pod命令执行完成后，会生成集成了的SDK的.xcworkspace后缀的工程文件，双击打开即可更新并安装SDK。

```
pod install
```

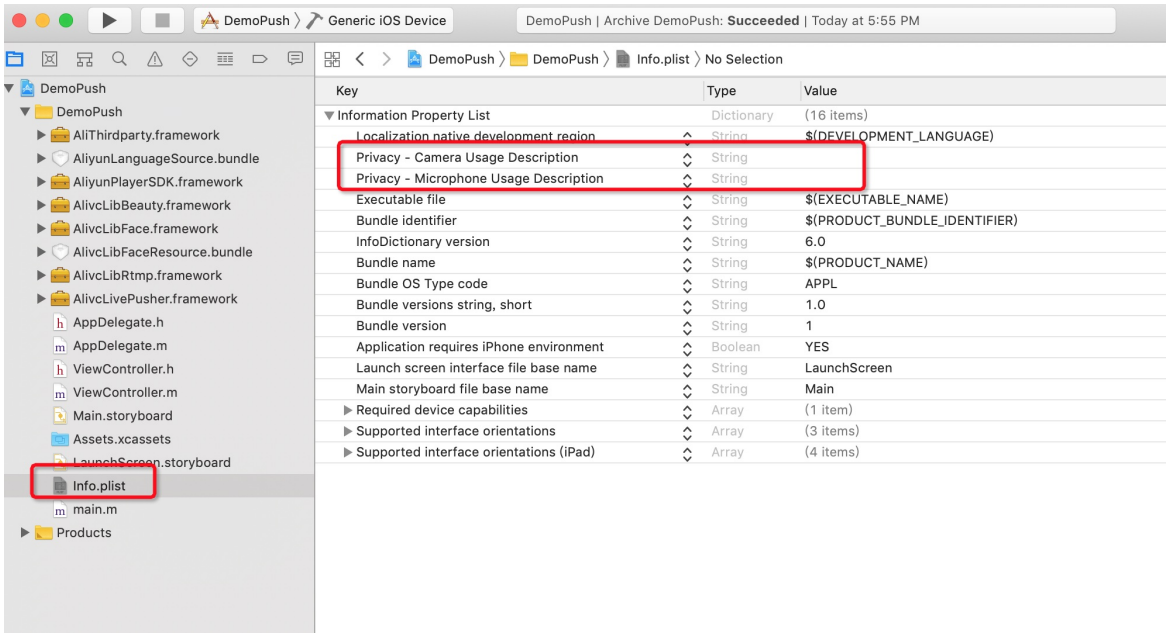
或者

```
pod update
```

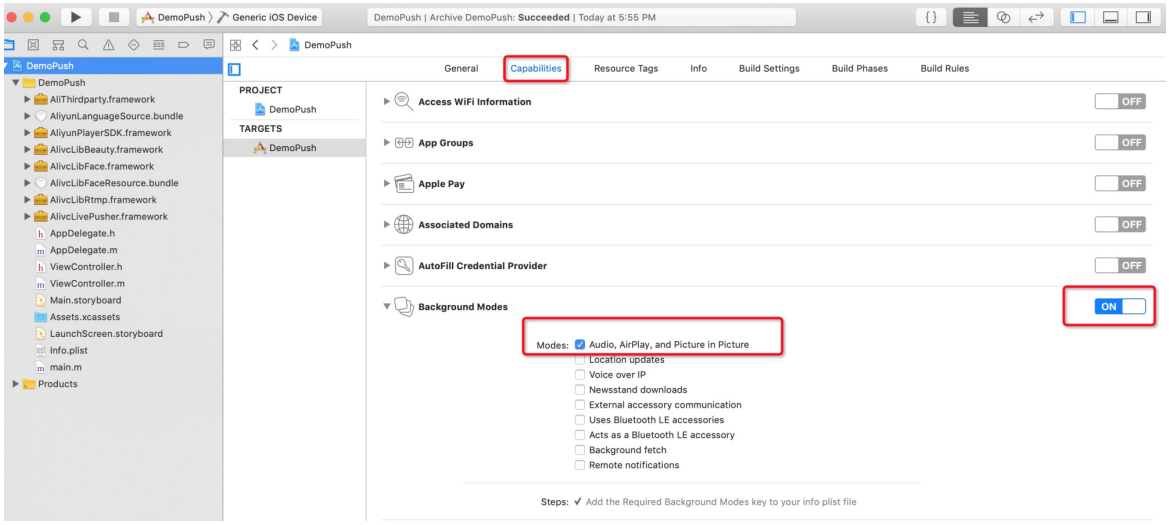
4. 添加请求权限。

> **说明** 请务必添加录音权限和相机权限。

在Info.plist文件中添加摄像头和麦克风权限Privacy - Camera Usage Description、Privacy - Microphone Usage Description。

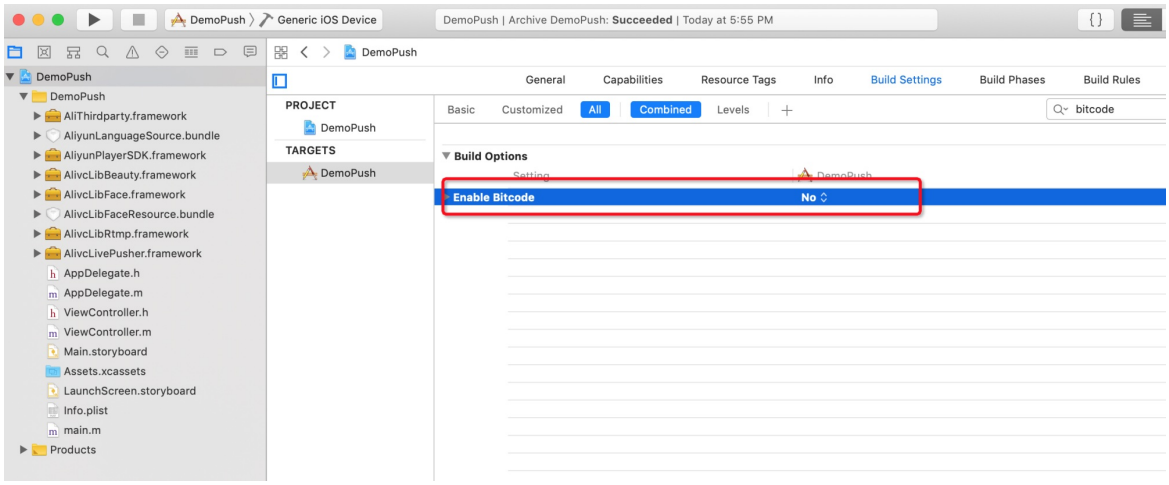


如果需要App在后台继续推流，需要打开后台音频采集模式，如图所示，勾选Audio，AirPlay，and Picture in Picture即可。



5. 关闭Bitcode。

由于SDK不支持Bitcode，所以需要在工程中关闭Bitcode选项，如图所示，将Enble Bitcode置为NO状态即可。



6. 具体使用说明。
   - API的详细注释说明，请参见V4.3.1接口说明（iOS），或查看SDK包里的API文件夹。
   - 具体SDK的API使用示例代码，请参见功能使用。

5.4. 功能使用

本文详细说明iOS端推流SDK API、SDK的基本使用流程，以及相关功能的使用示例。

> 说明 如果您需要使用移动端进行推流，详细操作请参见推流、拉流与播流。

iOS推流SDK特性

- 支持RTMP推流协议。
- 使用视频H.264编码以及音频AAC编码。
- 支持码控、分辨率、显示模式等自定义配置。
- 支持多种摄像头相关操作。
- 支持实时美颜和自定义美颜效果调节。
- 支持增、删动态贴纸实现动态水印效果。
- 支持录屏直播。
- 支持自定义YUV、PCM等外部音视频输入。
- 支持多路混流功能。
- 支持纯音视频推流以及后台推流。
- 支持背景音乐及其相关操作。
- 支持视频截图功能。
- 支持自动重连、异常处理。
- 支持音频3A算法。

功能限制

使用iOS推流SDK需注意以下限制：

- 您只能在推流之前设置横竖屏模式，不支持在直播的过程中实时切换。
- 在推流设定为横屏模式时，需设定界面为不允许自动旋转。
- 在硬编模式下，考虑编码器兼容问题分辨率会使用16的倍数，如设定为540P，则输出的分辨率为544*960，在设置播放器视图大小时需按输出分辨率等比缩放，避免黑边等问题。

推流SDK使用流程

SDK的基本使用流程如下：

步骤	描述	操作指引及代码示例
一、配置推流参数	完成推流基本配置、码率控制配置、分辨率自适应配置、美颜功能配置等。	配置推流参数
二、使用推流SDK推流	初始化SDK、注册推流回调、创建预览视图后可以开始推流。用户可以根据业务需求添加推流控制、设置背景音乐、摄像头、直播答题、外部音频、动态贴纸等。	使用推流SDK推流
三、设置录屏推流（按需）	如需使用录屏推流，可以实现录屏推流相关的配置。	设置录屏推流

配置推流参数

您可以使用AlivcLivePushConfig配置推流参数，每个参数有一个对应的默认值。关于默认值和参数范围，请参见V4.3.0接口说明（iOS）或注释。

> **说明** 如需在推流过程中实时修改参数，请参见AlivcLivePusher提供的属性和方法。

1. 基本推流配置。

   在需要使用推流器的ViewController中引用头文件 `#import <AlivcLivePusher/AlivcLivePusherHeader.h>` ，示例代码如下：

```
AlivcLivePushConfig *config = [[AlivcLivePushConfig alloc] init];//初始化推流配置类，也可使用initWithResolution来初始化。
config.resolution = AlivcLivePushResolution540P;//默认为540P，最大支持720P
config.fps = AlivcLivePushFPS20; //建议用户使用20fps
config.enableAutoBitrate = true; // 打开码率自适应，默认为true
config.videoEncodeGop = AlivcLivePushVideoEncodeGOP_2;//默认值为2，关键帧间隔越大，延时越高。建议设置为1-2。
config.connectRetryInterval = 2000; // 单位为毫秒，重连时长2s，重连间隔设置不小于1秒，建议使用默认值即可。
config.previewMirror = false; // 默认为false，正常情况下都选择false即可。
config.orientation = AlivcLivePushOrientationPortrait; // 默认为竖屏，可设置home键向左或向右横屏。
```

> 说明
> - 综合手机性能和网络带宽要求，建议您将分辨率设置为540P（主流移动直播App基本都采用540P）。
> - 基本推流配置对应参数都有默认值，建议采用默认值，即您可以进行简单初始化，不做配置。
> - 关闭自适应码率后，码率将固定在初始码率，不会在设定的目标码率和最小码率之间自适应调整。如果网络情况不稳定，可能造成播放卡顿，请慎用。

2. 配置码率控制。

推流SDK提供以下码率控制模式，请根据实际需求修改参数值。

码率控制模式	描述	示例代码
AlivcLivePushQualityModeResolutionFirst	清晰度优先模式。SDK内部会对码率参数进行配置，优先保障推流视频的清晰度。	config.qualityMode = AlivcLivePushQualityModeResolutionFirst；//默认为清晰度优先模式，可设置为流畅度优先模式和自定义模式。
AlivcLivePushQualityModeFluencyFirst	流畅度优先模式。SDK内部会对码率参数进行配置，优先保障推流视频的流畅度。	config.qualityMode = AlivcLivePushQualityModeFluencyFirst；//默认为流畅度优先模式，可设置为清晰度优先模式和自定义模式。

AlivcLivePushQualityModeCustom	自定义模式。SDK会根据开发者设置的码率进行配置。设置为自定义模式时，需要自己定义初始码率、最小码率和目标码率。
	○ 初始码率：开始直播时的码率。
	○ 最小码率：当网络较差时，码率会逐步减低到最小码率，以减少视频的卡顿。
	○ 目标码率：当网络较好时，码率会逐步提高到目标码率，以提高视频清晰度。
		config.qualityMode = AlivcLivePushQualityModeCustom//设置为自定义模式
		config.targetVideoBitrate = 1400; //目标码率1400Kbps
		config.minVideoBitrate = 600; //最小码率600Kbps
		config.initialVideoBitrate = 1000; //初始码率1000Kbps

> 说明
> - 选择清晰度优先或流畅度优先模式时，不需设置初始码率、最小码率和目标码率（initialVideoBitrate、minVideoBitrate、targetVideoBitrate）。推流SDK内部策略会自动保障在网络抖动情况下优先考虑视频清晰度或流畅度。
> - 选择自定义码率时，请参考阿里云推荐设置配置对应码率。推荐设置请参考下表内容。

自定义码率控制推荐设置（画质优先）

分辨率	初始码率 initialVideoBitrate	最小码率 minVideoBitrate	目标码率 targetVideoBitrate
360P	600	300	1000
480P	800	300	1200
540P	1000	600	1400
720P	1500	600	2000
1080P	1800	1200	2500

自定义码率控制推荐设置（流畅度优先）

分辨率	初始码率 initialVideoBitrate	最小码率 minVideoBitrate	目标码率 targetVideoBitrate
360P	400	200	600
480P	600	300	800
540P	800	300	1000
720P	1000	300	1200
1080P	1500	1200	2200

3. 配置分辨率自适应。

   开启动态调整推流分辨率功能后，当网络较差时会自动降低分辨率以提高视频的流畅度和清晰度。示例代码如下：

```
config.enableAutoResolution = YES; // 打开分辨率自适应，默认为NO
```

> 注意
>
> - 某些播放器可能不支持动态分辨率，如果您需要使用分辨率自适应功能，建议使用阿里云播放器。
> - 分辨率自适应只有在清晰度优先或流畅度优先时才会生效（AlivcQualityModeEnum参数配置），自定义模式时无效。

4. 配置美颜功能。

   阿里云推流SDK提供两种美颜模式：基础美颜和高级美颜。基础美颜支持美白、磨皮和红润。高级美颜支持基于人脸识别的美白、磨皮、红润、大眼、小脸、瘦脸等功能。此功能由简介提供，使用示例代码如下：

```
#pragma mark - "美颜类型和美颜参数API"/**
 * @brief 打开或者关闭某个美颜类型
 * @param type QueenBeautyType 类型的一个值
 * @param isOpen YES: 打开，NO:关闭
 *
 */
```

```
 /
- (void)setQueenBeautyType:(kQueenBeautyType)type enable:(BOOL)isOpen;
/**
 * @brief 设置美颜参数
 * @param param 美颜参数类型，QueenBeautyParams中的一个
 * @param value 需要设置的数值，值的范围都是[0,1]，小于0的置0，大于1的置1
 */
- (void)setQueenBeautyParams:(kQueenBeautyParams)param
value:(float)value;
#pragma mark - "滤镜相关API"
/**
 * @brief 设置滤镜图片,设置滤镜图片前需要将kQueenBeautyTypeLUT打开
 * @param imagePath 所要设置的滤镜图片的地址
 */
- (void)setLutImagePath:(NSString *)imagePath;
#pragma mark - "美型相关API"
/**
 * @brief 设置美型类型，设置前需要将kQueenBeautyTypeFaceShape打开
 * @param faceShapeType 需要设置美型的类型，参考QueenBeautyFaceShapeType
 * @param value 需要设置的值
 */
- (void)setFaceShape:(kQueenBeautyFaceShapeType)faceShapeType
value:(float)value;
#pragma mark - "美妆相关api"
/**
 * @brief 设置美妆类型和图片素材路径，设置美妆需要将kQueenBeautyTypeMakeup打开
 * @param makeupType 美妆类型
 * @param imagePaths 美妆素材地址集合
 * @param blend 混合类型
 */
- (void)setMakeupWithType:(kQueenBeautyMakeupType)makeupType
paths:(NSArray<NSString *> *)imagePaths
blendType:(kQueenBeautyBlend)blend;
/**
 * @brief 设置美妆类型和图片素材路径
 * @param makeupType 美妆类型
 * @param imagePaths 美妆素材地址集合
 * @param blend 混合类型
 * @param fps 对应的帧率
 */
- (void)setMakeupWithType:(kQueenBeautyMakeupType)makeupType
paths:(NSArray<NSString *> *)imagePaths
blendType:(kQueenBeautyBlend)blend fps:(int)fps;
/**
 * @brief 设置美妆透明度，可指定性别
 * @param makeupType 美妆类型
 * @param isFeMale 是否是女性，女性:YES，男性:NO
 * @param alpha 妆容透明度
 */
- (void)setMakeupAlphaWithType:(kQueenBeautyMakeupType)makeupType
female:(BOOL)isFeMale alpha:(float)alpha;
/**
 * @brief 设置美妆类型的混合类型
 * @param makeupType 美妆类型
 * @param blend 混合类型
```

```
 */
- (void)setMakeupBlendWithType:(kQueenBeautyMakeupType)makeupType
blendType:(kQueenBeautyBlend)blend;
/**
 * @brief 清除所有美妆
 */
- (void)resetAllMakeupType;
```

5. 配置图片推流。

   为了更好的用户体验，SDK提供了后台图片推流和码率过低时进行图片推流的设置。当SDK退至后台时默认暂停推流视频，只推流音频，此时可以设置图片来进行图片推流和音频推流。例如，在图片上提醒用户*主播离开片刻，稍后回来*。示例代码如下：

   ```
   config.pauseImg = [UIImage imageNamed:@"图片.png"];//设置用户后台推流的图片
   ```

   另外，当网络较差时您可以根据自己的需求设置推流一张静态图片。设置图片后，SDK检测到当前码率较低时，会推流此图片，避免视频流卡顿。示例代码如下：

   ```
   config.networkPoorImg = [UIImage imageNamed:@"图片.png"];//设置网络较差时推流
   ```

6. 配置水印。

   推流SDK提供了添加水印功能，并且最多支持添加多个水印，水印图片必须为PNG格式图片。示例代码如下：

   ```
   NSString *watermarkBundlePath = [[NSBundle mainBundle] pathForResource:
   [NSString stringWithFormat:@"watermark"] ofType:@"png"];//设置水印图片路径
   [config addWatermarkWithPath: watermarkBundlePath
       watermarkCoordX:0.1
       watermarkCoordY:0.1
       watermarkWidth:0.3];//添加水印
   ```

   > **说明**
   >
   > - coordX、coordY、width为相对值，例如watermarkCoordX:0.1表示水印的x值为推流画面x轴的10%位置，如果推流分辨率为540*960，则水印x值为54。
   > - 水印图片的高度，按照水印图片的真实宽高与输入的width值等比缩放。
   > - 要实现文字水印，可以先将文字转换为图片，再使用此接口添加水印。
   > - 为了保障水印显示的清晰度与边缘平滑，请您尽量使用和水印输出尺寸相同大小的水印源图片。如输出视频分辨率544*940，水印显示的w是0.1f，则尽量使用水印源图片宽度在544*0.1f=54.4左右。

7. 配置预览显示模式。

   推流SDK支持三种预览模式，预览显示模式不影响推流。

   - ALIVC_LIVE_PUSHER_PREVIEW_SCALE_FILL：预览显示时，铺满窗口。当视频比例和窗口比例不一致时，预览会有变形。
   - ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FIT：预览显示时，保持视频比例。当视频比例与窗口比例不一致时，预览会有黑边。（默认）
   - ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FILL：预览显示时，剪切视频以适配窗口比例。当视频比例和窗口比例不一致时，预览会裁剪视频。

示例代码如下：

```
mAlivcLivePushConfig.setPreviewDisplayMode(AlivcPreviewDisplayMode.ALIVC_LIVE_PUSHER_PREVIEW_ASPECT_FIT);
```

> 说明
> - 三种模式可以在AlivcLivePushConfig中设置，也可以在预览中和推流中通过API setpreviewDisplayMode进行动态设置。
> - 本设置只对预览显示生效，实际推出的视频流的分辨率和AlivcLivePushConfig中预设置的分辨率一致，并不会因为更改预览显示模式而变化。预览显示模式是为了适配不同尺寸的手机，您可以自由选择预览效果。

使用推流SDK推流

AlivcLivePusher为推流SDK的核心类，提供摄像头预览、推流回调、推流控制、推流过程中的参数调节等功能。通过下文操作步骤，您可以了解如何使用推流核心接口。

1. 初始化。

   在配置好推流参数后，可以使用推流SDK的initWithConfig方法进行初始化。示例代码如下：

   ```
   self.livePusher = [[AlivcLivePusher alloc] initWithConfig:config];
   ```

   > 说明 AlivcLivePusher目前不支持多实例，所以一个init必须对应有一个destory。

2. 注册推流回调。

   推流回调分为三种：

   - Info：主要做提示和状态检测使用。
   - Error：错误回调。
   - Network：主要为网络相关。

   注册delegate可接收对应的回调。示例代码如下：

   ```
   [self.livePusher setInfoDelegate:self];
   [self.livePusher setErrorDelegate:self];
   [self.livePusher setNetworkDelegate:self];
   ```

3. 开始预览。

   livePusher对象初始化完成之后，可以进行开始预览操作。预览时需要传入摄像头预览的显示view（继承自UIView）。示例代码如下：

   ```
   [self.livePusher startPreview:self.view];
   ```

4. 开始推流。

   预览成功后才可以开始推流，因此需监听AlivcLivePusherInfoDelegate的onPreviewStarted回调，在回调里面添加如下代码。

   ```
   [self.livePusher startPushWithURL:@"推流测试地址(rtmp://.....)"];
   ```

> 说明
> - 推流SDK同时提供了异步方法，可调用startPushWithURLAsync来实现。
> - 推流SDK支持RTMP的推流地址，阿里云推流地址获取请参见推流地址和播放地址。
> - 使用正确的推流地址开始推流后，可用播放器（阿里云播放器、FFplay、VLC等）进行拉流测试，拉流地址获取请参见推流地址和播放地址。

5. 设置其他推流控制。

   推流控制主要包括开始推流、停止推流、停止预览、重新推流、暂停推流、恢复推流、销毁推流等操作，用户可以根据业务需求添加按钮进行操作。示例代码如下：

```
/*正在推流状态下可调用暂停推流。暂停推流后，视频预览和视频推流保留在最后一帧，音频推流继续。*/
[self.livePusher pause];
/*暂停状态下可调用恢复推流。恢复推流后，音视频预览与推流恢复正常。*/
[self.livePusher resume];
/*推流状态下可调用停止推流，完成后推流停止。*/
[self.livePusher stopPush];
/*在预览状态下才可以调用停止预览，正在推流状态下，调用停止预览无效。预览停止后，预览画面定格在最后一帧。*/
[self.livePusher stopPreview];
/*推流状态下或者接收到所有Error相关回调状态下可调用重新推流，且Error状态下只可以调用此接口(或者reconnectPushAsync重连)或者调用destory销毁推流。完成后重新开始推流，重启ALivcLivePusher内部的一切资源，包括预览、推流等等restart。*/
[self.livePusher restartPush];
/*推流状态下或者接收到AlivcLivePusherNetworkDelegate相关的Error回调状态下可调用此接口,且Error状态下只可以调用此接口(或者restartPush重新推流)或者调用destory销毁推流。完成后推流重连，重新链接推流RTMP。*/
[self.livePusher reconnectPushAsync];
/*销毁推流后，推流停止，预览停止，预览画面移除。AlivcLivePusher相关的一切资源销毁。*/
[self.livePusher destory];
self.livePusher = nil;
/*获取推流状态。*/
AlivcLivePushStatus status = [self.livePusher getLiveStatus];
```

6. 美颜实时调整。

   推流SDK支持在推流时实时调整美颜参数，开启美颜开关，分别调整对应的参数值，此功能由智能美化特效提供，示例代码如下：

```
[_queenEngine setQueenBeautyType:kQueenBeautyTypeSkinBuffing enable:YES];
[_queenEngine setQueenBeautyType:kQueenBeautyTypeSkinWhiting enable:YES];
[_queenEngine setQueenBeautyParams:kQueenBeautyParamsWhitening value:0.8f];
[_queenEngine setQueenBeautyParams:kQueenBeautyParamsSharpen value:0.6f];
[_queenEngine setQueenBeautyParams:kQueenBeautyParamsSkinBuffing value:0.6];
```

7. 设置背景音乐。

   推流SDK提供了背景音乐播放、混音、降噪、耳返、静音等功能，背景音乐相关接口在开始预览之后才可调用。示例代码如下：

```
/*开始播放背景音乐。*/
[self.livePusher startBGMWithMusicPathAsync:musicPath];
/*停止播放背景音乐。若当前正在播放BGM，并且需要切换歌曲，只需要调用开始播放背景音乐接口即可，无需停止当前正在播放的背景音乐。*/
[self.livePusher stopBGMAsync];
/*暂停播放背景音乐，背景音乐开始播放后才可调用此接口。*/
[self.livePusher pauseBGM];
/*恢复播放背景音乐，背景音乐暂停状态下才可调用此接口。*/
[self.livePusher resumeBGM];
/*开启循环播放音乐*/
[self.livePusher setBGMLoop:true];
/*设置降噪开关。打开降噪后，将对采集到的声音中非人声的部分进行过滤处理。可能存在对人声稍微抑制作用，建议让用户自由选择是否开启降噪功能，默认不使用*/
[self.livePusher setAudioDenoise:true];
/*设置耳返开关。耳返功能主要应用于KTV场景。打开耳返后，插入耳机将在耳机中听到主播说话声音。关闭后，插入耳机无法听到人声。未插入耳机的情况下，耳返不起作用。*/
[self.livePusher setBGMEarsBack:true];
/*混音设置，提供背景音乐和人声采集音量调整。*/
[self.livePusher setBGMVolume:50];//设置背景音乐音量
[self.livePusher setCaptureVolume:50];//设置人声采集音量
/*设置静音。静音后音乐声音和人声输入都会静音。要单独设置音乐或人声静音可以通过混音音量设置接口来调整。*/
[self.livePusher setMute:isMute?true:false];
```

8. 摄像头相关操作。

   您只能在开始预览之后调用摄像头相关操作，包括推流状态、暂停状态、重连状态等，可操作摄像头切换、闪关灯、焦距、变焦和镜像设置等。未开始预览状态下调用如下接口无效。示例代码如下：

```
/*切换前后摄像头*/
[self.livePusher switchCamera];
/*开启/关闭闪光灯，在前置摄像头时开启闪关灯无效*/
[self.livePusher setFlash:false];
/*焦距调整，即可实现采集画面的缩放功能。传入参数为正数，则放大焦距，传入参数为负数则缩小焦距。*/
CGFloat max = [_livePusher getMaxZoom];
[self.livePusher setZoom:MIN(1.0, max)];
/*手动对焦。手动聚焦需要传入两个参数:1.point 对焦的点（需要对焦的点的坐标);2.autoFocus 是否需要自动对焦，该参数仅对调用接口的该次对焦操作生效。后续是否自动对 焦沿用上述自动聚焦接口设置值。*/
[self.livePusher focusCameraAtAdjustedPoint:CGPointMake(50, 50) autoFocus:true];
/*设置是否自动对焦*/
[self.livePusher setAutoFocus:false];
/*镜像设置。镜像相关接口有两个，PushMirror推流镜像和PreviewMirror预览镜像。PushMirror设置仅对播放画面生效，PreviewMirror仅对预览画面生效，两者互不影响。*/
[self.livePusher setPushMirror:false];
[self.livePusher setPreviewMirror:false];
```

9. 配置直播答题功能。

   直播答题功能可以通过在直播流里面插入SEI信息，播放器解析SEI来实现。在推流SDK里面提供了插入SEI的接口，在推流状态下，才能调用此接口。示例代码如下：

```
/*
msg: 需要插入流的SEI消息体，建议是JSON格式。阿里云播放器SDK可收到此SEI消息，解析后做具体展示。
repeatCount：发送的帧数。为了保证SEI不被丢帧，需设置重复次数，如设置100，则在接下去的100帧均插入此SEI消息。播放器会对相同的SEI进行去重处理。
delayTime：延时多少毫秒发送。
KeyFrameOnly：是否只发关键帧。
*/
[self.livePusher sendMessage:@"题目信息" repeatCount:100 delayTime:0 KeyFrameOnly:false];
```

10. 配置外部音视频输入。

    推流SDK支持将外部的音视频源输入进行推流，比如推送一个音视频文件。

    i. 在推流配置里面进行外部音视频输入配置。

    示例代码如下：

    ```
    config.externMainStream = true;//开启允许外部流输入
    config.externVideoFormat = AlivcLivePushVideoFormatYUVNV21;//设置视频数据颜色格式定义，这里设置为YUVNV21，可根据需求设置为其他格式。
    config.externMainStream = AlivcLivePushAudioFormatS16;//设置音频数据位深度格式，这里设置为S16，可根据需求设置为其他格式
    ```

    ii. 插入外部视频数据。

    示例代码如下：

    ```
    /*只支持外部视频yuv和rbg格式的连续buffer数据，才可以通过sendVideoData接口，发送视频数据buffer、长度、宽高、时间戳、旋转角度*/
    [self.livePusher sendVideoData:yuvData width:720 height:1280 size:dataSize pts:nowTime rotation:0];
    /*如果外部视频数据是CMSampleBufferRef格式，可以使用sendVideoSampleBuffer接口*/
    [self.livePusher sendVideoSampleBuffer:sampleBuffer]
    /*也可以将 CMSampleBufferRef格式转化为连续buffer后再传递给sendVideoData接口， 以下为转换的参考代码*/
    //获取samplebuffer长度
    - (int) getVideoSampleBufferSize:(CMSampleBufferRef)sampleBuffer {
    if(!sampleBuffer) {
       return 0;
    }
    int size = 0;
    CVPixelBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);
    CVPixelBufferLockBaseAddress(pixelBuffer, 0);
    if(CVPixelBufferIsPlanar(pixelBuffer)) {
      int count = (int)CVPixelBufferGetPlaneCount(pixelBuffer);
      for(int i=0; i<count; i++) {
         int height = (int)CVPixelBufferGetHeightOfPlane(pixelBuffer,i);
         int stride = (int)CVPixelBufferGetBytesPerRowOfPlane(pixelBuffer,i);
         size += stride*height;
      }
    }else {
      int height = (int)CVPixelBufferGetHeight(pixelBuffer);
      int stride = (int)CVPixelBufferGetBytesPerRow(pixelBuffer);
      size += stride*height;
    }
    CVPixelBufferUnlockBaseAddress(pixelBuffer, 0);
    return size;
    ```

```
}
//将samplebuffer转化为连续buffer
- (int) convertVideoSampleBuffer:(CMSampleBufferRef)sampleBuffer toNativeBuffer:(void*)native
Buffer
{
if(!sampleBuffer || !nativeBuffer) {
  return -1;
}
CVPixelBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);
CVPixelBufferLockBaseAddress(pixelBuffer, 0);
int size = 0;
if(CVPixelBufferIsPlanar(pixelBuffer)) {
  int count = (int)CVPixelBufferGetPlaneCount(pixelBuffer);
  for(int i=0; i<count; i++) {
     int height = (int)CVPixelBufferGetHeightOfPlane(pixelBuffer,i);
     int stride = (int)CVPixelBufferGetBytesPerRowOfPlane(pixelBuffer,i);
     void *buffer = CVPixelBufferGetBaseAddressOfPlane(pixelBuffer, i);
     int8_t *dstPos = (int8_t*)nativeBuffer + size;
     memcpy(dstPos, buffer, stride*height);
     size += stride*height;
  }
}else {
  int height = (int)CVPixelBufferGetHeight(pixelBuffer);
  int stride = (int)CVPixelBufferGetBytesPerRow(pixelBuffer);
  void *buffer = CVPixelBufferGetBaseAddress(pixelBuffer);
  size += stride*height;
  memcpy(nativeBuffer, buffer, size);
}
CVPixelBufferUnlockBaseAddress(pixelBuffer, 0);
return 0;
}
```

iii. 插入音频数据。

示例代码如下：

```
/*只支持外部pcm格式的连续buffer数据，sendPCMData，发送音频数据buffer、长度、时间戳*/
[self.livePusher sendPCMData:pcmData size:size pts:nowTime];
```

11. 动态贴纸。

推流SDK实现了在直播流中添加动态贴纸效果，使用此功能可实现动态水印效果。

i. 动态贴纸的制作可参考Demo提供的素材进行简单修改。自己制作动图贴纸的序列帧图片，并打开config.json文件自定义以下参数：

```
"du": 2.04,//播放一遍动画持续的时间
"n": "qizi",//动图的名称，制作动图时文件夹以动图名称命名，每一张图片以动图名称+序号命名，比如qizi0
"c": 68.0,//动画帧数，即一个完整动画的图片数量
"kerneframe": 51,//关键帧，即指定哪一张图片为关键帧，比如demo中指定第51帧为关键帧（需确保51帧是存在的）
"frameArry": [
  {"time":0,"pic":0},
  {"time":0.03,"pic":1},
  {"time":0.06,"pic":2},
  ],
//动画参数，上述参数即表示第0秒显示第一帧（qizi0），第0.03秒显示第二帧（qizi1）...以此规则填写所有帧的动画
```

> **说明** 其他字段可以直接使用demo提供的json文件中的内容，无需修改。

ii. 添加动态贴纸。

示例代码如下：

```
/**
* 添加动态贴纸
* @param path 贴纸文件路径，必须含config.json
* @param x 显示起始x位置(0~1.0f)
* @param y 显示起始y位置(0~1.0f)
* @param w 显示宽度(0~1.0f)
* @param h 显示高度(0~1.0f)
* @return id 贴纸id，删除贴纸时需设置id
*/
[self.livePusher addDynamicWaterMarkImageDataWithPath: "贴纸路径" x:0.2f y:0.2f w:0.2f h:0.2f];
```

iii. 删除动态贴纸。

示例代码如下：

```
[self.livePusher removeDynamicWaterMark:id];
```

12. 调试工具。

SDK提供UI调试工具DebugView。DebugView为可移动的全局悬浮窗，添加后始终悬浮在视图的最上层。内含推流日志查看、推流性能参数实时检测、推流主要性能折线图表等debug功能。

> **说明** 在您的release版本下，请勿调用添加DebugView的接口。

示例代码如下：

```
[AlivcLivePusher showDebugView];//打开调试工具
```

13. 其他接口的使用。

```
/*在自定义模式下，用户可以实时调整最小码率和目标码率。*/
[self.livePusher setTargetVideoBitrate:800];
[self.livePusher setMinVideoBitrate:200]
/*获取是否正在推流的状态*/
BOOL isPushing = [self.livePusher isPushing];
/*获取推流地址*/
NSString *pushURLString = [self.livePusher getPushURL];
/*获取推流性能调试信息。推流性能参数具体参数和描述参考API文档或者接口注释。*/
AlivcLivePushStatsInfo *info = [self.livePusher getLivePushStatusInfo];
/*获取版本号。*/
NSString *sdkVersion = [self.livePusher getSDKVersion];
/*设置log级别，根据需求过滤想要的调试信息*/
[self.livePusher setLogLevel:(AlivcLivePushLogLevelDebug)];
```

设置录屏推流

ReplayKit是iOS 9引入的支持屏幕录制功能。iOS 10在ReplayKit中新增了调用第三方App扩展来直播屏幕内容的功能。在iOS 10及以上系统中，使用推流SDK配合Extension录屏进程，可以实现录屏直播。

iOS为了保证系统运行流畅，给Extension录屏进程的资源相对较少，Extension录屏进程内存占用过大会被系统强杀退出。为了解决Extension录屏进程内存限制，推流SDK将录屏推流分成Extension录屏进程（Extension App）和主App进程（Host App）。Extension录屏进程负责抓取屏幕内容，并通过进程间通信将屏幕内容发送给主App进程。主App进程创建推流引擎AlivcLivePusher，并将屏幕数据推送到远端。由于在主App进程中完成整个推流过程，因此麦克风的采集和发送可以放到主App进程中进行，Extension录屏进程只负责屏幕内容采集。

> **注意** 推流SDK Demo是通过App Group实现Extension录屏进程和主App进程之间的进程通信，并将该部分逻辑封装在了 `AlivcLibReplayKitExt.framework` 中。

iOS上实现屏幕推流，Extension录屏进程由系统在录屏需要的时候创建，并负责接收系统采集到屏幕图像。需要如下对接操作步骤：

1. 创建App Group。

   需登录Apple Developer，完成以下操作：

   i. 在Certificates, IDs & Profiles 页面中注册App Group，具体操作步骤可以参考注册App Group。

   ii. 回到**Identifier**页面，选择**App IDs**，然后单击您的App ID（主App进程与Extension录屏进程的App ID需要进行同样的配置）启用App Group功能。具体操作步骤可以参考启用App Group。

   iii. 完成后重新下载对应的Provisioning Profile并配置到XCode中。

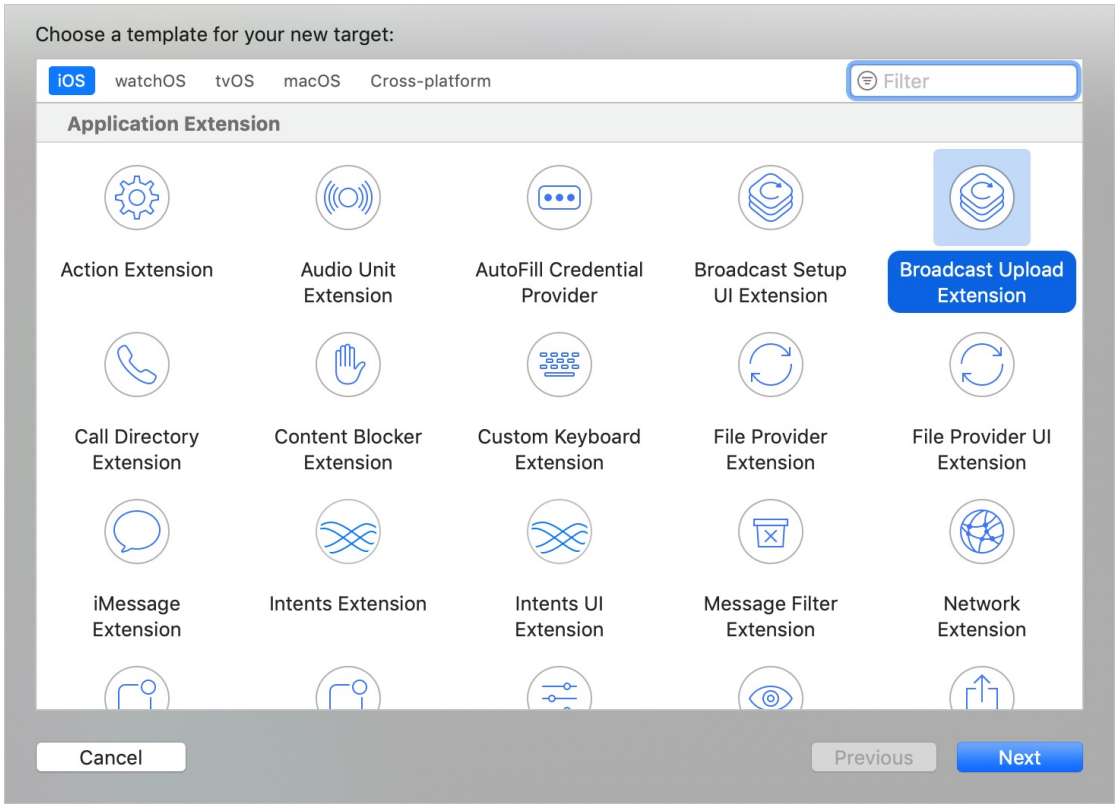
   操作正确完成后Extension录屏进程可以和主App进程之间进行进程通信。

   > **说明** 创建App Group完成后需保存App Group Identifier值，作为后续步骤的输入内容。

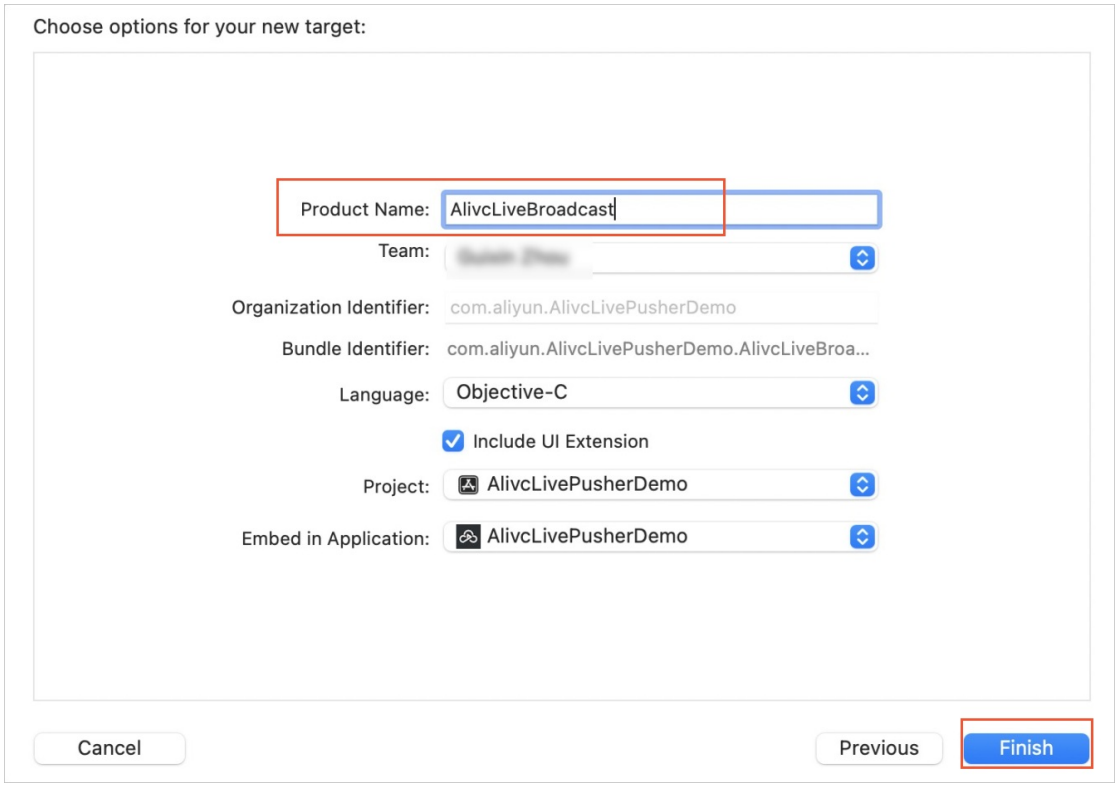
2. 创建Extension录屏进程。

   iOS推流SDK Demo中实现了支持录屏直播的App扩展AlivcLiveBroadcast和AlivcLiveBroadcastSetupUI。App中具体创建Extension录屏进程如下：

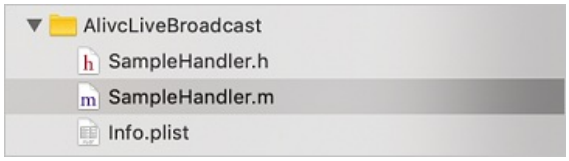
i. 在现有工程选择**New > Target...**，选择**Broadcast Upload Extension**，如下图：



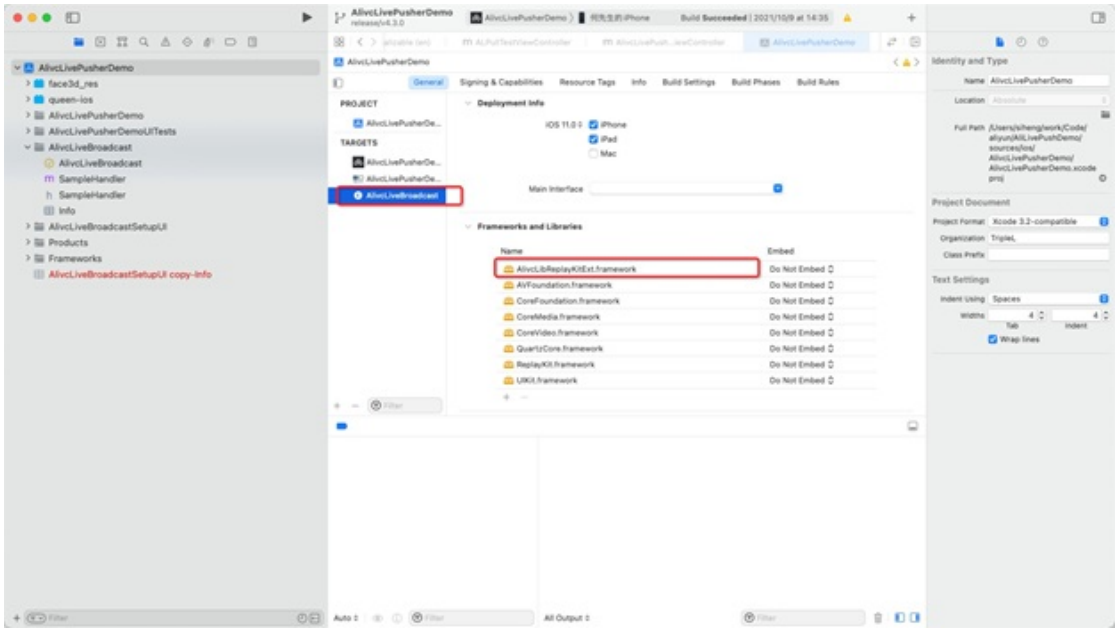
ii. 修改**Product Name**，勾选**Include UI Extension**，单击**Finish**创建直播扩展和直播UI，如下图：



iii. 配置直播扩展Info.plist，在新创建的Target中，Xcode会默认创建名为*SampleHandler*的头文件和源文件，如下图：



将 AlivcLibReplayKitExt.framework 拖到工程中，使得Extension Target依赖它。



替换SampleHandler.m中的代码成如下代码（需将代码中的KAPP Group替换成上文第一步骤创建的App Group Identifier）。示例代码如下：

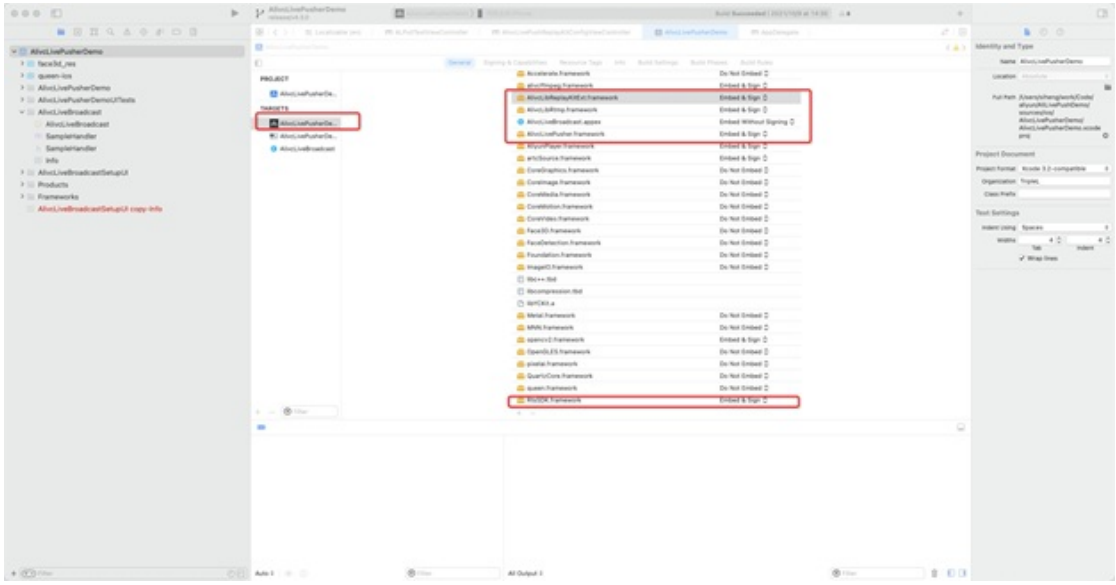
```
#import "SampleHandler.h"
#import <AlivcLibReplayKitExt/AlivcLibReplayKitExt.h>
@implementation SampleHandler
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *>
*)setupInfo {
  //User has requested to start the broadcast. Setup info from the UI extension can
be supplied but optional.
  [[AlivcReplayKitExt sharedInstance] setAppGroup:kAPPGROUP];
}
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:(RPSampleBufferType)s
ampleBufferType {
  if (sampleBufferType != RPSampleBufferTypeAudioMic) {
    //声音由主App进程采集发送
    [[AlivcReplayKitExt sharedInstance] sendSampleBuffer:sampleBuffer withType:sampleBufferTy
pe];
  }
}
- (void)broadcastFinished {
  [[AlivcReplayKitExt sharedInstance] finishBroadcast];
}
@end
```

在您的工程中，完成创建Broadcast Upload Extension的Target，在该Extension Target中集成为录屏扩展模块定制的 `AlivcLibReplayKitExt.framework` 。

3. 在录屏推流主App进程中集成直播SDK。

   在录屏推流主App进程中创建AlivcLivePushConfig、AlivcLivePusher对象，设置外置推流ExternMainStream为True, AudioFromExternal为False（该配置表示音频仍通过SDK内部采集），调用StartScreenCapture开始接受Extension App屏幕数据，开始和结束推流。具体可参考以下操作步骤：

   i. 录屏推流主App进程中加入AlivcLivePusher.framework、AlivcLibRtmp.framework、RtsSDK.framework和AlivcLibReplayKitExt.framework的依赖。

   ii. 初始化推流SDK，配置使用外部视频源。

   ExternMainStream设置为True, ExternVideoFormat设置为AlivcLivePushVideoFormatYUV420P，音频使用内部SDK采集AudioFromExternal设置为False，配置其他推流参数，请参见示例代码如下：

```
  self.pushConfig.externMainStream = true;
self.pushConfig.externVideoFormat = AlivcLivePushVideoFormatYUV420P;
self.pushConfig.audioSampleRate = 44100;
self.pushConfig.audioChannel = 2;
self.pushConfig.audioFromExternal = false;
self.pushConfig.videoEncoderMode = AlivcLivePushVideoEncoderModeSoft;
self.pushConfig.qualityMode = AlivcLivePushQualityModeCustom;
self.pushConfig.targetVideoBitrate = 2500;
self.pushConfig.minVideoBitrate = 2000;
self.pushConfig.initialVideoBitrate = 2000;
self.livePusher = [[AlivcLivePusher alloc] initWithConfig:self.pushConfig];
```

iii. 使用AlivcLivePusher来完成直播相关功能，调用如下函数：

- 开始接受录屏数据。

  需将代码中的 `kAPPGroup` 替换成上文创建的 `App Group Identifier` ，示例代码如下：

  ```
  [self.livePusher startScreenCapture:kAPPGROUP];
  ```

- 开始推流。

  示例代码如下：

  ```
  [self.livePusher startPushWithURL:self.pushUrl]
  ```

- 结束推流。

  示例代码如下：

  ```
  [self.livePusher stopPush];
  [self.livePusher destory];
  self.livePusher = nil;
  ```

注意事项

- 关于包大小。
  - SDK大小为10.7MB。
  - 集成SDK后，IPA包增加大小约为2.8MB。
- 适配机型。

  iPhone5s及以上版本，iOS8.0及以上版本。
- 关于历史版本升级说明。

  请先删除旧版本的推流SDK，再升级至最新版本的推流SDK。版本升级详情请参见V4.0.2升级至V4.1.0及以上迁移说明。

5.5. 异常及特殊场景处理

本文介绍使用iOS推流SDK可能出现的异常情况、特殊情况，及其处理办法。

当您收到AlivcLivePusherErrorDelegate时

- 当出现onSystemError系统级错误时，您需要退出直播。
- 当出现onSDKError错误（SDK错误）时，有两种处理方式，选择其一即可：销毁当前直播并重新创建，或调用restartPush或restartPushAsync重启AlivcLivePusher。
- 您需要特别处理App没有麦克风权限和没有摄像头权限的回调，App没有麦克风权限错误码为268455940，App没有摄像头权限错误码为268455939。

当您收到AlivcLivePusherNetworkDelegate时

- 网速慢时，回调onNetworkPoor，当您收到此回调说明当前网络对于推流的支撑度不足，此时推流仍在继续、没有中断。网络恢复时，回调onNetworkRecovery，您可以在此处理自己的业务逻辑，比如UI提醒用户。
- 网络出现相关错误时，回调onConnectFail、onReconnectError或onSendDataTimeout。有两种处理方式，您只需选择其一：销毁当前推流并重新创建，或调用reconnectAsync进行重连，建议您重连之前先

进行网络检测和推流地址检测。

- SDK内部每次自动重连或者开发者主动调用reconnectAsync重连接口的情况下，会回调onReconnectStart重连开始。每次重连都会对RTMP进行重连链接。

  RTMP链接建立成功之后会回调onReconnectSuccess，此时只是链接建立成功，并不意味着可以推流数据成功。如果链接成功之后，由于网络等原因导致推流数据发送失败，SDK会继续重连。

- 推流地址鉴权即将过期会回调onPushURLAuthenticationOverdue。如果您的推流开启了推流鉴权功能（推流URL中带有auth_key），我们会对推流URL做出校验。在推流URL过期前约1min，您会收到此回调，实现该回调后，您需要回传一个新的推流URL，以此保证不会因为推流地址过期而导致推流中断。示例代码如下：

```
- (NSString *)onPushURLAuthenticationOverdue:(AlivcLivePusher *)pusher {
return @"新的推流地址 rtmp://";
}
```

当您收到AlivcLivePusherBGMDelegate背景音乐错误回调时

- 背景音乐开启失败时会回调onOpenFailed，检查背景音乐开始播放接口所传入的音乐路径与该音乐文件是否正确，可调用startBGMAsync重新播放。
- 背景音乐播放超时会回调onDownloadTimeout，多出现于播放网络URL的背景音乐，提示主播检查当前网络状态，可调用startBGMAsync重新播放。

当网络中断时

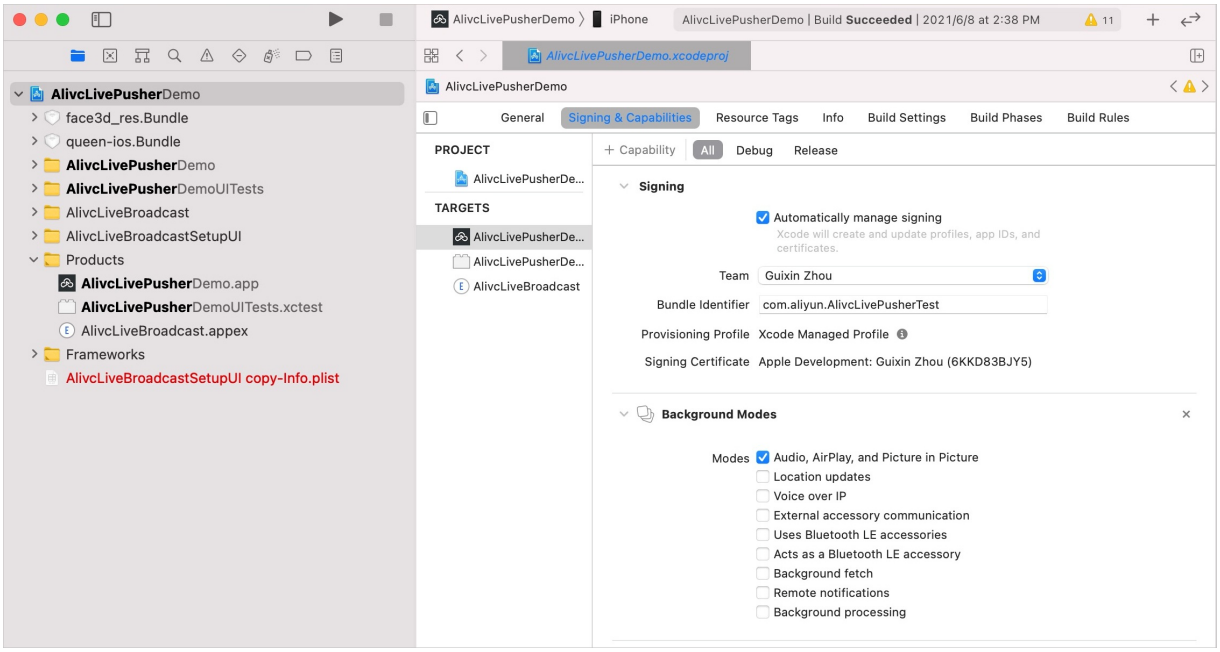
- 短时间断网和网络切换：即短时间的网络波动或者网络切换。一般情况下，中途断网时长在AlivcLivePushConfig设置的重连超时时长和次数范围之内，SDK会进行自动重连，重连成功之后将继续推流。若您使用阿里云播放器，建议播放器收到超时通知AliVcMediaPlayerPlaybackDidFinishNotification之后，短暂延时5s后再做重连操作。
- 长时间断网：断网时长超出AlivcLivePushConfig设置的重连超时时长和次数范围时，SDK自动重连失败，此时会回调 `onReconnectError:error:` ，等到网络恢复之后调用reconnectAsync接口进行重连。同时播放器也要配合做重连操作。
  - 建议您在SDK外部做网络监测。
  - 主播端和播放端在客户端无法进行直接通信，需要配合服务端使用。比如主播端断网，服务端会收到CDN的推流中断回调，此时可以推送给播放端，主播推流中断，播放端再做出相应处理。恢复推流同理。
  - 阿里云播放器重连需要先停止播放再开始播放。调用接口顺序stop>prepare>play。

```
[self.mediaPlayer stop];
AliVcMovieErrorCode err = [self.mediaPlayer prepareToPlay:[NSURL URLWithString:@"播放地址"]];
if(err != ALIVC_SUCCESS) {
 NSLog(@"play failed,error code is %d",(int)err);
 return;
}
[self.mediaPlayer play];
```

> **说明** 关于播放器请参见阿里云播放器SDK使用说明。

退后台和接听电话

SDK内部已经做好退后台相关处理，无需您做操作。退入后台SDK默认继续推流音频，视频保留在最后一帧。您需要在App的Capablities中打开Background Mode选项，选中**Audio，AirPlay and Picture in Picture**。保证App退后台可以正常采集音频。如图：



如果退后台时不需要保持音频推流，即退入后台停止推流，返回前台继续推流。您可以根据下面两种方式实现。

- 退后台设置为静音模式，调用setMute接口（建议方式）。
- 退后台停止推流，调用stopPush接口，回到前台继续推流，调用startPushWithURL或者startPushWithURLAsync接口。

> **说明** 在此方式下，退后台必须监听UIApplicationWillResignActiveNotification和UIApplicationDidBecomeActiveNotification，其他方式存在风险。

```
- (void)addNotifications {
[[NSNotificationCenter defaultCenter] addObserver:self
  selector:@selector(applicationWillResignActive:)
  name:UIApplicationWillResignActiveNotification
  object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
  selector:@selector(applicationDidBecomeActive:)
  name:UIApplicationDidBecomeActiveNotification
  object:nil];
}
 - (void)applicationWillResignActive:(NSNotification *)notification {
[self.livePusher stopPush];
}
 - (void)applicationDidBecomeActive:(NSNotification *)notification {
[self.livePusher startPushWithURLAsync:pushURL];
}
```

播放外部音效

如果您需要在推流页播放音效音乐等，由于SDK暂时与AudioServicesPlaySystemSound有冲突，建议您使用AVAudioPlayer，并且在播放后需要更新设置AVAudioSession、AVAudioPlayer播放音效示例代码：

```
- (void)setupAudioPlayer {
  NSString *filePath = [[NSBundle
mainBundle] pathForResource:@"sound" ofType:@"wav"];
  NSURL *fileUrl = [NSURL URLWithString:filePath];
 self.player = [[AVAudioPlayer alloc] initWithContentsOfURL:fileUrl error:nil];
 self.player.volume = 1.0;
 [self.player prepareToPlay];
}
 - (void)playAudio {
  self.player.volume = 1.0;
  [self.player play];
  // 配置AVAudioSession
  AVAudioSession *session = [AVAudioSession sharedInstance];
  [session setMode:AVAudioSessionModeVideoChat error:nil];
  [session overrideOutputAudioPort:AVAudioSessionPortOverrideSpeaker error:nil];
  [session setCategory:AVAudioSessionCategoryPlayAndRecord withOptions:AVAudioSessionCategoryOpti
onDefaultToSpeaker|AVAudioSessionCategoryOptionAllowBluetooth
| AVAudioSessionCategoryOptionMixWithOthers error:nil];
  [session setActive:YES error:nil];
}
```

推流过程中改变view的大小

请遍历您在调用startPreview或者startPreviewAsync接口时赋值的UIView。更改预览view的所有subView的frame。例如：

```
[self.livePusher startPreviewAsync:self.previewView];
for (UIView *subView in [self.previewView subviews]) {
 // ...
}
```

iPhoneX适配

一般场景下，预览view的frame设置为全屏可以正常预览，由于iPhoneX屏幕比例的特殊性，所以iPhoneX下预览view设置为全屏大小会有画面拉伸的现象。建议iPhoneX不要使用全屏大小的view来预览。

码率设置

SDK内部有动态变化码率策略，您可以在AlivcLivePushConfig中修改码率预设值。根据产品需求对于视频分辨率、视频流畅度、视频清晰度的要求不同，对应设置的码率范围也不同，具体参考如下：

- 视频清晰度：推流码率越高，则视频清晰度越高，所以推流分辨率越大，所需要设置的码率也就越大，以保证视频清晰度。
- 视频流畅度：推流码率越高，所需要的网络带宽越大，所以在较差的网络环境下，设置较高的码率有可能影响视频流畅度。

编译报错

当您收到 `Building for iOS, but the linked and embedded framework XXX.framework' was built for iOS + iOS Simulator` 编译报错时，请参见如下操作：

1. 单击Xcode菜单。
2. 选择File > Workspace Settings进入对话框设置。
3. 选择将build System更改为Legacy build system即可。

当编译缺少Queen的依赖库时

手动集成的情况下，出现Queen缺少依赖库的情况时，可以参见Queen_SDK_iOS文档添加对应的依赖库。

6.RTS推流（WebRTC推流）使用流程

本文介绍RTS推流时的使用流程，以及注意事项。

使用流程

1. 通过控制台或API完成添加域名、解析CNAME、关联域名等操作。请参见快速入门。
2. 通过控制台开启低延时推流功能。

   在域名管理页面，单击要配置的推流域名，在**直播管理 > 低延时推流**页面，开启低延时推流功能。

3. 生成推流地址。

   您可以通过**地址生成器**生成推流地址，请参见地址生成器。
4. 将推流地址的协议头改为 `artc` 。
5. 将推流地址填入推流SDK进行推流。

注意事项

RTS推流与RTMP推流相比有以下几处限制：

- 仅支持单声道。
- 目前不支持弱网自动推图片功能。
- 音频编码格式仅支持LC。
- 音频采样率仅支持48kHz。