阿里云
视频直播

推流SDK

文档版本：20190816

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
##	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。	`swich` *`{stand | slave}`*

目录

1 产品介绍

本文介绍推流SDK产品简介、功能特性、核心优势和使用场景。

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全面免费开放含人脸识别的高级美颜功能，可实现瘦脸、小脸、大眼、腮红以及基于人脸识别的美白功能。

功能特性

功能	描述
RTMP推流	支持低延时的rtmp协议直播推流，并支持rtmp、flv、hls直播拉流协议。分辨率支持180p-720p，建议使用540p。
录屏直播	iOS支持replayKit录屏直播，Android支持摄像头混流录屏直播。
直播答题	支持在直播流中插入SEI信息，通过播放器解析SEI实现直播答题功能。
动态水印	支持在直播中实时插入/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、红润、瘦脸、小脸、大眼、腮红等功能。
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）

功能	描述
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portrait、landscape left和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/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/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。
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1000+的直播节点为各区域的低延时提供了有效保障。

- 数据化运营支持

  提供多维度全景数据统计，高级别数据安全保密措施，丰富角度分析，客户画像描述助力业务拓。

使用场景

场景一：直播答题

- 场景描述：直播问答模式是用户在指定时间内登录直播间，在主持人引导下进行线上答题，答对12道题目即可冲顶奖金，瓜分每期设定的奖金。从2018年伊始，互联网圈就刮起了一阵“大佬狂撒币，网友喜答题”的热潮，您可以在自己的业务中增加这样强互动的方式，促成用户活跃的提升。详情参见 直播答题方案。
- 使用说明：开通阿里云直播服务并申请开通插入SEI服务，通过定制版OBS或者OpenAPI在直播流中插入SEI信息，当播放器在接收到SEI信息后会回调给App，这样用户就可以处理题目信息或答案信息，从而完成直播流里面同步展示答题信息。播放器部分参见 基础播放使用说明。

场景二：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和语言实时互动。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。

场景三：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、语言、表情、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。

场景四：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、语言、表情、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。

2 Android-推流SDK

2.1 概念说明

本文介绍Android推流SDK的概念说明。

码率控制

码率控制实际上是一种编码的优化算法，它用于控制视频流码流的大小。同样的视频编码格式，码流大，它包含的信息也就越多，那么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，我们可以通过丢弃视频帧，来缩短推流的延时。

耳返

耳返是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。因为声音通过骨骼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

混音是把多种来源的声音，整合至一个立体音轨或单音音轨中，SDK支持音乐和人声的混音。

混流

混流是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。

2.2 使用流程

本文介绍Android 推流SDK使用流程。

基本使用流程

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。

直播答题使用流程

直播答题通过推流SDK、定制版OBS或者阿里云OpenAPI在直播流中插入SEI信息（题目信息），当播放器解析到SEI信息后，会回调给用户的App，此时用户就可以将题目的具体题目内容展示出来。具体流程如下：

详细接入参见 直播答题方案，按照说明提交工单申请接入。播放器接入参见 答题播放器。OBS推流参见 OBS使用说明。

2.3 SDK集成

本文介绍Android推流SDK的快速上手集成。

集成环境

硬性要求

名称	要求
Android系统版本	>= Android 4.3
最小Android API版本	Jelly Bean (API 18)
CPU架构支持	ARM64、ARMV7
集成工具	Android Studio

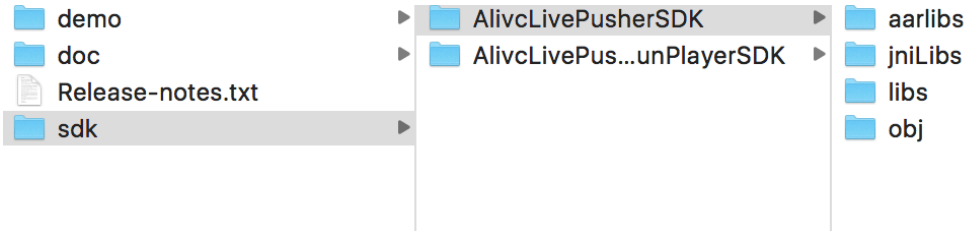
非硬性要求

仅仅是开发此Demo时的开发环境，目的是为了给编译运行源码的人员提供参考

名称	要求
Android Studio版本	3.2.1
JRE:	1.8.0_152-release-1136-b06 amd64
JVM:	OpenJDK 64-Bit
compileSdkVersion	26
buildToolsVersion	26.0.2
minSdkVersion	18
targetSdkVersion	26
gradle version	gradle-4.4-all
gradle plugin version	com.android.tools.build:gradle:3.0.1

推流SDK下载

阿里云官网Android版 推流SDK下载，推流SDK包含在解压包的AlivcLivePusher文件夹中，如下图所示：



上图中的文件内容作用如下：

文件目录	文件名称	文件说明
demo	AlivcLivePusherDemo	推流SDK演示Demo源代码工程
doc	api.zip	推流SDK API JavaDoc
sdk	AlivcLivePusherSDK	不带阿里云播放器的推流SDK
	AlivcLivePusherSDK+AliyunPlayerSDK	带阿里云播放器的推流SDK

上表中的sdk文件夹中的文件作用如下：

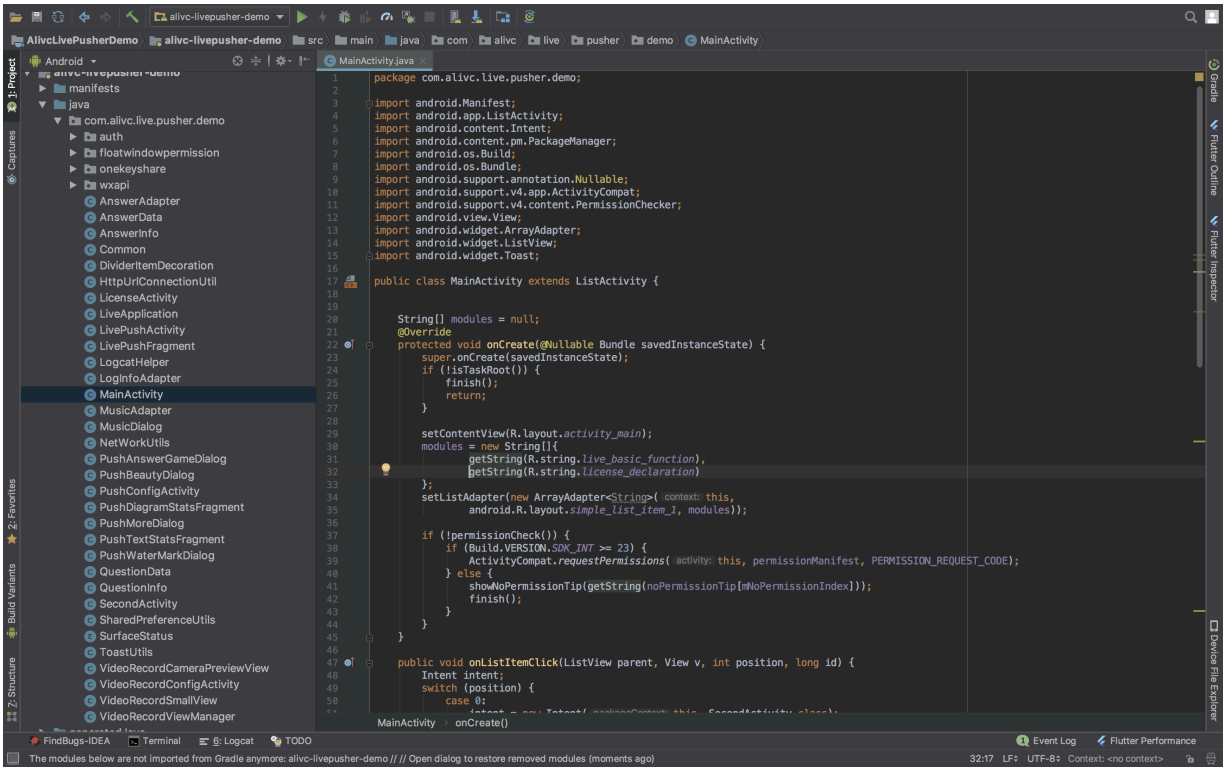
文件目录	文件说明
aarlibs	推流SDK所包含的各个组件aar包
jnilibs	推流SDK所包含的所有动态库包，分为arm64-v8a、armeabi-v7a。
libs	推流SDK所包含的各个组件jar包
obj	推流SDK所包含的动态库，用于定位底层问题。

说明:

- 可根据SDK包中的文件选择集成方式，采用aar集成或采用jar包+so集成。
- 使用背景音乐功能时必须集成播放器sdk(AlivcPlayer.aar)。

运行推流Demo

将AlivcLivePusherDemo工程导入到Android Studio中，如下图所示：



导入工程成功后，即可直接运行工程查看Demo效果。效果如下：


无服务
半夜11:31
直播场景列表
普通推流
录屏直播
允许Debug


无服务
半夜11:31
推流设置
rtmp://push-demo-rtmp.aliyunlive.c
防止多人使用同一地址推流造成异常，建议更换推流地址
分辨率
540P
显示模式
清晰度优
流畅度优
自定义
目标码率
1400
KB
最小码率
600
KB
初始码率
1400
KB
音频码率
64
KB
最小帧率
8
开始推流


答题
推流地址
rtmp://push-demo-rtmp.aliyunlive
.com/test/stream2350?&
auth_key=1550596840-0
-fa1b1cf8c2aeac46e32e9a42c96148c1
是否推流
true
停止预览
停止推流
暂停
重新推流
更多设置

推流url中填入有效的推流rtmp地址，推流成功后，拉流观看的效果可以使用阿里云播放器SDK、ffplay、VLC等工具查看。

推流SDK集成

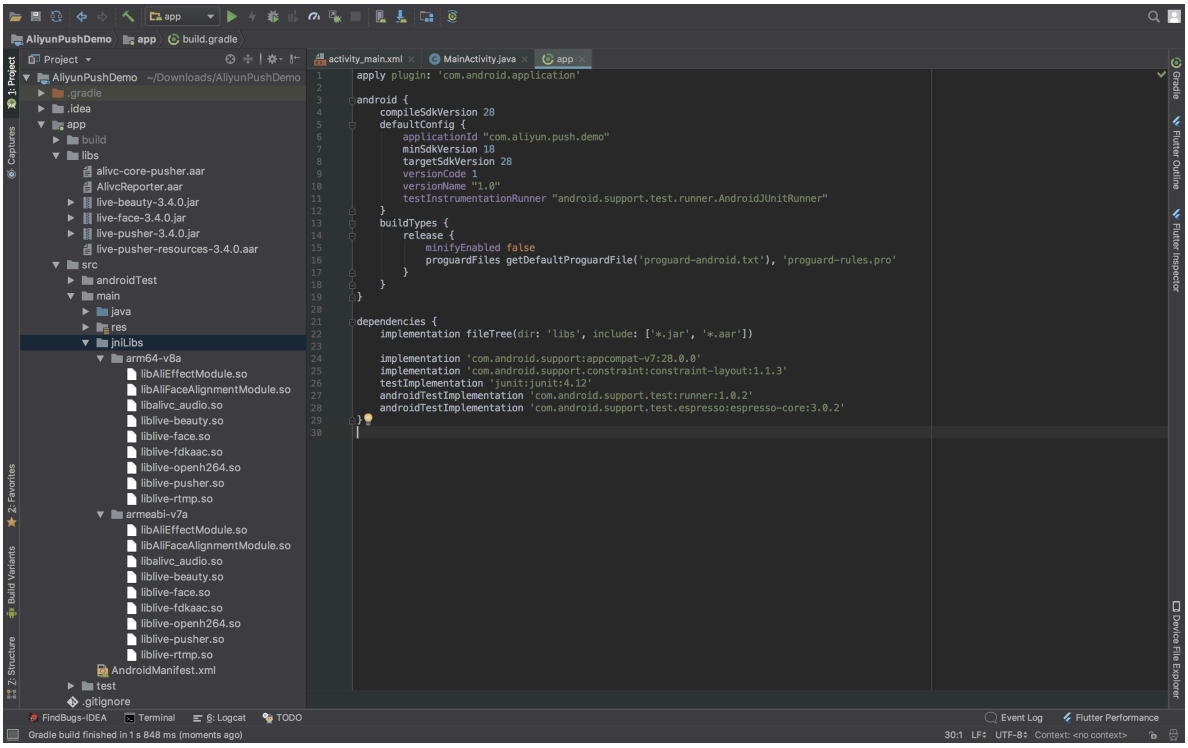
1. 使用aar的集成方式

   将所有SDK目录下文件拷贝到自己工程对应lib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'])
```

2. 使用jar+so的集成方式

   将SDK包下的jnilibs中的包拷贝到工程目录libs目录下，再将SDK包下的libs中的包拷贝到工程目录下jnilibs目录下，并在build.gradle中添加上相关的依赖，类似aar集成方式，具体如下图所示：



3. 添加请求权限

   在AndroidManifest文件下添加如下代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTE
RNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMO
UNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.MODIFY_AUD
IO_SETTINGS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/
>
<uses-permission android:name="android.permission.READ_EXTER
NAL_STORAGE"/>
```

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

说明:

请您记得添加录音权限和相机权限。

4. **添加混淆规则**

   在proguard-rules.pro文件下添加如下代码:

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

5. **具体使用说明**

   具体SDK的API使用请参考推流Android SDK的使用说明文档，API的详细注释说明请参考SDK包里的API JavaDoc文档。

2.4 SDK使用

本文详细说明如何使用Android推流SDK API以及相关功能。

推流SDK功能列表

- **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**
- **支持直播答题功能**
- **支持视频截图功能**
- **支持自动重连、异常处理**

推流主要接口类列表

类	说明
AlivcLivePushConfig	推流初始配置

类	说明
AlivcLivePusher	推流功能类
AlivcLivePushInfoListener	推流相关信息回调接口
AlivcLivePushNetworkListener	推流网络回调接口
AlivcLivePushErrorListener	推流报错回调接口
AlivcLivePushBGMListener	背景音乐回调接口
AlivcLivePushCustomFilter	自定义滤镜回调接口
AlivcLivePushCustomDetect	自定义人脸检测回调接口
AlivcSnapshotListener	截图回调接口

初始化

SDK 初始化主要包括以下几个步骤：

1. 创建AlivcLivePushConfig对象，并设置相关参数。

   AlivcLivePushConfig类用于推流的初始化设置，设置参数有：

参数	说明	默认
resolution	推流分辨率	540P
qualityMode	推流码率控制模式	分辨率优先模式
enableAutoBitrate	是否开启码率自适应模式	开启
enableAutoResolution	是否开启动态分辨率模式	关闭
fps	视频帧率	20fps
minFps	最小视频帧率	8fps
targetVideoBitrate	目标视频编码码率	800kbps
minVideoBitrate	最小视频编码码率	200kbps
initialVideoBitrate	初始视频编码码率	800kbps
audioBitrate	音频编码码率	64kbps
audioSampleRate	音频采样率	32000hz
audioChannel	音频声道数	双声道
videoEncodeGop	视频GOP大小	2秒
connectRetryCount	推流断开自动重连次数	5次
connectRetryInterval	推流断开自动重连间隔	1000毫秒
sendDataTimeout	发送网络数据超时时间	3000毫秒

参数	说明	默认
orientation	横竖屏设定	竖屏
cameraType	相机位置	前置
pushMirror	是否推流镜像	关闭
previewMirror	是否预览镜像	关闭
audioOnly	是否纯音频推流	关闭
videoOnly	是否纯视频推流	关闭
autoFocus	相机是否自动对焦	打开
beautyOn	是否打开美颜	打开
pauseImg	暂停退后台图片	空
networkPoorImg	弱网图片	空
beautyMode	美颜模式	普通美颜
beautyWhite	美白参数	70
beautyBuffing	磨皮参数	40
beautyRuddy	红润参数	40
beautyCheekPink	腮红参数	15
beautyThinFace	瘦脸参数	40
beautyShortenFace	收下巴参数	50
beautyBigEye	大眼参数	30
flash	是否打开手电筒	关闭
videoEncoderMode	视频编码器	硬件编码
audioEncoderProfile	音频编码格式	AAC LC
audioEncoderMode	音频编码器	软编码
externMainStream	自定义外部主流开关	关闭
externVideoFormat	自定义外部视频数据格式	
externAudioFormat	自定义外部音频数据格式	
previewDisplayMode	预览窗口显示模式	ASPECT FIT模式
addWatermarkWithPath	添加水印接口	
removeWatermarkWithPath	移除水印接口	
getAllWatermarks	获取所有水印信息	

参数	说明	默认
getPushResolution	获取推流分辨率	

a. 创建AlivcLivePushConfig对象

```
mAlivcLivePushConfig = new AlivcLivePushConfig();
```

b. 分辨率设置

```
mAlivcLivePushConfig.setResolution(AlivcResolutionEnum.RESOLUTION
_540P);
```

SDK支持的主播推流分辨率如下：

分辨率定义	分辨率
AlivcResolutionEnum.RESOLUTION_180P	192x320
AlivcResolutionEnum.RESOLUTION_240P	240x320
AlivcResolutionEnum.RESOLUTION_360P	368x640
AlivcResolutionEnum.RESOLUTION_480P	480x640
AlivcResolutionEnum.RESOLUTION_540P	544x960
AlivcResolutionEnum.RESOLUTION_720P	720x1280
AlivcResolutionEnum.RESOLUTION_PASS_THROUGH	屏幕分辨率

说明：

AlivcLivePushResolutionPassThrough只有在录屏直播模式下才有效，使用的是屏幕原始分辨率。

c. 码率控制设置

推流的码率控制有三种qualityMode模式可选：清晰度优先模式、流畅度优先模式和自定义模式。

- 清晰度优先模式（AlivcQualityModeEnum.QM_RESOLUTION_FIRST）：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。
- 流畅度优先模式（AlivcQualityModeEnum.QM_FLUENCY_FIRST）：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。
- 自定义模式（AlivcQualityModeEnum.QM_CUSTOM）：SDK会根据开发者设置的码率进行配置。

其中，清晰度优先模式和流畅度优先模式是根据日常使用场景由SDK定义的一套比较适合该场景的推荐使用模式，这两种模式下帧率和编码码率都是由定义好的，不可以外部调节，所以使用的时候需要特别注意。由于不同模式下您可以设置的参数有限制，以下是具体的组合说明：

qualityMode 码率控制模式	fps 帧率	minVideoBitrate 视频最小码率	initialVideoBitrate 视频初始码率	targetVideoBitrate 视频目标码率
清晰度优先模式（默认）	不可设置	不可设置	不可设置	不可设置
流畅度优先模式	不可设置	不可设置	不可设置	不可设置

qualityMode 码率控制模式	fps 帧率	minVideoBitrate 视频最小码率	initialVideoBitrate 视频初始码率	targetVideoBitrate 视频目标码率
自定义模式	可设置	可设置	可设置	可设置

从上表中可以看出只有自定义模式下才能设置视频的fps、目标码率、最小码率以及初始码率，而在清晰度模式和流畅度模式下这些值的设置并不生效，而是SDK内部自定义一套和不同分辨率对应的值，详细如下表。

清晰度优先模式各视频参数对应值：

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcResolutionEnum.RESOLUTION_180P	20	120	300	550
AlivcResolutionEnum.RESOLUTION_240P	20	180	450	750
AlivcResolutionEnum.RESOLUTION_360P	20	300	600	1000
AlivcResolutionEnum.RESOLUTION_480P	20	300	800	1200
AlivcResolutionEnum.RESOLUTION_540P	20	600	1000	1400

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcResolutionEnum.RESOLUTION_720P	20	600	1500	2000

流畅度优先模式各视频参数对应值：

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcResolutionEnum.RESOLUTION_180P	25	80	200	250
AlivcResolutionEnum.RESOLUTION_240P	25	120	300	350
AlivcResolutionEnum.RESOLUTION_360P	25	200	400	600
AlivcResolutionEnum.RESOLUTION_480P	25	300	600	800
AlivcResolutionEnum.RESOLUTION_540P	25	300	800	1000
AlivcResolutionEnum.RESOLUTION_720P	25	300	1000	1200

说明：

如果觉得清晰度优先模式和流畅度优先模式下的视频参数设置不满足推流场景要求，必须使用自定义模式来设置自定义的参数值。

码率自适应

码率自适应是推流码率控制中的一种策略，其定义表示推流过程中是否根据当前实际网络代码动态调整视频的编码码率，调整范围就是由最小码率、初始码率和，目标码率来限定的。

码率自适应打开时，推流过程中首先使用初始码率作为编码码率，当检测到当前上传带宽充足时，会慢慢上调编码码率，直至到目标码率就不再向上调整，而当检测到当前上传带宽不足时，开始下调视频编码码率，下调的最小值为最小码率，此时不再继续下调。

当码率自适应关闭时，整个推流过程中始终使用初始编码码率，不会动态去调整编码码率。

动态分辨率

动态分辨率是在推流过程中，根据码率调整到一定的范围后，可以动态切换相应的分辨率，也就是说此设置是依赖码率自适应的开关的，如果码率自适应关闭，此设置无效，且动态分辨率目前只支持清晰度优先模式和流畅度优先模式，不支持自定义模式。当动态分辨率开关生效后，对应的分辨率切换范围为：

分辨率	最小码率kbps	最大码率kbps率
AlivcResolutionEnum.RESOLUTION_480P	无	1000
AlivcResolutionEnum.RESOLUTION_540P	1000	1200
AlivcResolutionEnum.RESOLUTION_720P	1200	无

目前只支持分辨率在480P、540P和720P中切换，当码率调整在相应的区间中时，会动态调整成相应的分辨率。分辨率调整只能由初始分辨率向下调整，也就是说如果初始设置分辨率为540P，那么动态分辨率只能最大调整到540P。

 说明：

动态分辨率在某些播放器上可能无法正常播放，阿里云播放器已支持。使用动态分辨率功能时，建议您使用阿里云播放器。

d. 纯音视频推流设置

SDK支持纯音频和纯视频推流设置，通过打开audioOnly，SDK就会只推纯音频流，如果打开videoOnly，SDK只推纯视频流。不可以同时打开这两种模式。

示例代码如下：

```
//设置纯音频推流
mAlivcLivePushConfig.setAudioOnly(true);
```

```
//设置纯视频推流
mAlivcLivePushConfig.setVideoOnly(true);
```

e. 推流图片设置

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在退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：

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

另外，当网络较差时用户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：

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

说明：

此处的图片只支持png格式，且在app资源文件中能够访问到的图片，不支持网络图片。

f. 水印设置

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，目前最多支持3个水印。示例代码如下：

```
//添加水印
mAlivcLivePushConfig.addWaterMark(mWaterMarkPath, 0.1f, 0.1f, 0.3f
);
```

说明：

- coordX、coordY、width为相对值，例如watermarkCoordX:0.1表示水印的x值为推流画面的10%位置，即推流分辨率为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。

g. 显示模式设置

推流预览显示支持以下三种模式：

- AlivcPreviewDisplayMode.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。
- AlivcPreviewDisplayMode.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）。
- AlivcPreviewDisplayMode.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。

说明:

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。

2. 创建AlivcLivePusher对象

当AlivcLivePushConfig对象创建成功并设置了初始化参数后，根据AlivcLivePushConfig对象来创建AlivcLivePusher对象。示例代码如下：

```
mAlivcLivePusher = new AlivcLivePusher();
try {
    mAlivcLivePusher.init(getApplicationContext(), mAlivcLive
PushConfig);
} catch (IllegalArgumentException e) {
    e.printStackTrace();
} catch (IllegalStateException e) {
    e.printStackTrace();
}
```

3. 注册回调

推流主要回调分为三种：Info、Error、Network、背景音乐、截图、自定义滤镜和自定义人脸检测回调，注册相应接口可接收到对应的回调。示例代码如下：

```
mAlivcLivePusher.setLivePushInfoListener(this);
mAlivcLivePusher.setLivePushNetworkListener(this);
mAlivcLivePusher.setLivePushBGMListener(this);
mAlivcLivePusher.setLivePushErrorListener(this);
```

```
mAlivcLivePusher.setLivePushBGMListener(this);
mAlivcLivePusher.setCustomFilter(this);
mAlivcLivePusher.setCustomDetect(this);
```

预览

1. 开始预览

   初始成功之后，就是进入预览的环节，预览使用外部传入一个SurfaceView，调用startPreview接口后SDK内部会自动打开相机采集，并将采集画面渲染到预览 SurfaceView 上。示例代码如下：

   ```
   if(mAsync) {
     mAlivcLivePusher.startPreviewAysnc(mPreviewView);
   } else {
     mAlivcLivePusher.startPreview(mPreviewView);
   }
   ```

   说明：

   如果APP没有打开过系统相机使用权限，这一步会要求打开系统相机权限，如果选择禁止打开相机，会导致预览失败。

2. 停止预览

   调用stopPreview可以停止预览，此时画面会停留在最后一帧。示例代码如下：

   ```
   mAlivcLivePusher.stopPreview();
   ```

推流

1. 开始推流

   预览成功后，调用startPush接口开始推流，startPush接口需要传入有效的rtmp推流url地址，阿里云推流地址获取参见 快速开始。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，拉流地址如何获取参见 快速开始。示例代码如下：

```
mAlivcLivePusher.startPush("推流测试地址(rtmp://......)");
```

2. 停止推流

   通过调用stopPush接口可以停止推流。示例代码如下：

```
mAlivcLivePusher.stopPush();
```

3. 重新推流

   通过调用restartPush可以在推流中或者推流出错后，重新开始推流。示例代码如下：

```
mAlivcLivePusher.restartPush();
```

暂停恢复

在推流过程中，通过调用pause和resume接口来控制推流的暂停和恢复。

1. 暂停

   暂停是指推流过程中调用pause接口暂停相机的采集，此时的推流行为是只采集音频数据，视频数据不采集，所以此时推流只推音频数据，不推视频数据，但是如果用户设置了pauseImg，此时视频会推设置的图片数据。示例代码如下：

```
mAlivcLivePusher.pause();
```

2. 恢复

   恢复就是通过调用resume接口恢复正常推流状态。示例代码如下：

```
mAlivcLivePusher.resume();
```

重连

重连操作包括自动重连和手动重连两种。

1. 自动重连

   在SDK内部检测到socket连接失败或者断开的时候，会有自动重连机制，在AlivcLivePushConfig中的connectRetryCount重连次数和connectRetryInterval重连间隔用来控制自动重连的次数和每次之间的间隔，当所有重连次数用完还没有重连成功时，会上报重连失败错误后不再继续重连。

2. 手动重连

   手动重连是外部调用SDK接口reconnectPushAsync接口进行手动重连。

美颜控制

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、腮红等功能。

目前SDK 美颜采用外置对接方式，也就是说APP 需要对接实现美颜的回调才能使用美颜功能。首先app需要加载美颜的AAR文件live-beauty-xxxx.aar、人脸检测AAR文件live-face-xxxx.aar。然后实现美颜和人脸检测的回调。示例代码如下：

```
//人脸识别回调（只接入标准版美颜可不需要调用此接口）
mAlivcLivePusher.setCustomDetect(new AlivcLivePushCustomDetect()
{
   @Override
  public void customDetectCreate() {
      taoFaceFilter = new TaoFaceFilter(getApplicationContext());
      taoFaceFilter.customDetectCreate();
  }
  @Override
  public long customDetectProcess(long data, int width, int height,
int rotation, int format, long extra) {
      if(taoFaceFilter != null) {
          return taoFaceFilter.customDetectProcess(data, width, height
, rotation, format, extra);
      }
      return 0;
  }
  @Override
  public void customDetectDestroy() {
      if(taoFaceFilter != null) {
          taoFaceFilter.customDetectDestroy();
      }
  }
});

//美颜回调
mAlivcLivePusher.setCustomFilter(new AlivcLivePushCustomFilter()
{
   @Override
  public void customFilterCreate() {
      taoBeautyFilter = new TaoBeautyFilter();
      taoBeautyFilter.customFilterCreate();
  }
  @Override
  public void customFilterUpdateParam(float fSkinSmooth, float fWhiten
, float fWholeFacePink, float fThinFaceHorizontal, float fCheekPink,
float fShortenFaceVertical, float fBigEye) {
      if (taoBeautyFilter != null) {
          taoBeautyFilter.customFilterUpdateParam(fSkinSmooth,
fWhiten, fWholeFacePink, fThinFaceHorizontal, fCheekPink, fShortenFa
ceVertical, fBigEye);
      }
  }
  @Override
  public void customFilterSwitch(boolean on)
  {
      if(taoBeautyFilter != null) {
          taoBeautyFilter.customFilterSwitch(on);
```

```
        }
    }
    @Override
    public int customFilterProcess(int inputTexture, int textureWidth,
int textureHeight, long extra) {
        if (taoBeautyFilter != null) {
            return taoBeautyFilter.customFilterProcess(inputTexture,
textureWidth, textureHeight, extra);
        }
        return inputTexture;
    }
    @Override
    public void customFilterDestroy() {
        if (taoBeautyFilter != null) {
            taoBeautyFilter.customFilterDestroy();
        }
        taoBeautyFilter = null;
    }
});
```

通过设置AlivcLivePushConfig中的相关美颜参数，或者调用美颜相关接口都可以控制美颜开关、是否使用高级美颜以及调节美颜参数。

基本设置相关代码。示例代码如下：

```
mAlivcLivePushConfig.setBeautyOn(true);// 开启美颜
mAlivcLivePushConfig.setBeautyLevel(AlivcBeautyLevelEnum.BEAUTY_Pro
fessional);//设定为高级美颜
mAlivcLivePushConfig.setBeautyWhite(70);// 美白范围0-100
mAlivcLivePushConfig.setBeautyBuffing(40);// 磨皮范围0-100
mAlivcLivePushConfig.setBeautyRuddy(40);// 红润设置范围0-100
mAlivcLivePushconfig.setBeautyBigEye(30);// 大眼设置范围0-100
mAlivcLivePushConfig.setBeautyThinFace(40);// 瘦脸设置范围0-100
mAlivcLivePushConfig.setBeautyShortenFace(50);// 收下巴设置范围0-100
mAlivcLivePushConfig.setBeautyCheekPink(15);// 腮红设置范围0-100
```

动态更改可以通过接口setBeautyOn、setBeautyWhite、setBeautyBuffing、setBeautyRuddy、setBeautyCheekPink、setBeautyThinFace、setBeautyShortenFace和setBeautyBigEye来调节美颜相关设置。

说明:

由于美颜采用了外置对接方式，这样很方便app对接第三方的美颜和人脸检测的滤镜，只要在对应的回调处理中调用第三方滤镜的相关接口就能使SDK中完美对接第三方的美颜滤镜。

我们提供了丰富的美颜参数，建议您在UED同事配合下，调整出最符合自己应用风格的美颜参数。

以下几组美颜效果比较理想，您可参考下表：

档位	磨皮	美白	红润	大眼	瘦脸	收下巴	腮红
一档	40	35	10	0	0	0	0
二挡	60	80	20	0	0	0	0

档位	磨皮	美白	红润	大眼	瘦脸	收下巴	腮红
三挡	50	100	20	0	0	0	0
四挡	40	70	40	30	40	50	15
五档	70	100	10	30	40	50	0

相机控制

相机控制是包括切换相机、获取当前相机位置、对焦、缩放、曝光和手电筒操作。

1. 切换相机

   切换前置后置相机，预览和直播中都可以切换。

```
mAlivcLivePusher.switchCamera();
```

2. 设置手电筒

```
mAlivcLivePusher.setFlash(true);
```

3. 自动对焦

```
mAlivcLivePusher.setAutoFocus(true);
```

4. 手动对焦

```
float x = motionEvent.getX() / mPreviewView.getWidth();
float y = motionEvent.getY() / mPreviewView.getHeight();
try{
    mAlivcLivePusher.focusCameraAtAdjustedPoint(x, y, true);
} catch (IllegalStateException e) {
}
```

5. 设置缩放倍数

   缩放接口为增量设置，每次设置参数都是在当前基础上做增减，缩放倍数范围（1-3）。

```
//当前的缩放倍数+1
```

```
mAlivcLivePusher.setZoom(1.0f);
```

6. 获取缩放倍数

```
mAlivcLivePusher.getCurrentZoom();
```

7. 获取最大缩放倍数

```
mAlivcLivePusher.getMaxZoom();
```

8. 设置曝光度

```
mAlivcLivePusher.setExposure(mExposure);
```

9. 获取相机最小曝光度值

```
mAlivcLivePusher.getSupportedMinExposure();
```

10.获取相机最大曝光度值

```
mAlivcLivePusher.getSupportedMaxExposure();
```

11.获取当前曝光度值

```
mAlivcLivePusher.getCurrentExposure();
```

截图

截图功能是在推流和预览中可以截取视频生成图片，通过调用接口snapshot来实现，snapshot中有两个参数count和interval， count 表示要截取图片的个数，interval表示每张图片的间隔，如果需要立刻截取当前一张视频图片，只要传递count为1，interval为0即可。截图是异步代理回调操作，截图结果通过传入的AlivcSnapshotListener接口回调出来。示例代码如下：

```
mAlivcLivePusher.snapshot(1, 0, new AlivcSnapshotListener() {
    @Override
    public void onSnapshot(Bitmap bmp) {
      //保存视频流截图bitmap
    }
});
```

背景音乐

该功能提供背景音乐播放、混音、降噪、耳返、静音等功能，通过AlivcLivePusher中的如下接口实现以上功能：

```
//开始播放背景音乐
mAlivcLivePusher.startBGMAsync(mPath);

//停止播放背景音乐
mAlivcLivePusher.stopBGMAsync();

//暂停播放背景音乐
mAlivcLivePusher.pauseBGM();
```

```
//恢复播放背景音乐
mAlivcLivePusher.resumeBGM();

//设置是否循环播放背景音乐
mAlivcLivePusher.setBGMLoop();

//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(isOpen);

//设置降噪开关，默认为false不使用
//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
mAlivcLivePusher.setAudioDenoise(onDenoise);

//设置背景音乐音量
mAlivcLivePusher.setBGMVolume(50);

//设置人声采集音量
mAlivcLivePusher.setCaptureVolume(50);

//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(isMute);
```

说明:

该功能只能在开始预览之后才可调用，目前只支持mp3格式音乐文件。

自定义音视频流输入

该功能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、第三方设备采集的音视频数据等。具体功能使用介绍如下：

1. 全局配置自定义音视频输入

   在推流配置AlivcLivePushConfig中配置使用自定义音视频输入，也可包括与自定义输入流相关的其他配置，具体配置如下：

```
//是否使用外部音视频输入，并设置相关音视频输入格式
//AlivcImageFormat输入的视频图像格式
//AlivcSoundFormat输入的音频帧格式
mAlivcLivePushConfig.setExternMainStream(true, AlivcImageFormat.
IMAGE_FORMAT_YUVNV12, AlivcSoundFormat.SOUND_FORMAT_S16);

//设置输入视频分辨率，默认值 540P
mAlivcLivePushConfig.setResolution(mAlivcResolutionEnum);

//设置音频采样率，默认值 32000HZ
mAlivcLivePushConfig.setAudioSamepleRate(mAudioSampleRate);

//设置音频采集声道，默认2个
```

```
mAlivcLivePushConfig.setAudioChannels(mAudioChannels);
```

其中，视频图像格式目前支持：

- IMAGE_FORMAT_BGRA
- IMAGE_FORMAT_RGBA
- IMAGE_FORMAT_YUVNV12
- IMAGE_FORMAT_YUVNV21
- IMAGE_FORMAT_YUV420P

音频帧格式目前只支持PCM 格式，且为SOUND_FORMAT_S16。

2. 插入自定义音视频数据

- 插入视频数据

```
/**
 * 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 pts        视频图像pts (μs)
 * @param rotation   视频图像旋转角度
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
mAlivcLivePusher.inputStreamVideoData(mBuffer, 720, 1280, 1280*720
*3/2, System.nanoTime()/1000, 0);

/**
 * 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 pts        视频图像pts (μs)
 * @param rotation   视频图像旋转角度
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
mAlivcLivePusher.inputStreamVideoPtr(mDataptr, 720, 1280, 1280*720
*3/2, System.nanoTime()/1000, 0);

/**
 * 输入自定义视频流
 *
 * @param textureId 视频纹理id
 * @param width      视频图像宽度
 * @param height     视频图像高度
 * @param pts        视频图像pts (us)
 * @param rotation   视频图像旋转角度
 * @param extra      暂无作用
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
```

```
mAlivcLivePusher.inputStreamTexture(mTextureId, 720, 1280, 1280*
720*3/2, System.nanoTime()/1000, 0, 0);
```

- 插入音频数据

```
/**
* 输入自定义音频数据
*
* @param data 音频数据 byte array
* @param size 音频数据size
* @param pts  音频数据pts(μs)
* 注意：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(mBuffer, 2048, System.
nanoTime()/1000);

/**
* 输入自定义音频数据
*
* @param dataPtr  音频数据native内存指针
* @param size     音频数据size
* @param pts      音频数据pts(μs)
* 注意：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(mDatePrt, 2048, System.
nanoTime()/1000);
```

混流功能

混流顾名思义就是把一路或多路音视频流混合成单路流推到流媒体服务器。具体使用介绍如下：

1. 添加混流

添加混流包括混流格式设置、旋转角度设置、混流位置、音频采样率设置等，添加后获取到的返回值用于标示这一路混流，假如返回-1则混流失败，具体代码调用示例如下：

```
/**
 * 添加一路视频混流，并设置狂高，设置混在主流的相对位置
 *
 * @param format     当前输入视频图像格式
 * @param width      当前输入视频图像宽度
 * @param height     当前输入视频图像高度
 * @param rotation   当前输入视频图像旋转角度
 * @param displayX   当前混流起始位置X
 * @param displayY   当前混流起始位置Y
 * @param displayW   当前混流显示区域Width
 * @param displayH   当前混流显示区域Height
 * @return 返回当前视频混流videoHandler，用于标示这一路混流，该值用于改变、移
除混流等
 */
mAlivcLivePusher.addMixVideo(AlivcImageFormat.IMAGE_FORMAT_RGBA,
mWidth, mHeight, 0, mRect.mStartX, mRect.mStartY, mRect.mWidth,
mRect.mHeight);

/**
 * 添加一路音频混流，并设置相应音频格式和采样率
 *
 * @param channels     当前音频流声道数
 * @param format       当前音频流格式
 * @param audioSample  当前音频流采样率
```

```
 * @return 返回当前音频混流audioHandler，用于标示这一路音频流，该值用于移除混
流
 */
mAlivcLivePusher.addMixAudio(mNumChannels, AlivcSoundFormat.
SOUND_FORMAT_S16, mNumSamples * 100);
```

2. 更新混流位置

```
/**
 * 更新视频混流位置
 *
 * @param videoHandler 已有混流的videoHandler值
 * @param displayX   当前混流起始位置X
 * @param displayY   当前混流起始位置Y
 * @param displayW   当前混流显示区域Width
 * @param displayH   当前混流显示区域Height
 * @return status 返回当前混流状态
 */
mAlivcLivePusher.mixStreamChangePosition(mMixVideoHandler, mRect.
mStartX, mRect.mStartY, mRect.mWidth, mRect.mHeight);
```

3. 输入混流数据

确认添加好混流后，将已获得的混流返回值用于实际音视频流数据输入，具体示例如下：

```
/**
 * 输入视频混流数据
 *
 * @param videoHandler  已有混流的videoHandler值
 * @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size
 * @param pts           视频图像pts (μs)
 * @param rotation      视频图像旋转角度
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
mAlivcLivePusher.inputMixVideoPtr(mMixVideoHandler, mDataPtr, 720,
1280, 1280*720*3/2, System.nanoTime()/1000, 0);

/**
 * 输入视频混流数据
 *
 * @param videoHandler  已有混流的videoHandler值
 * @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 pts           视频图像pts (μs)
 * @param rotation      视频图像旋转角度
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
mAlivcLivePusher.inputMixVideoData(mMixVideoHandler, mBuffer, 720,
1280, 1280*720*3/2, System.nanoTime()/1000, 0);

/**
 * 输入视频混流数据
 *
 * @param videoHandler  已有混流的videoHandler值
 * @param textureId     视频图像纹理id
 * @param width         视频图像宽度
 * @param height        视频图像高度
```

```
 * @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
 * 注意：此接口不控制时序，需要调用方控制输入视频帧的时序
 */
mAlivcLivePusher.inputMixTexture(mMixVideoHandler, mTextureId, 720,
1280, 1280*720*3/2, System.nanoTime()/1000, 0);

/**
 * 输入音频混流数据
 *
 * @param audioHandler  已有混流的audioHandler值
 * @param dataPtr       音频数据native内存指针
 * @param size          音频数据size
 * @param pts           音频数据pts(μs)
 * 注意：此接口不控制时序，需要调用方控制输入音频帧的时序
 */
mAlivcPusher.inputMixAudioPtr(mMixAudioHandler, mDataPtr, 2048,
System.nanoTime()/1000);

/**
 * 输入音频混流数据
 *
 * @param audioHandler  已有混流的audioHandler值
 * @param data          音频数据 byte array
 * @param size          音频数据size
 * @param pts           音频数据pts(μs)
 * 注意：此接口不控制时序，需要调用方控制输入音频帧的时序
 */
mAlivcPusher.inputMixAudioData(mMixAudioHandler, mBuffer, 2048,
System.nanoTime()/1000);
```

4. 删除混流

清理已有混流，包括移除音频和视频，具体示例如下：

```
/**
 * 移除已添加的一路视频混流
 *
 * @param handler 已有混流的videoHandler值
 */
mAlivcLivePusher.removeMixVideo(mMixVideoHandler);

/**
 * 移除已添加的一路音频混流
 *
 * @param handler 已有混流的audioHandler值
 */
mAlivcLivePusher.removeMixAudio(mMixAudioHandler);
```

直播答题

该功能可以通过在直播流里面插入SEI信息并发送后阿里云播放器SDK可收到此SEI消息，解析后做具体展示。另外为了保证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，这个过程中播放器会对相同的SEI进行去重处理。示例代码如下：

```
/*
 * 推流端发送自定义消息
 *
 * @param info          需要插入流的SEI消息体，建议是json格式
```

```
 * @param repeat       发送的帧数
 * @param delay        延时多少毫秒发送
 * @param isKeyFrame   是否只发送关键帧
 */
mAlivcLivePusher.sendMessage(mQuestionInfo, 100, 0, false);
```

说明:

只有在推流状态下，才能调用此接口。

录屏直播

录制是把当前手机屏幕画面作为推流数据源，同时可叠加摄像头预览以及添加自定义混流数据。具体使用如下:

1. 开启录屏

   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。

   ```
   //在config中设置权限请求的返回数据
   if (resultCode == Activity.RESULT_OK) {
       mAlivcLivePushConfig.setMediaProjectionPermissionResultData(data
   );
   }
   ```

2. 设置录屏模式

   录屏主要有三种模式:

   - 录屏时不开启摄像头，即只需开启录屏模式在config中设置权限请求的返回数据即可。
   - 录屏时开启摄像头，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）。

     ```
     //在config中设置权限请求的返回数据
     if (resultCode == Activity.RESULT_OK) {
         mAlivcLivePushConfig.setMediaProjectionPermissionResultData(
     data);
     }

     //调用StartCamera传入surfaceView
     if(mLivePusher != null ) {
         mAlivcLivePusher.startCamera(mCameraPreview);
     }
     ```

   - 录屏时开启摄像头，主播端无摄像头预览，观众端有摄像头画面叠加。

     ```
     //在config中设置权限请求的返回数据
     if (resultCode == Activity.RESULT_OK) {
         mAlivcLivePushConfig.setMediaProjectionPermissionResultData(
     data);
     }

     //调用StartCamera传入surfaceView
     if(mLivePusher != null ) {
     ```

```
    mAlivcLivePusher.startCamera(mCameraPreview);
}

//调用startCameraMix传入观众端摄像头画面显示位置
if(mAlivcLivePusher != null && mMixOn) {
    mAlivcLivePusher.startCameraMix(0.6f,0.08f, 0.3f,0.3f);
}
```

3. 录屏其他设置

录屏相关其他设置示例如下：

```
//开启和关闭设想头预览
//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，stopCamera再startCamera;
// 如果主播端不需要预览，则surfaceview填为null;

//开启摄像头预览
mAlivcLivePusher.startCamera(mSurfaceView);

//关闭摄像头预览
mAlivcLivePusher.stopCamera();


//摄像头混流
//开启摄像头混流
mAlivcLivePusher.startCameraMix(x, y, w, h);

//停止摄像头混流
mAlivcLivePusher.stopCameraMix();


//设置屏幕旋转
//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

//设置感应的屏幕旋转角度，支持横屏/竖屏录制
mAlivcLivePusher.setScreenOrientation(0);


//开关隐私保护
//说明：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

//开启隐私保护
mAlivcLivePusher.pauseScreenCapture();

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

 说明:

录屏只支持5.0及以上的系统。

接口回调处理

1. 错误回调处理

   当收到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：销毁当前直播重新创建、调用restartPush/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_MIC_OPEN_FAILED
     - APP没有摄像头权限错误码为：ALIVC_PUSHER_ERROR_SDK_CAPTURE_CAMERA_OPEN_FAILED

   AlivcLivePushErrorListener接口回调示例代码如下：

```
//系统错误回调
@Override
void onSystemError(AlivcLivePusher livePusher, AlivcLivePushError
error);

//SDK错误回调
@Override
void onSDKError(AlivcLivePusher livePusher, AlivcLivePushError error
);
```

2. 网络监听回调处理

   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，此时推流仍在继续并没有中断。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。
   - 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。

AlivcLivePushNetworkListener接口回调示例代码如下：

```
//当前网络质量差
@Override
void onNetworkPoor(AlivcLivePusher pusher);

//网络恢复
@Override
void onNetworkRecovery(AlivcLivePusher pusher);

//开始重连
@Override
void onReconnectStart(AlivcLivePusher pusher);

//连接失败
@Override
void onConnectionLost(AlivcLivePusher pusher);

//重连失败
@Override
void onReconnectFail(AlivcLivePusher pusher);

//重连成功
@Override
void onReconnectSucceed(AlivcLivePusher pusher);

//发送数据超时
@Override
void onSendDataTimeout(AlivcLivePusher pusher);

//连接失败
@Override
void onConnectFail(AlivcLivePusher pusher);

//推流Url鉴权过期
@Override
String onPushURLAuthenticationOverdue(AlivcLivePusher pusher) {
    return "新的推流地址 rtmp://";
}

//发送SEI消息
@Override
void onSendMessage(AlivcLivePusher pusher);

//推流过程丢包回调
@Override
```

```
void onPacketsLost(AlivcLivePusher pusher);
```

3. 背景音乐播放回调处理

当您收到AlivcLivePushBGMListener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。

AlivcLivePushBGMListener接口回调示例代码如下：

```
//播放开始事件
@Override
void onStarted();

//播放停止事件
@Override
void onStoped();

//播放暂停事件
@Override
void onPaused();

//播放恢复事件
@Override
void onResumed();

//播放进度事件
@Override
void onProgress(long progress, long duration);

//播放结束通知
@Override
void onCompleted();

//播放器超时事件
@Override
void onDownloadTimeout();

//流无效通知
@Override
void onOpenFailed();
```

特殊情况处理

1. 网络中断处理

- 短时间断网和网络切换：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：断网时长在AlivcLivePushConfig设置的重连超时时长和次数范围之外的情况下，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。
- 针对网络情况提供以下建议说明：
  - 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，此时可以推动给播放端，主播推流中断，播放端在作出相应处理。恢复推流同理。
  - 阿里云播放器重连需要先停止播放在开始播放。调用接口顺序stop > prepareAndPlay 。

2. 退后台和锁屏

- 当app退后台或锁屏时，您可调用 AlivcLivePusher 的 pause()/resume()接口暂停/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/false)来决定后台时是否采集音频。

结束销毁推流

当使用推流结束时需要调用推流相关的销毁接口，清理当前使用资源。

- 正常推流模式下的具体示例

```
if(mAlivcLivePusher != null) {
  //停止推流
  try {
    mAlivcLivePusher.stopPush();
  } catch (Exception e) {
  }

  //停止预览
  try {
    mAlivcLivePusher.stopPreview();
  } catch (Exception e) {
  }

  //释放推流
  mAlivcLivePusher.destroy();
  mAlivcLivePusher.setLivePushInfoListener(null);
  mAlivcLivePusher = null;
}
```

- 录屏推流模式下的具体示例

```
if (mAlivcLivePusher != null) {
```

```
    //关闭摄像头预览
    try {
      mAlivcLivePusher.stopCamera();
    } catch (Exception e) {
    }

    //停止摄像头混流
    try {
      mAlivcLivePusher.stopCameraMix();
    } catch (Exception e) {
    }

    //停止推流
    try {
      mAlivcLivePusher.stopPush();
    } catch (Exception e) {
    }

    //停止预览
    try {
      mAlivcLivePusher.stopPreview();
    } catch (Exception e) {
    }

    //释放推流
    mAlivcLivePusher.destroy();
    mAlivcLivePusher.setLivePushInfoListener(null);
    mAlivcLivePusher = null;
}
```

注意事项

- **混淆规则说明**

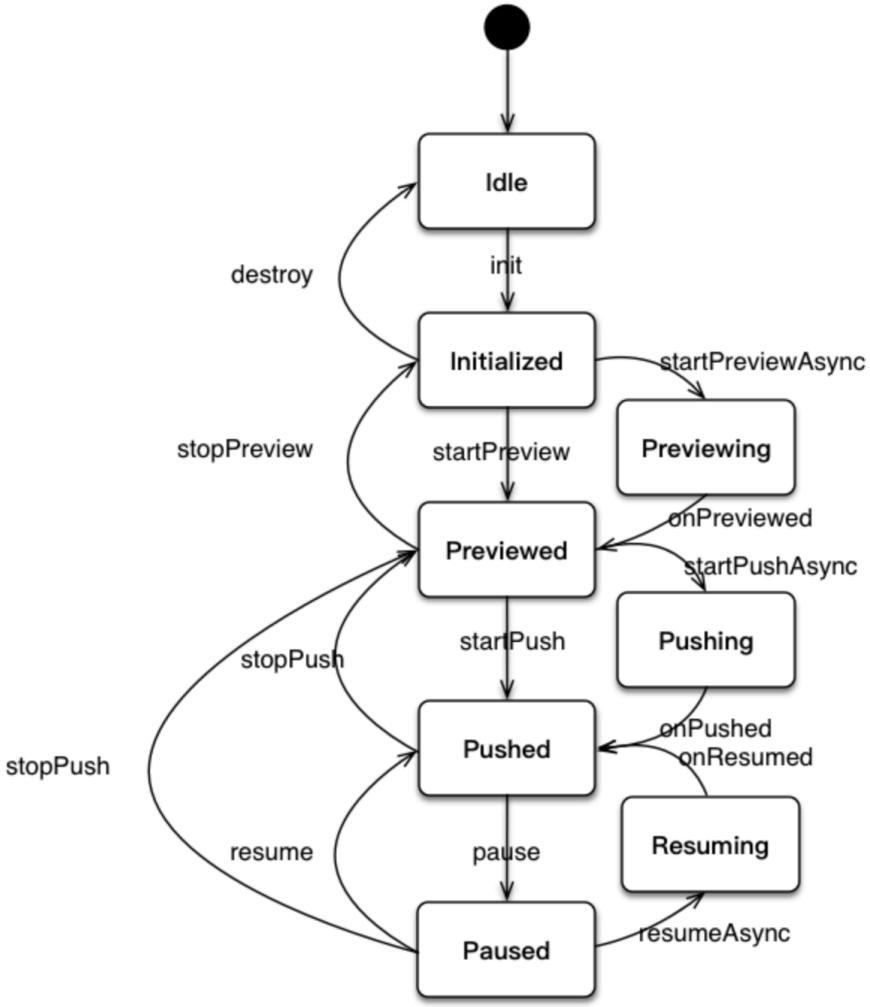
  检查混淆，确认已将SDK相关包名加入了不混淆名单中。

```
-keep class com.alibaba.livecloud.** { *;}
```

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

· 接口调用

- 同步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接口都会在发生错误或者调用顺序不对时 thorws 出异常，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：



· 关于包大小

- SDK大小：arm64-v8a：7.6M、armv7a：5.2M
- 集成SDK后apk会增加约5M

· 功能限制说明

- 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。

- 关于历史版本升级说明
  - 推流SDK V1.3升级至 [V3.0.0-3.3.3]、连麦SDK升级至推流 [V3.0.0-3.3.3]，请下载 升级说明文档。
  - 从V3.3.4版开始不支持推流V1.3版兼容，升级时建议您重新接入最新版SDK。
  - 推流SDK升级至V3.1.0，如果您未集成阿里云播放器SDK，需集成该SDK（已包含推流SDK下载包里面）。
  - 推流SDK升级至V3.2.0+，提供包含阿里云播放器的版本和不包含阿里云播放器两个版本。如果您需要背景音乐功能必须集成播放器，否则可以不使用播放器SDK。

2.5 关于Demo

本文介绍Android Demo相关内容。

Demo目录结构

Android Demo目录结构

- 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

java源码目录结构

V3.0.0版本 demo源码目录结构

- onekeyshare 分享源码
- java文件 ui控制源码

V1.3.0版本 demo源码目录结构

- adapter 数据适配器

- **demo ui控制源码**
- **utils 工具类**

3 iOS-推流SDK

3.1 概念说明

本文介绍iOS推流SDK的相关概念。

码率控制

码率控制实际上是一种编码的优化算法，用于根据网络情况实时控制视频流码流的大小。同样的视频编码格式，码流大，它包含的信息也就越多，那么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，导致视频帧堆积非常严重，我们可以通过丢弃视频帧，来缩短推流的延时。

动态库

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。等程序运行时，动态库才会被真正加载进来。

说明:

Xcode加载动态库需要加载在 Embedded Binaries 中，而不是 Linked Frameworks and Libraries中。

耳返

耳返是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。因为声音通过骨骼传入耳朵和通过空气传入耳朵差异很大，而主播有需求直接听到观众端的效果。

混音

混音是把多种来源的声音，整合至一个立体音轨或单音音轨中，SDK支持音乐和人声的混音。

3.2 使用流程

本文介绍iOS推流SDK的基本使用流程和直播答题使用流程。

基本使用流程

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。

直播答题使用流程

直播答题通过推流SDK、定制版OBS或者阿里云OpenAPI在直播流中插入SEI信息（题目信息），当播放器解析到SEI信息后，会回调给用户的App，此时用户就可以将题目的具体题目内容展示出来。具体流程如下：

详细接入参见 直播答题方案，按照说明提交工单申请接入。播放器接入参见 答题播放器。OBS推流参见 OBS使用说明。

3.3 SDK集成

本文介绍iOS推流SDK的快速上手集成。

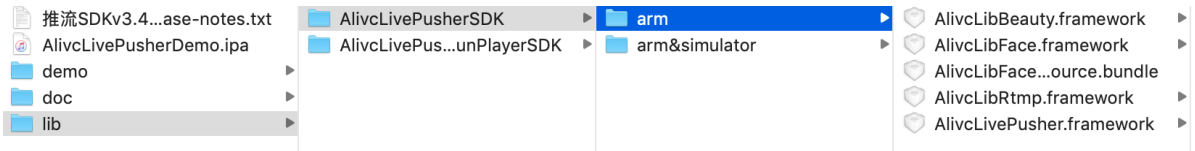
集成环境

硬性要求

名称	要求
iOS系统版本	>= 8.0
机器型号	iPhone 5S及以上
CPU架构支持	ARMv7、ARMv7s、ARM64
集成工具	Xcode 8.0及以上版本
bitcode	关闭

推流SDK下载

阿里云官网iOS版 推流SDK下载，推流SDK包含在解压包的AlivcLivePusher文件夹中，如下图所示：



上图中的文件内容作用如下：

文件目录	文件名称	文件说明
demo	AlivcLivePusherDemo	推流SDK演示Demo源代码工程
doc	html	推流SDK API 说明
lib	AlivcLivePusherSDK/arm	不带阿里云播放器的推流SDK，纯arm版本。
	AlivcLivePusherSDK/arm&simulator	不带阿里云播放器的推流SDK，arm+模拟器版本。
	AlivcLivePusherSDK+AliyunPlayerSDK/arm	带阿里云播放器的推流SDK，纯arm版本。
	AlivcLivePusherSDK+AliyunPlayerSDK/arm&simulator	带阿里云播放器的推流SDK，arm+模拟器版本。

上表中的sdk文件夹中的文件作用如下：

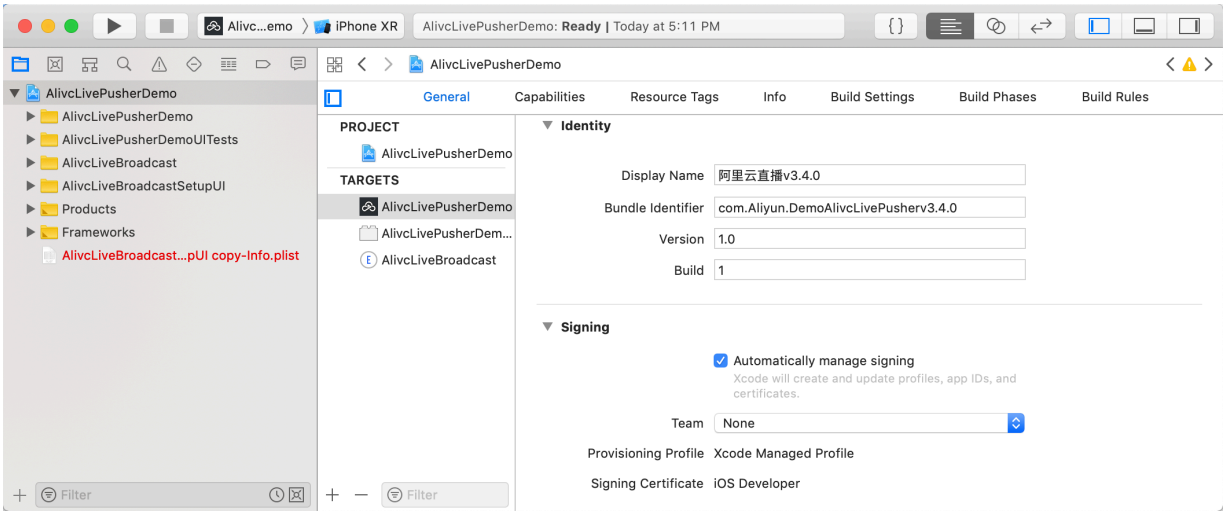
库文件	文件说明
AlivcLivePusher.framework	推流SDK
AlivcLibRtmp.framework	RTMPSDK
AlivcLibBeauty.framework	美颜库
AlivcLibFace.framework	人脸检测库
AlivcLibFaceResource.bundle	人脸检测资源文件
AliyunPlayerSDK.framework	播放器库
AliThirdparty.framework	播放器依赖的第三方库
AliyunLanguageSource.bundle	播放器依赖的资源文件

说明：

- 推流SDK中包含背景音乐相关功能。如果您需要使用该功能，要使用依赖播放器SDK的版本；如果您不需要背景音乐功能，则使用不依赖播放器SDK的版本即可。
- AlivcLibFaceResource.bundle是人脸识别资源文件，如果您需要使用美颜的人脸识别高级功能，则必须导入开发工程；反之则不需要。
- 每个版本均包含 arm 和 arm&simulator 两套SDK，arm仅支持真机调试。arm&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。

运行推流Demo

xcode打开AlivcLivePusherDemo工程，如下图所示：



即可直接运行工程查看Demo效果。效果如下：

推流url中填入有效的推流rtmp地址，推流成功后，拉流观看的效果可以使用阿里云播放器SDK、ffplay、VLC等工具查看。

推流SDK集成

1. 使用Cocoapods集成

   a. 修改Podfile

      A. 依赖播放器版本 添加如下语句至您的 Podfile文件中。

      ```
      pod 'AlivcLivePusherWithPlayer'
      ```

      B. 不依赖播放器版本 添加如下语句至您的 Podfile 文件中。

      ```
      pod 'AlivcLivePusher'
      ```

   b. 执行 pod install

   ```
   pod install :每次您编辑您的Podfile（添加、移除、更新）的时使用
   ```

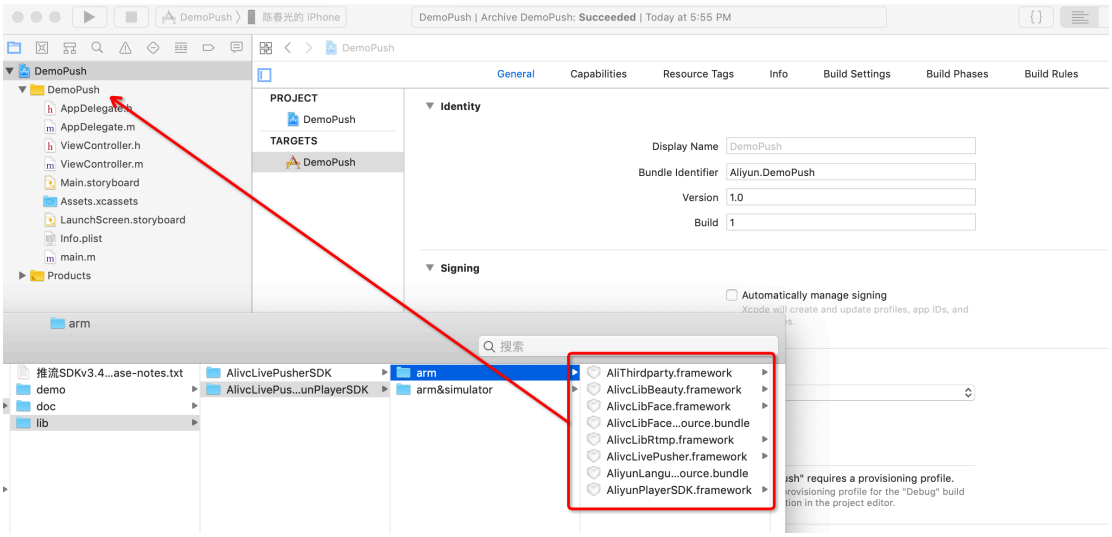
2. 手动导入集成

   a. 新建SDK测试工程 Single View App > DemoPush 。

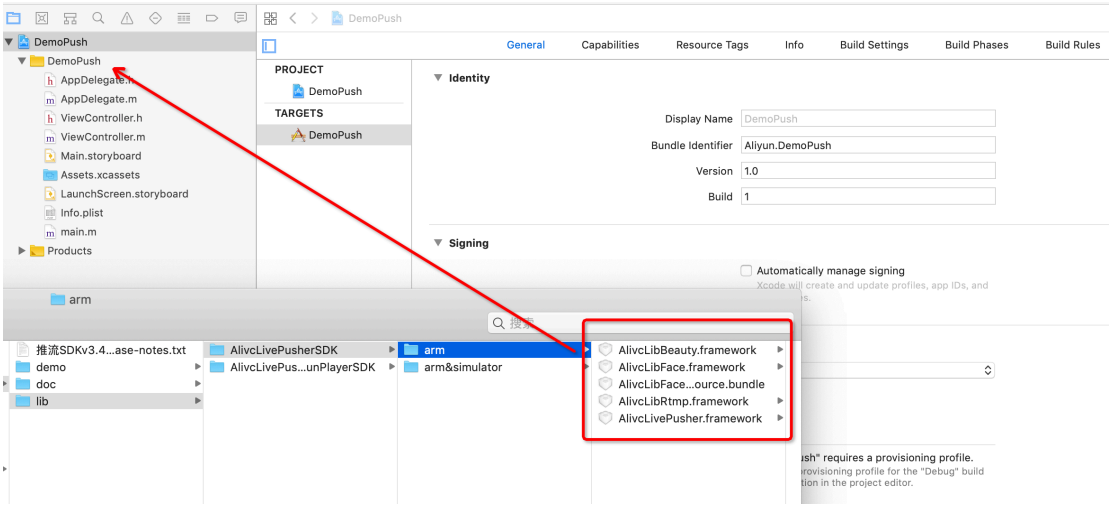
   b. 导入SDK

      · 分别将以下文件拖入您的Xcode工程中：

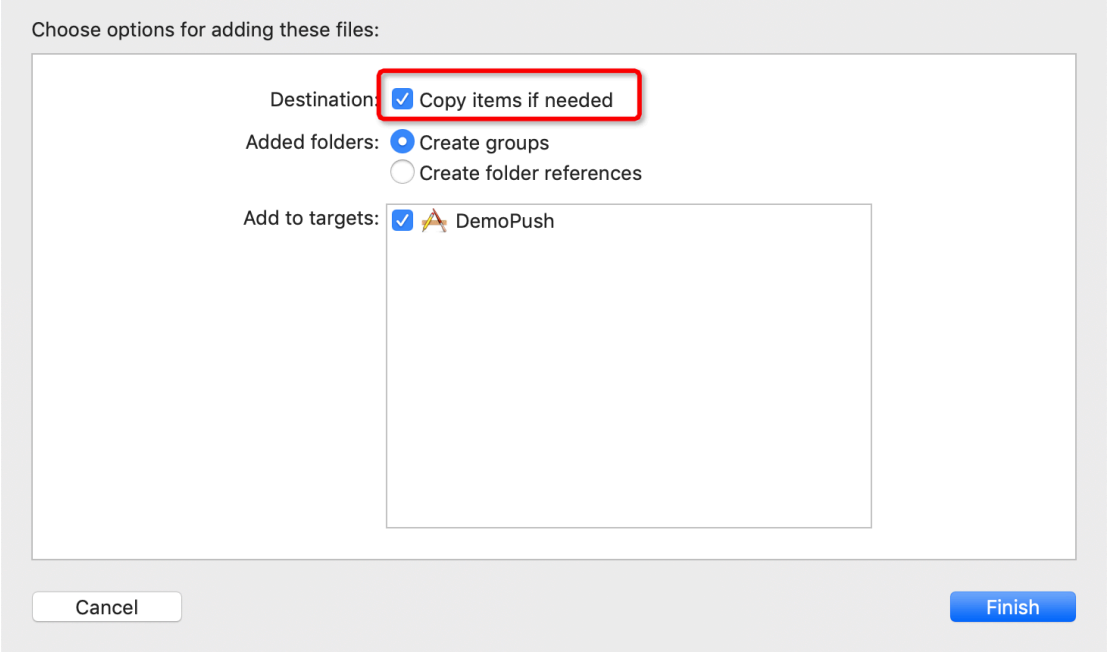
      - AlivcLibRtmp.framework
      - AlivcLivePusher.framework
      - AlivcLibBeauty.framework
      - AlivcLibFace.framework
      - AlivcLibFaceResource.bundle
      - AliyunPlayerSDK.framework
      - AliThirdparty.framework
      - AliyunLanguageSource.bundle



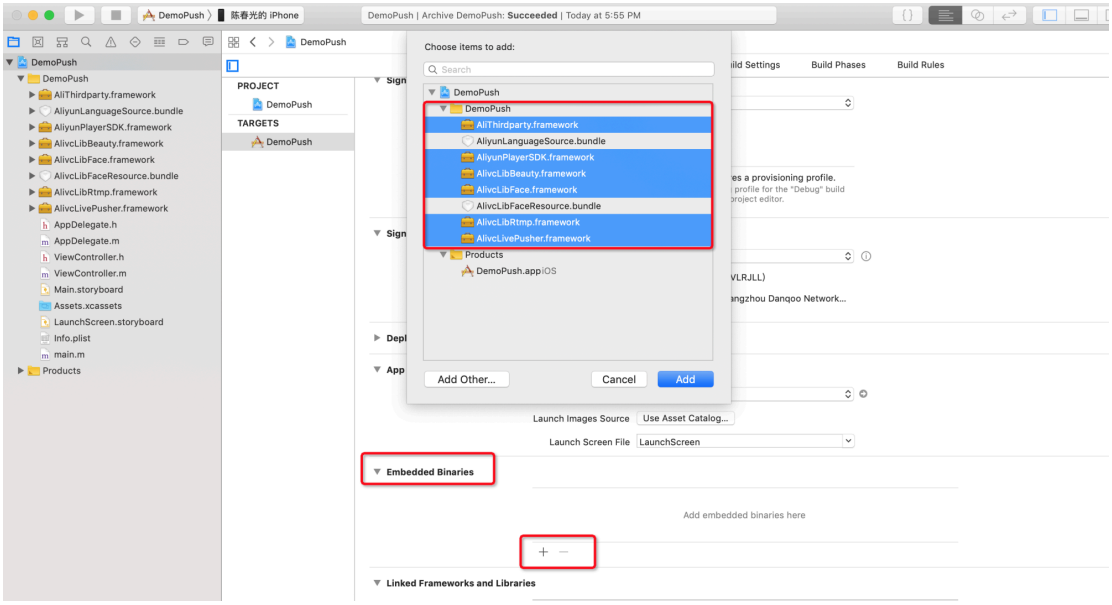
- 如果您不需要依赖播放器SDK的版本，则只需要将以下文件：
  - AlivcLibRtmp.framework
  - AlivcLivePusher.framework
  - AlivcLibBeauty.framework
  - AlivcLibFace.framework
  - AlivcLibFaceResource.bundle



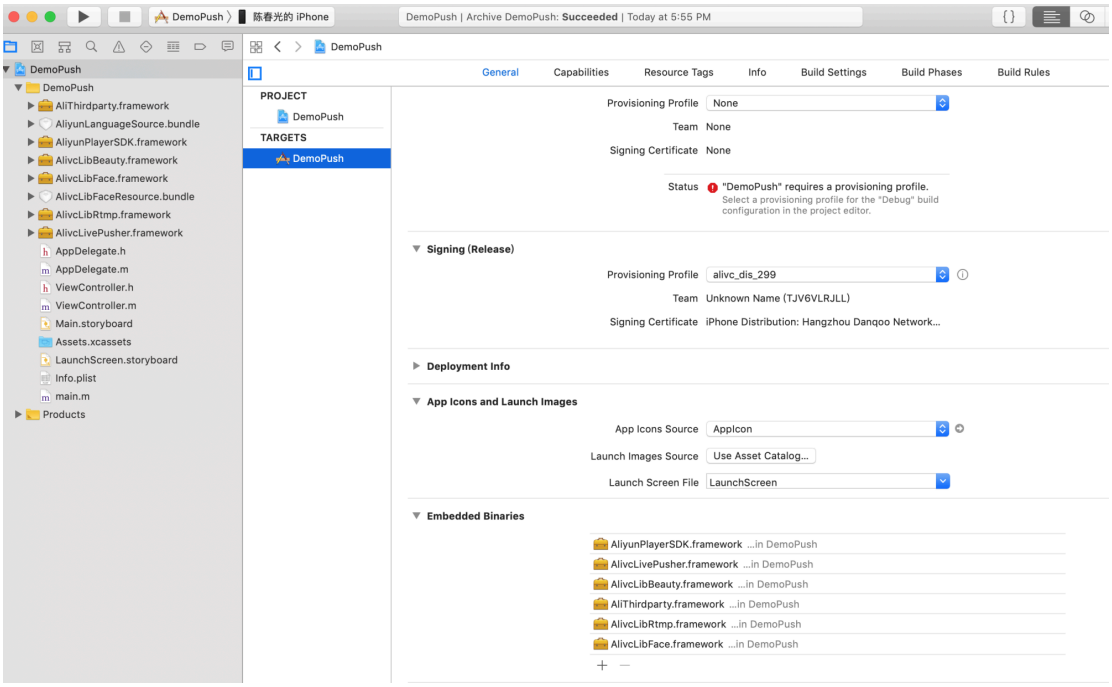
- 按图示勾选 Copy items if needed：



- 导入SDK成功之后，在 Xcode > General > Embedded Binaries中添加SDK依赖：

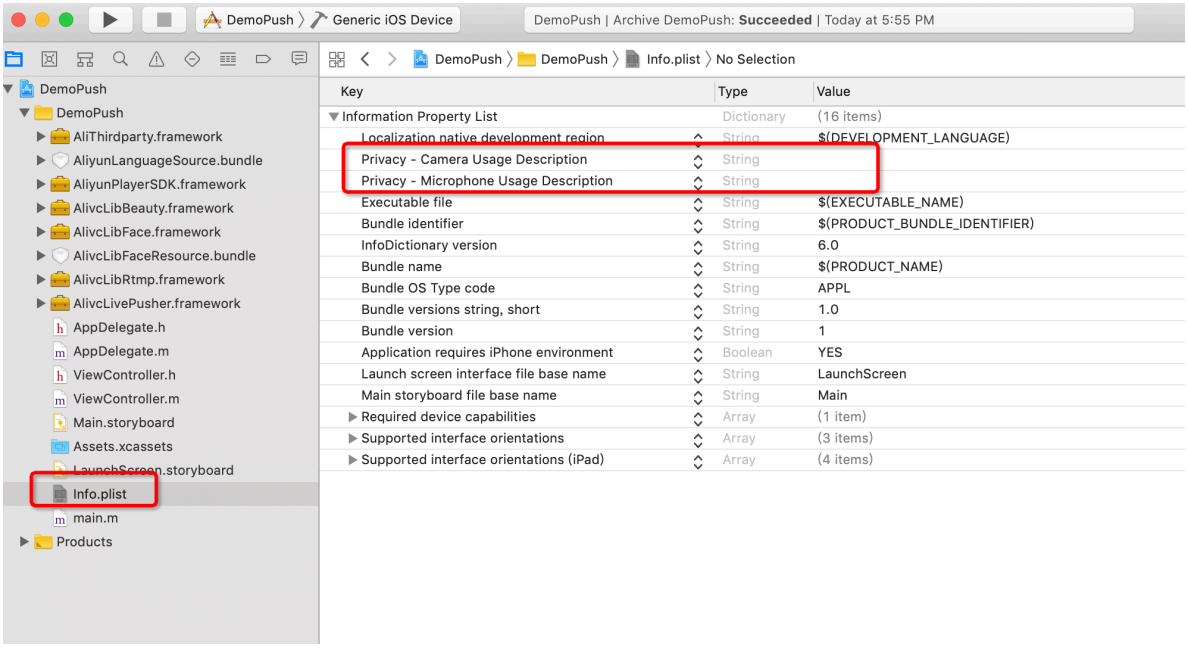


- 添加依赖成功后如图所示：

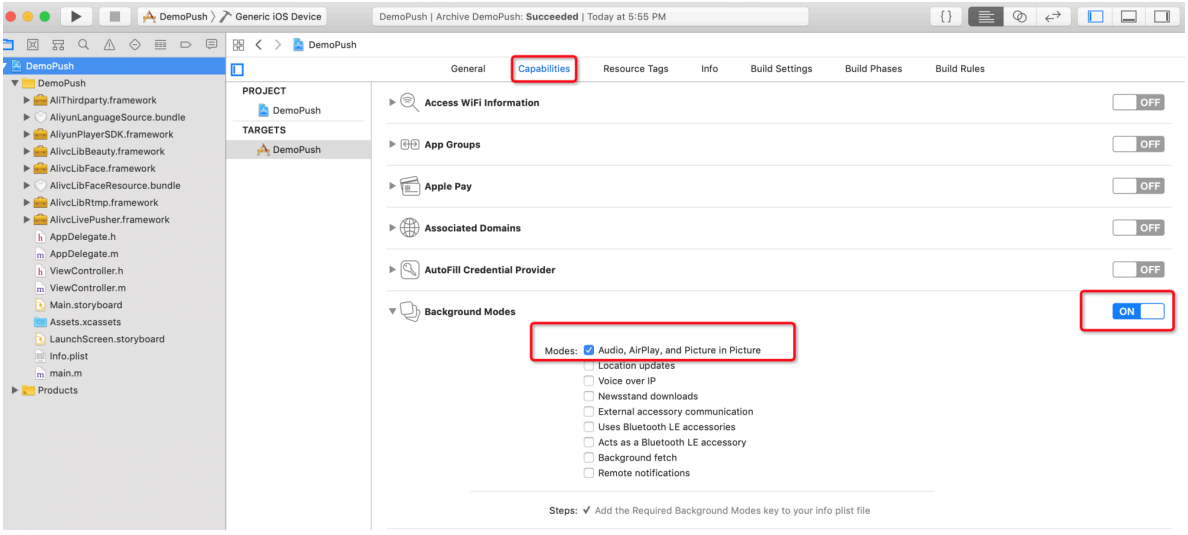

DemoPush 〉 陈春光的 iPhone
DemoPush | Archive DemoPush: Succeeded | Today at 5:55 PM
DemoPush
DemoPush
AliThirdparty.framework
AliyunLanguageSource.bundle
AliyunPlayerSDK.framework
AlivcLibBeauty.framework
AlivcLibFace.framework
AlivcLibFaceResource.bundle
AlivcLibRtmp.framework
AlivcLivePusher.framework
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
Products
DemoPush
General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
DemoPush
TARGETS
DemoPush
Provisioning Profile None
Team None
Signing Certificate None
Status "DemoPush" requires a provisioning profile.
Select a provisioning profile for the "Debug" build configuration in the project editor.
Signing (Release)
Provisioning Profile alivc_dis_299
Team Unknown Name (TJV6VLRJLL)
Signing Certificate iPhone Distribution: Hangzhou Danqoo Network...
Deployment Info
App Icons and Launch Images
App Icons Source AppIcon
Launch Images Source Use Asset Catalog...
Launch Screen File LaunchScreen
Embedded Binaries
AliyunPlayerSDK.framework ...in DemoPush
AlivcLivePusher.framework ...in DemoPush
AlivcLibBeauty.framework ...in DemoPush
AliThirdparty.framework ...in DemoPush
AlivcLibRtmp.framework ...in DemoPush
AlivcLibFace.framework ...in DemoPush

3. 添加请求权限

在 Info.plist 文件中添加麦克风和摄像头权限`Privacy - Microphone Usage Description`、`Privacy - Camera 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。

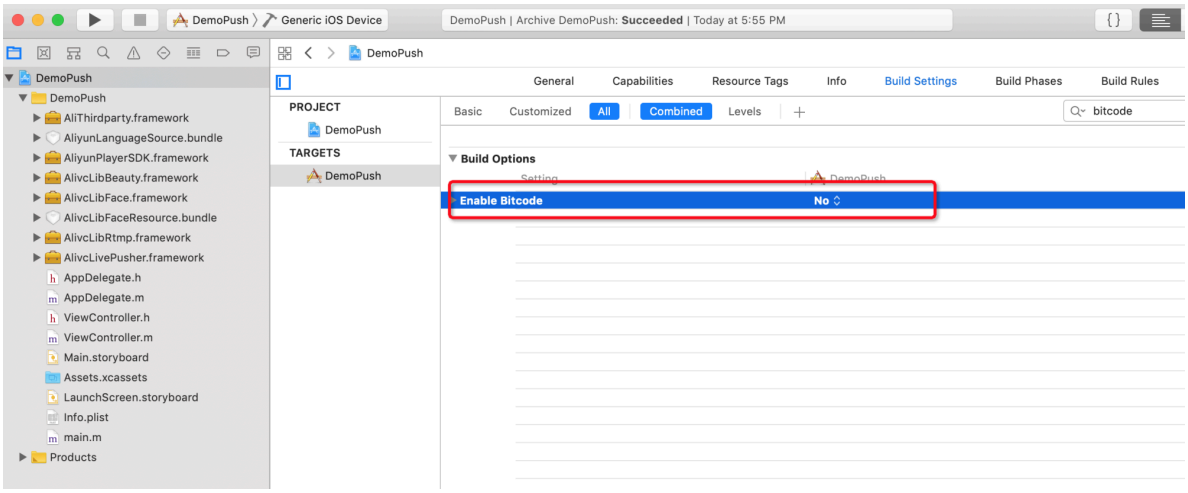


说明:

请您记得添加录音权限和相机权限。

4. 关闭bitcode

由于SDK不支持bitcode，所以需要在工程中关闭bitcode选项。



5. 具体使用说明

具体SDK的API使用请参考推流iOS SDK的使用说明文档，API的详细注释说明请参考SDK包里的API 文档。

3.4 SDK使用

本文详细说明如何使用iOS推流SDK API以及相关功能。本文介绍SDK的基本使用流程。

推流SDK功能列表

- 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
- 支持直播答题功能
- 支持视频截图功能
- 支持自动重连、异常处理

直播主要接口类列表

类	说明
AlivcLivePushConfig	推流初始设置
AlivcLivePusher	推流功能类
AlivcLivePusherErrorDelegate	错误回调
AlivcLivePusherNetworkDelegate	网络相关通知回调
AlivcLivePusherInfoDelegate	推流相关信息回调
AlivcLivePusherBGMDelegate	背景音乐回调
AlivcLivePusherCustomFilterDelegate	自定义滤镜回调
AlivcLivePusherCustomDetectorDelegate	自定义人脸检测回调
AlivcLivePusherSnapshotDelegate	截图回调

引入头文件

在代码中引入以下头文件便可使用直播推流的所有API和类对象。

```
#import <AlivcLivePusher/AlivcLivePusherHeader.h>
```

初始化

SDK 初始化主要包括以下几个步骤：

1. 创建AlivcLivePushConfig对象，并设置相关参数。

   AlivcLivePushConfig类用于推流的初始化设置，设置参数有：

参数	说明	默认
resolution	推流分辨率	540P
qualityMode	推流码率控制模式	分辨率优先模式
enableAutoBitrate	是否开启码率自适应模式	开启
enableAutoResolution	是否开启动态分辨率模式	关闭
fps	视频帧率	20fps
minFps	最小视频帧率	8fps
targetVideoBitrate	目标视频编码码率	800kbps
minVideoBitrate	最小视频编码码率	200kbps
initialVideoBitrate	初始视频编码码率	800kbps
audioBitrate	音频编码码率	64kbps

参数	说明	默认
audioSampleRate	音频采样率	32000hz
audioChannel	音频声道数	双声道
videoEncodeGop	视频GOP大小	2秒
connectRetryCount	推流断开自动重连次数	5次
connectRetryInterval	推流断开自动重连间隔	1000毫秒
sendDataTimeout	发送网络数据超时时间	3000毫秒
orientation	横竖屏设定	竖屏
cameraType	相机位置	前置
pushMirror	是否推流镜像	关闭
previewMirror	是否预览镜像	关闭
audioOnly	是否纯音频推流	关闭
videoOnly	是否纯视频推流	关闭
autoFocus	相机是否自动对焦	打开
beautyOn	是否打开美颜	打开
pauseImg	暂停退后台图片	空
networkPoorImg	弱网图片	空
beautyMode	美颜模式	普通美颜
beautyWhite	美白参数	70
beautyBuffing	磨皮参数	40
beautyRuddy	红润参数	40
beautyCheekPink	腮红参数	15
beautyThinFace	瘦脸参数	40
beautyShortenFace	收下巴参数	50
beautyBigEye	大眼参数	30
flash	是否打开手电筒	关闭
videoEncoderMode	视频编码器	硬编码
audioEncoderProfile	音频编码格式	AAC LC
audioEncoderMode	音频编码器	软编码
externMainStream	自定义外部主流开关	关闭
externVideoFormat	自定义外部视频数据格式	unknown

参数	说明	默认
externAudioFormat	自定义外部音频数据格式	unknown
previewDisplayMode	预览窗口显示模式	ASPECT FIT模式
addWatermarkWithPath	添加水印接口	
removeWatermarkWithPath	移除水印接口	
getAllWatermarks	获取所有水印信息	
getPushResolution	获取推流分辨率	

a. 创建AlivcLivePushConfig对象

```
self.pushConfig = [[AlivcLivePushConfig alloc]init];
```

b. 分辨率设置

```
self.pushConfig.resolution = AlivcLivePushResolution540P;
```

SDK支持的主播推流分辨率如下：

分辨率定义	分辨率
AlivcLivePushResolution180P	192x320
AlivcLivePushResolution240P	240x320
AlivcLivePushResolution360P	368x640
AlivcLivePushResolution480P	480x640
AlivcLivePushResolution540P	544x960
AlivcLivePushResolution720P	720x1280
AlivcLivePushResolutionPassThrough	屏幕分辨率

 说明:

AlivcLivePushResolutionPassThrough只有在录屏直播模式下才有效，使用的是屏幕原始分辨率。

c. 码率控制设置

推流的码率控制有三种qualityMode模式可选：清晰度优先模式、流畅度优先模式和自定义模式。

- 清晰度优先模式（AlivcQualityModeEnum.QM_RESOLUTION_FIRST）：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。
- 流畅度优先模式（AlivcQualityModeEnum.QM_FLUENCY_FIRST）：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。
- 自定义模式（AlivcQualityModeEnum.QM_CUSTOM）：SDK会根据开发者设置的码率进行配置。

其中，清晰度优先模式和流畅度优先模式是根据日常使用场景由SDK定义的一套比较适合该场景的推荐使用模式，这两种模式下帧率和编码码率都是由定义好的，不可以外部调节，所以使用的时候需要特别注意。由于不同模式下您可以设置的参数有限制，以下是具体的组合说明：

qualityMode 码率控制模式	fps 帧率	minVideoBitrate 视频最小码率	initialVideoBitrate 视频初始码率	targetVideoBitrate 视频目标码率
清晰度优先模式（默认）	不可设置	不可设置	不可设置	不可设置
流畅度优先模式	不可设置	不可设置	不可设置	不可设置
自定义模式	可设置	可设置	可设置	可设置

从上表中可以看出只有自定义模式下才能设置视频的fps、目标码率、最小码率以及初始码率，而在清晰度模式和流畅度模式下这些值的设置并不生效，而是SDK内部自定义一套和不同分辨率对应的值，详细如下表。

清晰度优先模式各视频参数对应值：

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcLivePushResolution180P	20	120	300	550

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcLivePushResolution240P	20	180	450	750
AlivcLivePushResolution360P	20	300	600	1000
AlivcLivePushResolution480P	20	300	800	1200
AlivcLivePushResolution540P	20	600	1000	1400
AlivcLivePushResolution720P	20	600	1500	2000

流畅度优先模式各视频参数对应值：

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcLivePushResolution180P	25	80	200	250
AlivcLivePushResolution240P	25	120	300	350
AlivcLivePushResolution360P	25	200	400	600
AlivcLivePushResolution480P	25	300	600	800
AlivcLivePushResolution540P	25	300	800	1000

分辨率	fps	minVideoBitrate 视频最小码率kbps	initialVideoBitrate 视频初始码率kbps	targetVideoBitrate 视频目标码率kbps
AlivcLivePushResolution720P	25	300	1000	1200

说明:

如果觉得清晰度优先模式和流畅度优先模式下的视频参数设置不满足推流场景要求，必须使用自定义模式来设置自定义的参数值。

码率自适应

码率自适应是推流码率控制中的一种策略，其定义表示推流过程中是否根据当前实际网络代码动态调整视频的编码码率，调整范围就是由最小码率、初始码率和，目标码率来限定的。

码率自适应打开时，推流过程中首先使用初始码率作为编码码率，当检测到当前上传带宽充足时，会慢慢上调编码码率，直至到目标码率就不再向上调整，而当检测到当前上传带宽不足时，开始下调视频编码码率，下调的最小值为最小码率，此时不再继续下调。

当码率自适应关闭时，整个推流过程中始终使用初始编码码率，不会动态去调整编码码率。

动态分辨率

动态分辨率是在推流过程中，根据码率调整到一定的范围后，可以动态切换相应的分辨率，也就是说此设置是依赖码率自适应的开关的，如果码率自适应关闭，此设置无效，且动态分辨率目前只支持清晰度优先模式和流畅度优先模式，不支持自定义模式。当动态分辨率开关生效后，对应的分辨率切换范围为:

分辨率	最小码率kbps	最大码率kbps率
AlivcLivePushResolution480P	无	1000
AlivcLivePushResolution540P	1000	1200

分辨率	最小码率kbps	最大码率kbps率
AlivcLivePushResolution720P	1200	无

目前只支持分辨率在480P、540P和720P中切换，当码率调整在相应的区间中时，会动态调整成相应的分辨率，注意分辨率调整只能由初始分辨率向下调整，也就是说如果初始设置分辨率为540P，那么动态分辨率只能最大调整到540P。

说明:

动态分辨率在某些播放器上可能无法正常播放，阿里云播放器已支持。使用动态分辨率功能时，建议您使用阿里云播放器。

d. 纯音视频推流设置

SDK支持纯音频和纯视频推流设置，通过打开audioOnly，SDK就会只推纯音频流，如果打开videoOnly，SDK只推纯视频流。不可以同时打开这两种模式。

示例代码如下:

```
self.pushConfig.audioOnly = YES;//设置纯音频推流
```

```
self.pushConfig.videoOnly = YES;//设置纯视频推流
```

e. 推流图片设置

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在退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:

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

另外，当网络较差时用户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:

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

说明:

此处的图片只支持png格式，且在app资源文件中能够访问到的图片，不支持网络图片。

f. 水印设置

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，目前最多支持3个水印。示例代码如下：

```
NSString *watermarkBundlePath = [[NSBundle mainBundle] pathForRes
ource:
                                  [NSString stringWithFormat:@"
watermark"]
                                  ofType:@"png"];//设置水印图片路径
[self.pushConfig 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值为推流画面的10%位置，即推流分辨率为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。

g. 显示模式设置

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。

说明：

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。

2. 创建AlivcLivePusher对象

当AlivcLivePushConfig对象创建成功并设置了初始化参数后，根据AlivcLivePushConfig对象来创建AlivcLivePusher对象。示例代码如下：

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

3. 注册回调

推流主要回调分为Info、Error、Network、背景音乐、截图、自定义滤镜和自定义人脸检测回调，注册delegate可接受对应的回调。示例代码如下：

```
[self.livePusher setInfoDelegate:self];
[self.livePusher setErrorDelegate:self];
[self.livePusher setNetworkDelegate:self];
[self.livePusher setBGMDelegate:self];
[self.livePusher setSnapshotDelegate:self];
[self.livePusher setCustomFilterDelegate:self];
[self.livePusher setCustomDetectorDelegate:self];
```

预览

1. 开始预览

初始化成功之后，就是进入预览的环节，预览使用外部传入一个UIView，调用startPreview接口后SDK内部会自动打开相机采集，并将采集画面渲染到预览 UIView上。示例代码如下：

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

说明：

如果APP没有打开过系统相机使用权限，这一步会要求打开系统相机权限，如果选择禁止打开相机，会导致预览失败。

2. 停止预览

调用stopPreview可以停止预览，此时画面会停留在最后一帧。示例代码如下：

```
[self.livePusher stopPreview];
```

推流

1. 开始推流

预览成功后，调用startPushWithURL接口开始推流，startPushWithURL接口需要传入有效的rtmp推流url地址，阿里云推流地址获取参见 快速开始。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，拉流地址如何获取参见 快速开始。示例代码如下：

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

2. 停止推流

通过调用stopPush接口可以停止推流。示例代码如下：

```
[self.livePusher stopPush];
```

3. 重新推流

通过调用restartPush可以在推流中或者推流出错后，重新开始推流。示例代码如下：

```
[self.livePusher restartPush];
```

暂停恢复

在推流过程中，通过调用pause和resume接口来控制推流的暂停和恢复。

1. 暂停

暂停是指推流过程中调用pause接口暂停相机的采集，此时的推流行为是只采集音频数据，视频数据不采集，所以此时推流只推音频数据，不推视频数据，但是如果用户设置了pauseImg，此时视频会推设置的图片数据。示例代码如下：

```
[self.livePusher pause];
```

2. 恢复

恢复就是通过调用resume接口恢复正常推流状态。示例代码如下：

```
[self.livePusher resume];
```

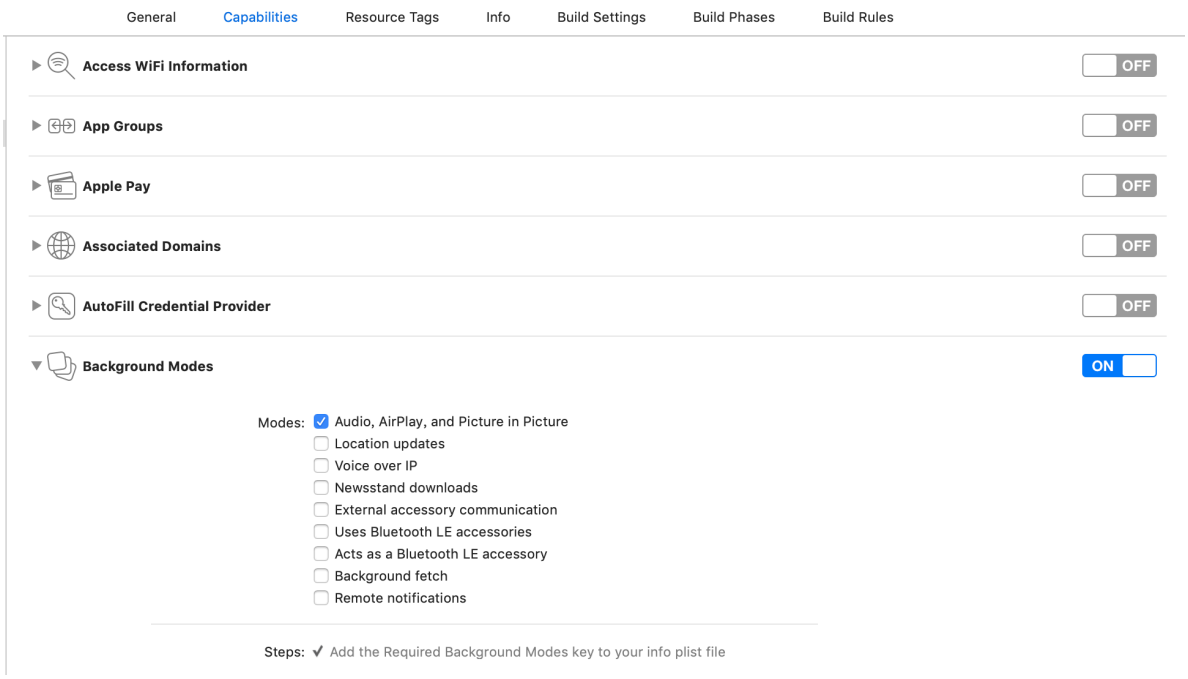
前后台切换

当应用受到其他系统行为的影响比如来电话等会导致应用出现前后台切换的操作，以及用户主动切换前后台操作都会影响推流的行为。

由于受到系统的限制，在应用退到后台时，相机采集会停止工作，此时用两种选择，是否需要退后台继续推流，就是在后台推流不中断。

1. 后台继续推流

首先需要在app 的Capabilities中打开Background Modes 中的Audio，AirPlay，and Picture in Picture 模式，支持后台音频访问，保持app在后台工作。



当app进入后台的时候，目前SDK已经侦听了该系统通知，并对推流做暂停处理，此时推流行为和暂停表现一致，也就是只推音频，不推视频，如果设置了背景图片的话视频推背景图片。App不用做任何处理，会自动在后台推流。

回到前台时，SDK也会自动恢复推流，不需要app做任何处理。

2. 后台停止推流

当app切换到后台需要停止推流时，需要在以下系统通知中做停止推流处理。示例代码如下：

```
- (void)applicationWillResignActive:(NSNotification *)notification {
    // 如果退后台不需要继续推流，则停止推流
    if ([self.livePusher isPushing]) {
        [self.livePusher stopPush];
    }
}
```

注意此时推流已经中断，回到前台需要继续推流时，需要做以下操作。示例代码如下：

```
- (void)applicationDidBecomeActive:(NSNotification *)notification {

    [self.livePusher startPushWithURLAsync:self.pushURL];
}
```

重连

重连操作包括自动重连和手动重连两种。

1. 自动重连

   在SDK内部检测到socket连接失败或者断开的时候，会有自动重连机制，在AlivcLivePushConfig中的connectRetryCount重连次数和connectRetryInterval重连间隔用来控制自动重连的次数和每次之间的间隔，当所有重连次数用完还没有重连成功时，会上报重连失败错误后不再继续重连。

2. 手动重连

   手动重连是外部调用SDK接口reconnectPushAsync接口进行手动重连。

美颜控制

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、腮红等功能。

目前SDK 美颜采用外置对接方式，也就是说APP 需要对接实现美颜的回调才能使用美颜功能。首先app需要加载美颜的库文件AlivcLibBeauty.framework、人脸检测库AlivcLibFace.framework以及人脸检测资源包AlivcLibFaceResource.bundle。

然后实现美颜和人脸检测的回调。示例代码如下：

```
/**
 外置美颜滤镜创建回调
 */
- (void)onCreate:(AlivcLivePusher *)pusher context:(void*)context
{
    [[AlivcLibBeautyManager shareManager] create:context];
}

/**
 外置美颜滤镜更新参数回调
 */
- (void)updateParam:(AlivcLivePusher *)pusher buffing:(float)buffing
 whiten:(float)whiten pink:(float)pink cheekpink:(float)cheekpink
thinface:(float)thinface shortenface:(float)shortenface bigeye:(float)
bigeye
{
    [[AlivcLibBeautyManager shareManager] setParam:buffing whiten:
whiten pink:pink cheekpink:cheekpink thinface:thinface shortenface:
shortenface bigeye:bigeye];
}

/**
 外置美颜滤镜开关回调
 */
- (void)switchOn:(AlivcLivePusher *)pusher on:(bool)on
{
    [[AlivcLibBeautyManager shareManager] switchOn:on];
}

/**
 外置美颜滤镜开关回调
 */
```

```
- (void)switchOn:(AlivcLivePusher *)pusher on:(bool)on
{
    [[AlivcLibBeautyManager shareManager] switchOn:on];
}

/**
 通知外置滤镜销毁回调
 */
- (void)onDestory:(AlivcLivePusher *)pusher
{
    [[AlivcLibBeautyManager shareManager] destroy];
}

/**
 外置人脸检测创建回调
 */
- (void)onCreateDetector:(AlivcLivePusher *)pusher
{
    [[AlivcLibFaceManager shareManager] create];
}

/**
 外置人脸检测处理回调
 */
- (long)onDetectorProcess:(AlivcLivePusher *)pusher data:(long)data w:
(int)w h:(int)h rotation:(int)rotation format:(int)format extra:(long)
extra
{
     return [[AlivcLibFaceManager shareManager] process:data width:w
height:h rotation:rotation format:format extra:extra];
}

/**
 外置人脸检测销毁回调
 */
- (void)onDestoryDetector:(AlivcLivePusher *)pusher
{
    [[AlivcLibFaceManager shareManager] destroy];
}
```

通过设置AlivcLivePushConfig中的相关美颜参数，或者调用美颜相关接口都可以控制美颜开关、是否使用高级美颜以及调节美颜参数。

基本设置相关代码。示例代码如下：

```
self.pushConfig.beautyOn = true; // 开启美颜
self.pushConfig.beautyMode = AlivcLivePushBeautyModeProfessional; //设定
为高级美颜
self.pushConfig.beautyWhite = 70; // 美白范围0-100
self.pushConfig.beautyBuffing = 40; // 磨皮范围0-100
self.pushConfig.beautyRuddy = 40;// 红润设置范围0-100
self.pushConfig.beautyBigEye = 30;// 大眼设置范围0-100
self.pushConfig.beautyThinFace = 40;// 瘦脸设置范围0-100
self.pushConfig.beautyShortenFace = 50;// 收下巴设置范围0-100
self.pushConfig.beautyCheekPink = 15;// 腮红设置范围0-100
```

动态更改可以通过接口setBeautyOn、setBeautyWhite、setBeautyBuffing、setBeautyRuddy、setBeautyCheekPink、setBeautyThinFace、setBeautyShortenFace和setBeautyBigEye来调节美颜相关设置。

说明:

由于美颜采用了外置对接方式，这样很方便app对接第三方的美颜和人脸检测的滤镜，只要在对应的回调处理中调用第三方滤镜的相关接口就能使SDK中完美对接第三方的美颜滤镜。

我们提供了丰富的美颜参数，建议您在UED同事配合下，调整出最符合自己应用风格的美颜参数。

以下几组美颜效果比较理想，您可参考下表：

档位	磨皮	美白	红润	大眼	瘦脸	收下巴	腮红
一档	40	35	10	0	0	0	0
二挡	60	80	20	0	0	0	0
三挡	50	100	20	0	0	0	0
四挡	40	70	40	30	40	50	15
五档	70	100	10	30	40	50	0

相机控制

相机控制是包括切换相机、获取当前相机位置、对焦、缩放、曝光和手电筒操作。

1. **切换相机**

   切换前置后置相机，预览和直播中都可以切换。

```
[self.livePusher switchCamera];
```

2. **设置手电筒**

```
[self.livePusher setFlash:YES];
```

3. **自动对焦**

```
[self.livePusher setAutoFocus:YES];
```

4. **手动对焦**

```
[self.livePusher focusCameraAtPoint:point needAutoFocus:YES];
```

5. **设置缩放倍数**

缩放接口为增量设置，每次设置参数都是在当前基础上做增减，缩放倍数范围（1-3）。

```
[self.livePusher setZoom:1.0];//当前的缩放倍数+1
```

6. 获取缩放倍数

```
self.zoom = [self.livePusher getCurrentZoom];
```

7. 获取最大缩放倍数

```
self.zoom = [self.livePusher getMaxZoom];
```

8. 设置曝光度

```
self.zoom = [self.livePusher setExposure:self.exposure];
```

9. 获取相机最小曝光度值

```
[self.livePusher getSupportedMinExposure];
```

10.获取相机最大曝光度值

```
[self.livePusher getSupportedMaxExposure];
```

11.获取当前曝光度值

```
[self.livePusher getCurrentExposure];
```

截图

截图功能是在推流和预览中可以截取视频生成图片，通过调用接口snapshot来实现，snapshot中有两个参数count和interval， count 表示要截取图片的个数，interval表示每张图片的间隔，如果需要立刻截取当前一张视频图片，只要传递count为1，interval为0即可。示例代码如下：

```
[self.livePusher snapshot:self.count interval:self.interval];
```

截图是异步代理回调操作，截图结果通过代理AlivcLivePusherSnapshotDelegate回调给app，app必须实现代理中的onSnapshot方法。示例代码如下：

```
- (void)onSnapshot:(AlivcLivePusher *)pusher image:(UIImage *)image {
}
```

背景音乐

该功能提供背景音乐播放、混音、降噪、耳返、静音等功能，通过AlivcLivePusher中的如下接口实现以上功能：

```
/*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];
```

说明:

该功能只能在开始预览之后才可调用，目前只支持mp3格式音乐文件。

自定义音视频流输入

该功能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、第三方设备采集的音视频数据等。具体功能使用介绍如下：

1. 全局配置自定义音视频输入

   在推流配置AlivcLivePushConfig中配置使用自定义音视频输入，也可包括与自定义输入流相关的其他配置，具体配置如下：

   ```
   self.pushConfig.externMainStream = true;//开启允许外部流输入
   self.pushConfig.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。
   ```

```
self.pushConfig.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。
```

其中，视频图像格式目前支持：

- IMAGE_FORMAT_BGRA
- IMAGE_FORMAT_RGBA
- IMAGE_FORMAT_YUVNV12
- IMAGE_FORMAT_YUVNV21
- IMAGE_FORMAT_YUV420P

音频帧格式目前只支持PCM 格式，且为SOUND_FORMAT_S16。

2. 插入自定义音视频数据

- 插入yuv 或者rgba视频数据，使用sendVideoData接口。

```
[self.livePusher sendVideoData:yuvData
                         width:720
                         height:1280
                         size:dataSize
                         pts:nowTime
                         rotation:0];
```

- 插入CMSampleBufferRef格式视频数据，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*)nativeBuffer {
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;
}
```

- **插入音频PCM数据**

```
[self.livePusher sendPCMData:pcmData size:size pts:nowTime];
```

混流功能

混流顾名思义就是把一路或多路音视频流混合成单路流推到流媒体服务器。具体使用介绍如下：

1. 添加混流

添加混流包括混流格式设置、旋转角度设置、混流位置、音频采样率设置等，添加后获取到的返回值用于标示这一路混流，假如返回-1则混流失败，具体代码调用示例如下：

```
//添加视频混流
int mixId = [self.livePusher addMixVideo:IMAGE_FORMAT_BGRA // 视频格
式
                          width:720 //视频原始宽度
                         height:480 //视频原始高度
                       rotation:90  //旋转90度
                       displayX:0.5 //相对x坐标
                       displayY:0.5 //相对y坐标
                       displayW:0.5 //相对主视频的宽度
                       displayH:0.5 //相对主视频的高度
                   adjustHeight:NO];//不按照高度一致对齐（多路混流的场景）
```

```
//添加音频混流
int mixId = [self.livePusher addMixAudio:2 //声道数
                                  format:SOUND_FORMAT_S16 //格式
                              audioSample:44100]; //采样率
```

2. 更新混流位置

已经添加过的视频混流可以修改其显示位置。

```
[self.livePusher changeMixVideoPosition:mixId
                    displayX:0.5
                    displayY:0.5
                    displayW:0.5
                    displayH:0.5];
```

3. 输入混流数据

确认添加好混流后，将已获得的混流返回值用于实际音视频流数据输入，具体示例如下：

```
//输入视频混流数据
[self.livePusher inputMixVideoData:mixId //混流id
                              data:dataptr //数据指针
                             width:width //长度
                            height:height //宽度
                            stride:stride //步幅
                              size:size   //数据长度
                               pts:pts    //pts
                          rotation:rotation]; //旋转角度

//输入音频混流数据
 [self.livePusher inputMixAudioData:mixID //混流id
                              data:dataptr //数据指针
                              size:size //数据长度
                               pts:pts]; //pts
```

4. 删除混流

清理已有混流，包括移除音频和视频，具体示例如下：

```
//移除已添加的一路视频混流
[self.livePusher removeMixVideo:mixID];

//移除已添加的一路音频混流
[self.livePusher removeMixAudio:mixID];
```

直播答题

该功能可以通过在直播流里面插入SEI信息并发送后阿里云播放器SDK可收到此SEI消息，解析后做具体展示。另外为了保证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，这个过程中播放器会对相同的SEI进行去重处理。示例代码如下：

```
/*
 * 推流端发送自定义消息
 *
 * @param info          需要插入流的SEI消息体，建议是json格式
 * @param repeatCount   发送的帧数
 * @param delayTime     延时多少毫秒发送
```

```
 * @param KeyFrameOnly 是否只发送关键帧
 */

[self.livePusher sendMessage:@"题目信息" repeatCount:100 delayTime:0
KeyFrameOnly:false];
```

说明:

只有在推流状态下，才能调用此接口。

录屏直播

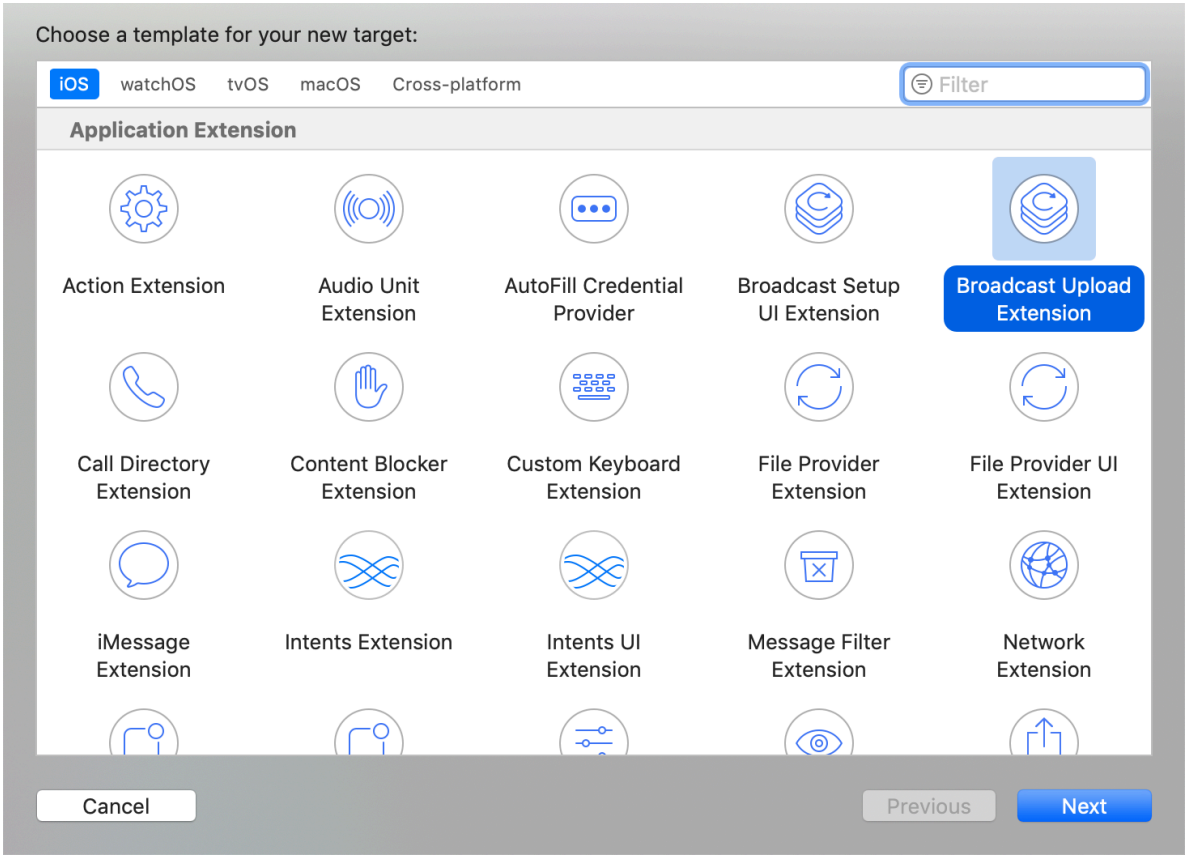
iOS 系统从iOS11版本开始的replaykit2中增加了系统录屏直播的功能，此功能可以录制手机的整个屏幕的音视频和麦克风采集内容并对接第三方扩展，而不受任何的限制，通过实现录屏的扩展对接阿里云直播SDK，可以非常方便地实现iOS手机的录屏直播功能。

说明:

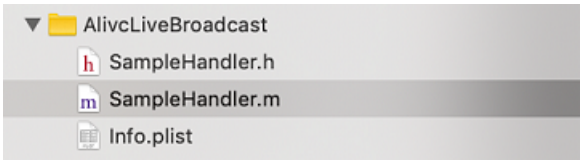
手机需要升级到iOS11及以上系统。

1. 创建录屏扩展

在APP中创建新的target，选择 Broadcast Upload Extension, 创建该App 对应的录屏直播扩展。

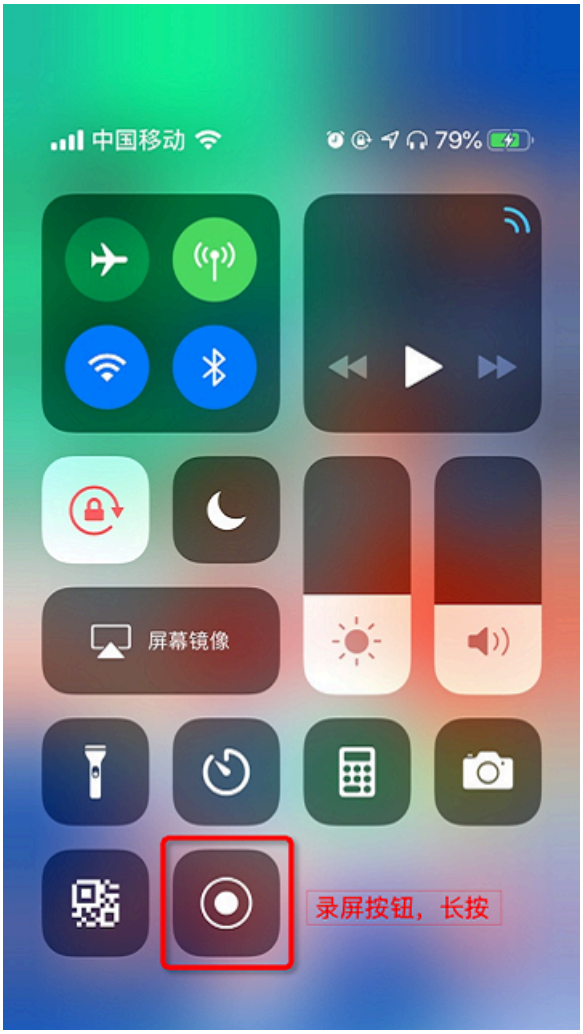


创建成功后，会在工程项目中出现以下扩展相关内容。



此时运行安装APP后，该扩展会一起和APP安装到手机上。

如下图，打开系统录屏菜单，找到录屏按钮。如果找不到录屏按钮，请在设置 > 控制中心 > 自定控制中添加"屏幕录制"选项。



长按录屏按钮后会跳出选项，刚刚创建的录屏扩展就在选项中，选中扩展，按下开始直播按钮，通过debug就会发现系统开始录屏并将录屏的音视频数据通过SampleHandler.m 中的接口processSampleBuffer回调给录屏扩展。

2. 录屏扩展对接阿里云推流SDK

当录屏扩展能够接收到录屏的音视频数据后，下一步就是通过阿里云直播SDK就录屏数据直播出去，在扩展中加载直播SDK头文件并初始化 AlivcLivePushConfig 和AlivcLivePusher对象。注意因为录屏直播使用的是系统推送的音视频数据进行推流直播，所以在配置直播config时，需要打开外部数据推流设置。

开启录屏直播示例代码如下：

```
- (void)startLivePush {
    @synchronized(self) {
        _pushConfig = [[AlivcLivePushConfig alloc]init];

        //设置外部数据推流
        _pushConfig.externMainStream = true;

        //设置视频输入格式
```

```
        _pushConfig.externVideoFormat = AlivcLivePushVideoFo
rmatYUVNV12;

        //设置输出分辨率
        _pushConfig.resolution = self.resolution;

        //设置音频输出
        _pushConfig.audioSampleRate = 44100;
        _pushConfig.audioChannel = 1;

        //设置输出横屏,默认竖屏
        _pushConfig.orientation =  self.orientation;

        //关闭美颜
        _pushConfig.beautyOn = false;

        //创建推流
        _livePusher = [[AlivcLivePusher alloc] initWithConfig:self.
pushConfig];

        //设置推流地址，开始推流
        [_livePusher startPushWithURL:self.pushUrl];

    }
}
```

由于是系统回调的录屏音视频数据传递给录屏扩展，需要讲音视频数据输入给推流SDK，通过接口sendVideoSampleBuffer和sendAudioSampleBuffer可以讲音视频数据发送给SDK进行推流。示例代码如下：

```
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType
:(RPSampleBufferType)sampleBufferType {
     @synchronized(self) {
        switch (sampleBufferType) {
            case RPSampleBufferTypeVideo:
                // Handle video sample buffer
                [self.livePusher sendVideoSampleBuffer:sampleBuffer
];
                break;
            case RPSampleBufferTypeAudioApp:
                // Handle audio sample buffer for app audio
                [self.livePusher sendAudioSampleBuffer:sampleBuffer
withType:sampleBufferType];
                break;
            case RPSampleBufferTypeAudioMic:
                // Handle audio sample buffer for mic audio
                [self.livePusher sendAudioSampleBuffer:sampleBuffer
withType:sampleBufferType];
                break;

            default:
                break;
        }
     }
}
```

停止录屏直播封装。示例代码如下：

```
- (void)stopLivePush {
```

```
    if(self.livePusher) {
        [self.livePusher stopPush];
        [self.livePusher destory];
        self.livePusher = nil;
    }
}
```

3. 录屏直播控制

由于扩展和APP是两个独立进程运行，所以录屏直播的控制需要通过APP 和扩展之间进程间CFNotificationCenter通信才能完成，详细可以参考demo中的具体实现。

4. 问题说明

- 由于iOS系统对扩展使用内存的限制，扩展的内存大小不能超过50M，所以在使用时建议录屏分辨率设置为passthrough模式，保持屏幕原始分辨率直播，因为SDK内部进行分辨率转换时会增加额外的内存开销，容易导致扩展内存超过最大限制的风险。
- 系统的锁屏、来电等一些操作会强制中断录屏操作，此时推流也会一并中断，需要重新启动录屏直播才能继续。
- 录屏时，一些APP前后台操作会影响APP声音的采集，就是当前有声音的APP在前台时录屏直播声音正常，但是此时其他APP切换到前台时，之前的APP的声音就会被禁音，这是iOS系统的录屏采集行为。

接口回调处理

1. 错误回调处理

当收到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：销毁当前直播重新创建、调用`restartPush/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描述为：capture camera open failed.
  - APP没有摄像头权限错误描述为：capture mic open failed.

AlivcLivePusherErrorDelegate接口回调如下：

```
/**
 系统错误回调

 @param pusher 推流AlivcLivePusher
 @param error error
 */
- (void)onSystemError:(AlivcLivePusher *)pusher error:(AlivcLiveP
ushError *)error;


/**
```

```
 SDK错误回调

 @param pusher 推流AlivcLivePusher
 @param error error
 */
- (void)onSDKError:(AlivcLivePusher *)pusher error:(AlivcLiveP
ushError *)error;
```

2. 网络监听回调处理

当您收到AlivcLivePusherNetworkDelegate回调时：

- 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，此时推流仍在继续并没有中断。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 onSendData Timeout。有两种处理方式，您只需选择其一：销毁当前推流重新创建或调用 reconnectA sync 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。

  说明:
  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会在继续重连。
- 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。

AlivcLivePusherNetworkDelegate接口回调如下：

```
/**
 网络差回调

 @param pusher 推流AlivcLivePusher
 */
- (void)onNetworkPoor:(AlivcLivePusher *)pusher;


/**
 推流链接失败

 @param pusher 推流AlivcLivePusher
 @param error error
 */
- (void)onConnectFail:(AlivcLivePusher *)pusher error:(AlivcLiveP
ushError *)error;
```

```
/**
 网络恢复

 @param pusher 推流AlivcLivePusher
 */
- (void)onConnectRecovery:(AlivcLivePusher *)pusher;


/**
 重连开始回调

 @param pusher 推流AlivcLivePusher
 */
- (void)onReconnectStart:(AlivcLivePusher *)pusher;


/**
 重连成功回调

 @param pusher 推流AlivcLivePusher
 */
- (void)onReconnectSuccess:(AlivcLivePusher *)pusher;

/**
 连接被断开

 @param pusher 推流AlivcLivePusher
 */
- (void)onConnectionLost:(AlivcLivePusher *)pusher;


/**
 重连失败回调

 @param pusher 推流AlivcLivePusher
 @param error error
 */
- (void)onReconnectError:(AlivcLivePusher *)pusher error:(AlivcLiveP
ushError *)error;


/**
 发送数据超时

 @param pusher 推流AlivcLivePusher
 */
- (void)onSendDataTimeout:(AlivcLivePusher *)pusher;


/**
 推流URL的鉴权时长即将过期(将在过期前1min内发送此回调)

 @param pusher 推流AlivcLivePusher
 @return 新的推流URL
 */
- (NSString *)onPushURLAuthenticationOverdue:(AlivcLivePusher *)
pusher;


/**
 发送SEI Message 通知
```

```
 @param pusher 推流AlivcLivePusher
 */
- (void)onSendSeiMessage:(AlivcLivePusher *)pusher;
```

3. 背景音乐播放回调处理

当收到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。

AlivcLivePusherBGMDelegate回调如下：

```
/**
 背景音乐开始播放

 @param pusher 推流AlivcLivePusher
 */
- (void)onStarted:(AlivcLivePusher *)pusher;

/**
 背景音乐停止播放

 @param pusher 推流AlivcLivePusher
 */
- (void)onStoped:(AlivcLivePusher *)pusher;

/**
 背景音乐暂停播放

 @param pusher 推流AlivcLivePusher
 */
- (void)onPaused:(AlivcLivePusher *)pusher;

/**
 背景音乐恢复播放

 @param pusher 推流AlivcLivePusher
 */
- (void)onResumed:(AlivcLivePusher *)pusher;

/**
 背景音乐当前播放进度

 @param pusher 推流AlivcLivePusher
 @param progress 播放时长
 @param duration 总时长
 */
- (void)onProgress:(AlivcLivePusher *)pusher progress:(long)progress
 duration:(long)duration;

/**
 背景音乐播放完毕

 @param pusher 推流AlivcLivePusher
 */
- (void)onCompleted:(AlivcLivePusher *)pusher;
```

```
/**
 背景音乐开启失败

 @param pusher 推流AlivcLivePusher
 */
- (void)onOpenFailed:(AlivcLivePusher *)pusher;

/**
 背景音乐下载播放超时

 @param pusher 推流AlivcLivePusher
 */
- (void)onDownloadTimeout:(AlivcLivePusher *)pusher;
```

特殊情况处理

1. 网络中断处理

   - 短时间断网和网络切换：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：断网时长在AlivcLivePushConfig设置的重连超时时长和次数范围之外的情况下，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。
   - 针对网络情况提供以下建议说明：
     - 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，此时可以推动给播放端，主播推流中断，播放端在作出相应处理。恢复推流同理。
     - 阿里云播放器重连需要先停止播放在开始播放。调用接口顺序stop > prepareAndPlay。

2. 退后台和锁屏

   - 当app退后台或锁屏时，您可调用 AlivcLivePusher 的 pause/resume接口暂停/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setMute来决定后台时是否采集音频。

结束销毁推流

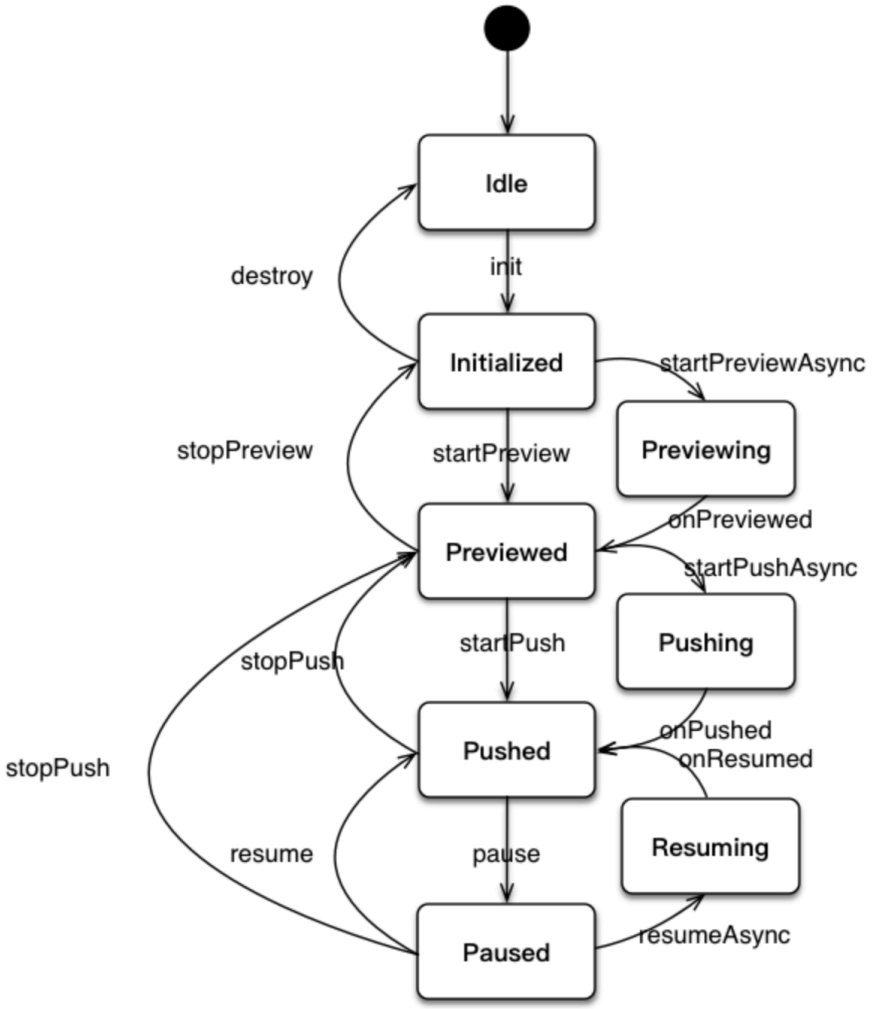
当使用推流结束时需要调用推流相关的销毁接口，清理当前使用资源。

在正常推流模式下，具体示例如下：

```
if(self.livePusher) {
    [self.livePusher stopPush];
    [self.livePusher destory];
    self.livePusher = nil;
 }
```

注意事项

- 接口调用
  - 同步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接口都会在发生错误或者调用顺序不对时 thorws 出异常，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：



- 关于包大小
  - SDK大小为23.9MB。
  - 集成SDK后，ipa包增加大小约为：带播放器版本14MB，不带播放器版本9MB。
  - 适配机型：iPhone5s及以上版本，iOS8.0及以上版本。
- 功能限制说明
  - 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。
- 关于历史版本升级说明
  - 推流SDK V1.3升级至 [V3.0.0-3.3.3]、连麦SDK升级至推流 [V3.0.0-3.3.3]，请下载 升级说明文档。
  - 从V3.3.4版开始不支持推流V1.3版兼容，升级时建议您重新接入最新版SDK。
  - 推流SDK升级至V3.1.0，如果您未集成阿里云播放器SDK，需集成该SDK（已包含推流SDK下载包里面）。
  - 推流SDK升级至V3.2.0+，提供包含阿里云播放器的版本和不包含阿里云播放器两个版本。如果您需要背景音乐功能必须集成播放器，否则可以不使用播放器SDK。

3.5 关于Demo

本文介绍iOS推流SDKDemo相关内容。

Demo架构

Demo使用 MVC 架构。目录结构如下：

各个Controller对应如下

- Others
  - AlivcNavigationController – Navigation基类
  - AlivcRootViewController – 首页列表
  - AlivcCopyrightInfoViewController – 版权信息页
  - AlivcQRCodeViewController – 二维码扫描页

- AlivcLivePusher – 推流SDKv3.0
  - AlivcLivePushConfigViewController – 推流参数设置页
  - AlivcLivePusherViewController – 推流页
- AlivcLiveSessoin – 推流SDKv1.3
  - AlivcLiveConfigViewController – 推流参数设置页
  - AlivcLiveViewController – 推流页

Demo使用

1. 打开SDK Demo工程 `AlivcLivePusherDemo.xcodeproj`。
2. 配置真机调试证书，选择调试真机。
3. 修改 `PrefixHeader.pch` 中的宏 `AlivcTextPushURL` 为您的测试推流地址。

   说明:

   请务必修改推流地址为您的测试推流地址，避免出现多人使用同一推流地址造成推流异常的情况。或者在Demo中通过扫描二维码修改推流地址。
4. 运行，提示Buidling Success。即可在真机环境测试Demo。

播放地址获取

一般情况下，rtmp推流地址格式如下：

```
rtmp://push-#YourCompanyDomainName#/#YourAPPName#/#YourstreamName#
```

对应的播放地址如下：

```
rtmp://pull-#YourCompanyDomainName#/#YourAPPName#/#YourstreamName#
```

Demo描述

- 列表页，列表页可以跳转到推流SDKv3.0版本Demo页和推流SDKv1.3版本Demo页。
- 推流SDKv3.0版本为最新SDK版本，主要使用 `AlivcLivePusher` 以及接口。
- 推流SDKv1.3版本为老接口版本，主要使用 `AlivcLiveSession` 以及相关接口。