阿里云
视频直播

推流SDK

文档版本：20181219

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。
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。	`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。

Demo体验

扫描以下二维码，端云一体化Demo进行体验。

说明：

受微信和QQ限制，请使用钉钉或其他第三方扫码软件扫描安装。

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。

功能	描述
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）
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。

功能	描述
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的集成。

SDK信息

下载后的 **SDK** 目录：

- **AlivcLivePusherSDK** 目录：不带阿里云播放器。
  - **aarlibs** 目录：
  - **jniLibs** 目录：
  - **libs** 目录：
  - **obj** 目录(用于定位底层问题)：
- **AlivcLivePusherSDK+AliyunPlayerSDK** 目录：带阿里云播放器。
  - **aarlibs** 目录：
  - **jniLibs** 目录：
  - **libs** 目录：
  - **obj** 目录(用于定位底层问题)：

> **说明：**
>
> - 集成方式分两种，一种采用aar集成，另一种采用jar包+so集成，选择其一即可。
> - 使用背景音乐功能时必须集成播放器sdk(AlivcPlayer.aar)，采用aar集成方式时需要同时引入播放器SDK的aar；采用jar包+so集成时，可以在application层引入播放器SDK的aar。v3.2.0+添加了调试工具(live-pusher-resources.aar), 用法同AlivcPlayer.aar。AlivcReporter.aar也使用同样的方法引用。

系统要求

- SDK支持Android 4.3及以上版本
- 硬件CPU支持ARM64、ARMV7

运行环境

- SDK编译支持Android 4.3及以上版本
- Android Studio 2.3及以上版本
- APP开发环境与SDK保持兼容即可。
- Android SDK Tools: android-sdk_25.0.0
  - minSdkVersion 18
  - targetSdkVersion 23

SDK集成

下载SDK包，点击 *SDK下载*。

1. Android Studio 新建工程：

2. copy JAR包和SO库：

3. 工程配置：

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

jniLibs 目录为默认so存放位置。如果自己设置位置，需要在app的build.gradle中配置：

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

```
}
}
```

4. 编译工程 Rebuild Project。

2.4 SDK使用

本文介绍Android推流SDK的使用。

SDK的基本使用流程如下：

1. 配置推流参数。
2. 创建推流预览视图。
3. 开始推流。
4. 根据业务需求实时调整推流参数。

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，参见API文档或注释。您可以根据需求修改对应的属性值。如您要在推流过程中实时修改参数，参见AlivcLivePusher提供的属性和方法。

- 基本配置

```
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(AlivcPreviewOrientat
ionEnum.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
mAlivcLivePushConfig.setEnableBitrateControl(true);// 打开码率自适
应，默认为true
```

说明：

- 以上参数配置有默认值，建议采用默认值，即您可以进行简单初始化，不做配置。
- 综合手机性能和网络带宽要求，建议您采用540P（主流移动直播App基本都采用540P）。
- 关闭自适应码率后，码率永远固定在初始码率，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会比较卡顿，请慎用。

- 码控配置

  SDK提供三种推流模式供开发者选择。您可按照下列模式修改AlivcQualityModeEnum参数：

  — **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。

  — **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。

  — **QM_CUSTOM** 自定义模式。SDK会根据开发者设置的码率进行配置。

  清晰度优先或流畅度优先模式，示例代码如下：

```
mAlivcLivePushConfig.setQualityMode(AlivcQualityModeEnum.QM_RESOLUT
ION_FIRST);
```

   说明：

  选择 清晰度优先 或 流畅度优先 模式时，不需设置`targetVideoBitrate`,`minVideoBitrate`,`initialVideoBitrate`。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。

  自定义模式，示例代码如下：

```
mAlivcLivePushConfig.setQualityMode(AlivcQualityModeEnum.QM_CUSTOM);
mAlivcLivePushConfig.setTargetVideoBitrate(1200); //目标码率1200Kbps
mAlivcLivePushConfig.setMinVideoBitrate(400); //最小码率400Kbps
mAlivcLivePushConfig.setInitialVideoBitrate(900); //初始码率900Kbps
```

  设置为自定义模式时，您需自己定义初始码率、最小码率和目标码率。初始码率为开始直播时的码率。最小码率为当网络较差时，码率会逐步减低到最小码率，以减少视频的卡顿。目标码率为当网络较好时，码率会逐步提高到目标码率，以提高视频清晰度。

  自定义模式建议参数：

分辨率	targetVideoBitrate	minVideoBitrate	initialVideoBitrate
360p	800	200	600
480p	1000	300	700
540p	1200	400	900
720p	1500	600	1200

- 分辨率自适应配置

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。动态分辨率可能在某些播放器上不能正常播放，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。

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

说明：

AlivcQualityModeEnum只有在清晰度和流畅度优先时起生效，自定义模式时，分辨率自适应无效。

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。高级美颜需设置美颜模式为 `BEAUTY_Professional`，并且必须添加人脸资源库到开发工程中。

集成美颜功能需要引入美颜库：

1. aar方式引入：live-face-beauty*.aar
2. jar+so方式引入：

标准版美颜：live-beauty.jar + libAliEffectModule.so + liblive-beauty.so;

专业版美颜：live-beauty.jar + libAliEffectModule.so + liblive-beauty.so + live-face.jar + liblive-face.so + libAliFaceAlignmentModule.so;

另外需要在代码中设置处理两个回调：

```
/**
* 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;
  }
});
```

基础美颜无需集成人脸库，示例代码如下：

```
mAlivcLivePushConfig.setBeautyOn(true); // 开启美颜
mAlivcLivePushConfig.setBeautyLevel(AlivcBeautyLevelEnum.BEAUTY_Nor
mal);//设定为基础美颜
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
```

高级美颜必须集成人脸库，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.setBeautyThinFace(40);// 瘦脸设置范围0-100
mAlivcLivePushConfig.setBeautyBigEye(30);// 大眼设置范围0-100
mAlivcLivePushConfig.setBeautyShortenFace(50);// 小脸设置范围0-100
mAlivcLivePushConfig.setBeautyCheekPink(15);// 腮红设置范围0-100
```

 说明：

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

 说明：

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

- 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、int textureWidth、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。

- 图片推流配置

  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：

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

- 水印配置

  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。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:0.1`表示水印的x值为推流画面x轴的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。

- 针对其他android设备（非android手机）的配置

  针对有些android设备（如电视盒子等）增加了如下两项配置：

您可定制化配置摄像头旋转角度：

```
mAlivcLivePushConfig.setPreviewRotation(AlivcPreviewRotationEnum
rotation);//摄像头预览旋转角度
```

有些设备摄像头的连续对焦支持不好，sdk提供传感器感应移动实时对焦：

```
mAlivcLivePushConfig.setFocusBySensor(true);
```

说明：

两个接口，android手机用户不要调用。

- 预览显示模式配置

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

  - 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。

说明：

这三种预览显示模式不影响推流。

推流器的使用

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。

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。

- 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。

- 注册推流回调

推流回调分为三种：Info、Error、Network。Info主要做提示和状态检测使用；Error为错误回调；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 AlivcLivePushErrorLi
stener() {
  @Override
  public void onSystemError(AlivcLivePusher livePusher, AlivcLiveP
ushError 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, AlivcLiveP
ushError error) {
      if(error != null) {
          //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 AlivcLivePushInfoLis
tener() {
  @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 AlivcLivePushNetwork
Listener() {
  @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
  }
});
```

- 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
```

- 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。

- 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();
```

- 美颜实时调整

  推流SDK支持在推流时实时调整美颜参数，根据在参数配置中设置的基础美颜和高级美颜模式，分别调整对应的参数值，示例代码如下：

```
mAlivcLivePusher.setBeautyOn(true); // 实时开启/关闭美颜
/*基础美颜可实时调整以下参数：*/
mAlivcLivePusher.setBeautyWhite(70);// 美白范围0-100
mAlivcLivePusher.setBeautyBuffing(40);// 磨皮范围0-100
mAlivcLivePusher.setBeautyRuddy(40);// 红润设置范围0-100
/* 高级美颜除了支持基础美颜参数调整外，还支持以下参数实时调整： */
mAlivcLivePusher.setBeautyThinFace(40);// 瘦脸设置范围0-100
mAlivcLivePusher.setBeautyBigEye(30);// 大眼设置范围0-100
mAlivcLivePusher.setBeautyShortenFace(50);// 小脸设置范围0-100
mAlivcLivePusher.setBeautyCheekPink(15);// 腮红设置范围0-100
```

- 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。*/
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);
```

- 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*/
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);
```

- 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里提供插入SEI的接口后，在推流状态下，才能调用此接口。示例代码如下：

```
/*
info: 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。
repeat：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。
delay：延时多少毫秒发送。
isKeyFrame：是否只发关键帧。
*/
mAlivcLivePusher.sendMessage("题目信息", 100, 0, false)
```

- 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，第三方设备采集的音视频数据等。具体使用如下：

  1. 在推流配置里面进行外部音视频输入配置，示例如下：

```
/**
* 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，inputStrea
mAudioData等接口。
*/
mAlivcLivePushConfig.setExternMainStream(true,AlivcImageFormat.
IMAGE_FORMAT_YUVNV12, AlivcSoundFormat.SOUND_FORMAT_S16);
```

2. 插入外部视频数据，示例如下：

```
/**
*  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 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(buffer, 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 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 size, long pts, int rotation);
```

3. 插入音频数据，示例如下：

```
/**
* 输入自定义音频数据
* @param data 音频数据 byte array
* @param size
* @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(buffer, 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
* @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
*/
```

```
AlivcLivePusher.inputStreamAudioPtr(long dataPtr, int size, long
pts);
```

- 动态贴纸

  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。

  - 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，上诉参数即表示第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，不需要修改。

  - 添加动态贴纸，示例如下：

    ```
    /**
    * 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);
    ```

  - 删除动态贴纸，示例如下：

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

- 调试工具

SDK提供UI调试工具DebugView，用户问题诊断。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。示例代码如下：

```
AlivcLivePusher.showDebugView(context);//打开调试工具
AlivcLivePusher.hideDebugView(context);//移除调试工具
```

说明：

在您的release版本下， 不要调用添加DebugView的接口。

- 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();
```

录屏功能

- 录屏模式

  录屏有以下三种模式可以选择：

  — 录屏时不开启摄像头：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）。

  1. 在config中设置权限请求的返回数据。
  2. 调用StartCamera传入surfaceView。

— 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。

1. 在config中设置权限请求的返回数据。
2. 调用StartCamera无需传入surfaceView。
3. 调用startCameraMix传入观众端摄像头画面显示位置。

- 设置开启录屏权限

  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。请在推流配置里面进行配置，示例代码如下：

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

- 摄像头预览

  在录屏开启成功后，调用开启/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，stopCamera再startCamera。
  - 如果主播端不需要预览，则surfaceview填为null。

- 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
```

- 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/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。

- 隐私设置

  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。

异常及特殊场景处理

- 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，重新创建或调用`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_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。

  > 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。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：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。
    - 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。

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

说明：

关于播放器参见 阿里云播放器使用说明。

- 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;
- 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。

注意事项

- 检查混淆，确认已将SDK相关包名加入了不混淆名单

```
-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。

- 关于包大小
  - SDK大小：arm64-v8a：11.7M；v7a：9M。
  - 集成SDK后apk会增加约3.6M。
- 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，需注意集成播放器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SDK集成。

SDK信息

您可以在阿里云官网更新，参见 *SDK*下载。

下载后的SDK目录结构如下：

- 依赖播放器版本：包含 `AlivcLibRtmp.framework`、`ALivcLivePusher.framework`两个推流动态库和 `AlivcLibFaceResource.bundle`人脸资源库，同时包含 `AliyunPlayerSDK.framework`播放器动态库和 `AliyunLanguageSource.bundle`播放器资源库。

- 不依赖播放器版本：包含 `AlivcLibRtmp.framework`、`ALivcLivePusher.framework` 两个推流动态库和 `AlivcLibFaceResource.bundle` 人脸资源库。

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。

系统要求

- SDK支持iOS 8.0及以上版本系统
- 硬件CPU支持ARMv7、ARMv7s、ARM64

开发环境

- SDK编译环境Xcode 8.0及以上版本
- Xcode运行环境OS X 10.10 及以上版本

导入**SDK**

- 手动导入

  示例开发环境为Xcode9.0。

  1. 新建SDK测试工程 **Single View App** > **DemoPush** 。
  2. 将`AlivcLibRtmp.framework`和`AlivcLivePusher.framework`两个动态库、`AlivcLibFaceResource.bundle`一个人脸识别资源库以及`AliyunPlayerSDK.framework`和`AliyunLanguageSource.bundle`两个播放器库拖入您的Xcode工程中。

     如果您不需要依赖播放器SDK的版本，则只需要将`AlivcLibRtmp.framework`和`AlivcLivePusher.framework`两个动态库以及`AlivcLibFaceResource.bundle`人脸识别资源库拖入您的工程即可。

     后续将以依赖播放器版本的SDK进行演示。
  3. 按图示勾选 **Copy items if needed**。

4. 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：

5. 在 **Build Phases** > **Copy Bundle Resources** 中添加资源文件依赖。

   > **说明**：
   > 资源文件添加成功后，才能使用人脸识别相关功能。否则调用人脸识别相关接口无效。

6. 在 **Building Setting** > **Enable Bitcode** 修改为 **NO**。

7. 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`。

8. 配置iOS开发证书，选择测试真机，单击 **Run**，提示 **Buidling Success**，SDK添加成功。

- Cocoapods导入

  1. 修改Podfile。

     - 依赖播放器版本

       添加如下语句至您的 **Podfile**文件中。

       ```
       pod 'AlivcLivePusherWithPlayer'
       ```

     - 不依赖播放器版本

       添加如下语句至您的 **Podfile** 文件中。

       ```
       pod 'AlivcLivePusher'
       ```

  2. 执行 **pod install**。

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

  3. 关闭 **bitcode**。

  4. 配置 **Info.plist**。

3.4 SDK使用

本文介绍SDK的基本使用流程。

SDK的基本使用流程如下：

1. 配置推流参数。
2. 创建推流预览视图。
3. 开始推流。
4. 根据业务需求实时调整推流参数。

配置推流参数

您可以使用**AlivcLivePushConfig**配置推流参数，每个属性参数有一个对应的默认值，关于默认值和参数范围，参考API文档或注释。您可以根据需求修改对应的属性值。如您要在推流过程中实时修改参数，参见**AlivcLivePusher**类提供的属性和方法。

- 基本配置

  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：

  - 以上参数配置有有默认值，建议采用默认值。
  - 综合手机性能和网络带宽要求，建议您采用540P（主流移动直播App基本都采用540P）。
  - 关闭自适应码率后，码率永远固定在初始码率，不会在设定的目标码率和最小码率之间自适应调整，码率永远固定在初始码率。如果网络情况不稳定可能会比较卡顿，请慎用。

- 码控配置

SDK提供三种推流模式供开发者选择。修改qualityMode参数，具体值如下：

- 清晰度优先模式（**AlivcLivePushQualityModeResolutionFirst**）：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（**AlivcLivePushQualityModeFluencyFirst**）：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（**AlivcLivePushQualityModeCustom**）：SDK会根据开发者设置的码率进行配置。

清晰度优先或流畅度优先模式，示例代码如下：

```
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。
```

说明：

选择 清晰度优先 或 流畅度优先 模式时，不需设置**targetVideoBitrate**, **minVideoBitrate**, **initialVideoBitrate**。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。

自定义模式下，示例代码如下：

```
config.qualityMode = AlivcLivePushQualityModeCustom//设置为自定义模式
config.targetVideoBitrate = 1200; //  目标码率1200Kbps
config.minVideoBitrate = 400; //  最小码率400Kbps
config.initialVideoBitrate = 900; //  初始码率900Kbps
```

设置为自定义模式时，您需自己定义初始码率、最小码率和目标码率。初始码率为开始直播时的码率。最小码率为当网络较差时，码率会逐步减低到最小码率，以减少视频的卡顿。目标码率为当网络较好时，码率会逐步提高到目标码率，以提高视频清晰度。

自定义模式建议参数：

分辨率	targetVideoBitrate	minVideoBitrate	initialVideoBitrate
360p	800	200	600
480p	1000	300	700
540p	1200	400	900
720p	1500	600	1200

- 分辨率自适应配置

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。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。

```
config.enableAutoResolution = true; // 打开分辨率自适应，默认为false
```

说明：

qualityMode只有在清晰度和流畅度优先时生效。自定义模式时，分辨率自适应无效。

- 美颜配置

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。高级美颜需设置美颜模式为 **AlivcLivePushBeautyModeProfessional**，并且必须添加 **AlivcLibFaceResource.bundle**人脸资源库到开发工程中。

基础美颜需要集成美颜库**AlivcLibBeauty.framework**，必须实现 **AlivcLivePusherCustomFilterDelegate**中的所有代理回调。

使用示例代码如下：

美颜配置代码

```
config.beautyOn = true; // 开启美颜
config.beautyMode = AlivcLivePushBeautyModeNormal；//设定为基础美颜
config.beautyWhite = 70; // 美白范围0-100
config.beautyBuffing = 40; // 磨皮范围0-100
config.beautyRuddy = 40;// 红润设置范围0-100
```

美颜回调处理代码

```
/**
 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];
}
```

```
/**
 通知外置滤镜处理回调
 */
- (int)onProcess:(AlivcLivePusher *)pusher texture:(int)texture
textureWidth:(int)width textureHeight:(int)height extra:(long)extra
{
    return [[AlivcLibBeautyManager shareManager] process:texture
width:width height:height extra:extra];
}
```

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

高级美颜必须集成外置人脸检测库**AlivcLibFace.framework** 和人脸检测资源

包**AlivcLibFaceResource.bundle**，示例代码如下：

高级美颜配置代码

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

人脸检测回调处理代码

```
/**
 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];
}
```

关于使用第三方美颜滤镜和人脸检测滤镜的方法参考以上步骤，只要调用对应的接口即可。

 说明：

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

- 图片推流配置

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：

```
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：

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

- 水印配置

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。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
[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。

- 预览显示模式配置

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。

推流器的使用

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。

- 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。

- 注册推流回调

  推流回调分为三种：Info、Error、Network。Info主要做提示和状态检测使用；Error为错误回调；Network主要为网络相关，注册delegate可接受对应的回调。

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

- 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];
```

- 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。

- 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，您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];
```

- 美颜实时调整

  推流SDK支持在推流时实时调整美颜参数，根据在参数配置中设置的基础美颜和高级美颜模式，分别调整对应的参数值，示例代码如下：

```
[self.livePusher setBeautyOn:true]; // 实时开启/关闭美颜
/*基础美颜可实时调整以下参数：*/
self.livePusher.beautyWhite = 70; // 美白范围0-100
self.livePusher.beautyBuffing = 40; // 磨皮范围0-100
self.livePusher.beautyRuddy = 40;// 红润设置范围0-100
/* 高级美颜除了支持基础美颜参数调整外，还支持以下参数实时调整： */
self.livePusher.beautyBigEye = 30;// 大眼设置范围0-100
self.livePusher.beautyThinFace = 40;// 瘦脸设置范围0-100
self.livePusher.beautyShortenFace = 50;// 小脸设置范围0-100
self.livePusher.beautyCheekPink = 15;// 腮红设置范围0-100
```

- 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];
```

- 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];
```

- 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];
```

- 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，第三方设备采集的音视频数据等。具体使用如下：

  1. 首先在推流配置里面进行外部音视频输入配置，示例如下：

```
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。
```

2. 插入外部视频数据，示例如下：

```
/*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]
/*也可以讲 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;
}
```

3. 插入音频数据，示例如下：

```
/*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];
```

- 动态贴纸

  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。

  — 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，上诉参数即表示第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，不需要修改。

  — 添加动态贴纸，示例如下：

```
/*
waterMarkDirPath：贴纸文件夹路径，必须含json文件
x,y：显示屏幕位置（0~1.0f）
w,h：显示屏幕长宽（0~1.0f）
```

```
return : 返回贴纸的vid编号,删除贴纸时需设置vid。
*/
int vid = [self.livePusher addDynamicWaterMarkImageDataWithPath:
path x:x y:y w:w h:h];
```

- 删除动态贴纸，示例如下：

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

- 调试工具

  SDK提供UI调试工具DebugView，用户问题诊断。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。示例代码如下：

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

  说明：

  在您的release版本下， 不要调用添加DebugView的接口。

- 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 getLivePushStatusInf
o];
/*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)];
```

Replaykit录屏直播使用

Replaykit是iOS9上新加的支持屏幕录制的功能，在iOS10中新增了调用第三方的App扩展来直播屏幕内容。使用阿里云直播SDK配合APP扩展可以实现iOS10以上的录屏直播功能。

- 如何创建直播扩展

  上述ReplayKit的录屏是通过App扩展来实现的。App扩展跟普通的App不同，它不能单独发布，需要内置在一个普通App中，称为容器App。但是扩展的执行与容器App完全独立。扩展由宿主App发起请求来启动，与宿主App进行交互。

在我们的直播Demo中已经实现了支持录屏直播的App扩展AlivcLiveBroadcast和AlivcLiveBroadcastSetupUI。App中具体创建直播扩展如下：

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

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：

配置直播扩展 **Info.plist**，如下图：

此时扩展已经创建成功，运行APP，会自动将直播扩展和APP一同安装到手机，打开一款支持Replaykit直播的APP，比如游戏TowerDash，单击直播按钮，会弹出直播扩展选择，可以看到刚刚创建的扩展。如下图：

- 如何集成直播SDK

  1. 直播扩展中加入**AlivcLivePusher.framework** 和 **AlivcLibRtmp.framework**的依赖。
  2. 修改UI扩展，可以根据需求修改UI扩展的页面原属，配置包含直播地址，分辨率，横竖屏等参数，如下图：
  3. 使用AlivcLivePusher来完成直播相关功能，**SampleHandle**接口中包含了Replaykit录屏直播的所有功能接口，对应的在这些功能接口中调用阿里云直播SDK中**AlivcLivePusher**来实现具体功能。

     - 开始推流

       在**broadcastStartedWithSetupInfo**配置推流参数并实现推流。示例代码如下：

       ```
       - (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *, NSObject *> *)setupInfo {
       self.pushConfig = [[AlivcLivePushConfig alloc] init];
       self.livePusher = [[AlivcLivePusher alloc];
       self.pushConfig.externMainStream = true;
       [self.livePusher initWithConfig:self.pushConfig];
       [self.livePusher startPushWithURL:pushUrl];
       }
       ```

     - 暂停推流

在**broadcastPaused**中实现暂停推流功能。示例代码如下：

```
- (void)broadcastPaused {
[self.livePusher pause];
}
```

- 恢复推流

  在**broadcastResumed**中实现恢复推流功能。示例代码如下：

```
- (void)broadcastResumed {
[self.livePusher resume];
}
```

- 结束推流

  在**broadcastFinished**中实现结束推流功能。示例代码如下：

```
- (void)broadcastFinished {
[self.livePusher stopPush];
[self.livePusher destory];
self.livePusher = nil;
}
```

- 发送音视频数据

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

异常及特殊场景处理

- 当您收到**AlivcLivePusherErrorDelegate**时：

  — 当出现onSystemError系统级错误，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，重新创建或调用`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，建议您重连之前先做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导致的推流中断。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 `startBGMWithMusicPathAsync`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 `startBGMWithMusicPathAsync`重新播放。
- 当网络中断时：

- 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 `AliVcMediaPlayerPlaybackDidFinishNotification` 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:error:` 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 > prepare > play 。示例如下：

```
[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：

关于播放器参见 阿里云播放器使用说明。

- 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），或者，

- 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。

  例如：

```
- (void)addNotifications {
     [[NSNotificationCenter defaultCenter] addObserver:self
                                              selector:@selector(
applicationWillResignActive:)
                                                  name:UIApplicat
ionWillResignActiveNotification
                                                object:nil];
   [[NSNotificationCenter defaultCenter] addObserver:self
                                              selector:@selector(
applicationDidBecomeActive:)
                                                  name:UIApplicat
ionDidBecomeActiveNotification
                                                object:nil];
}
    - (void)applicationWillResignActive:(NSNotification *)notificati
on {
    [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:AVAudioSessionPortOverrideSpea
ker error:nil];
    [session setCategory:AVAudioSessionCategoryPlayAndRecord
withOptions:AVAudioSessionCategoryOptionDefaultToSpeaker|AVAudioSes
sionCategoryOptionAllowBluetooth | AVAudioSessionCategoryOptionMi
xWithOthers 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。

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。

- 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、连麦SDK升级至推流V3.0+，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。

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。