视频点播

SDK使用手册

SDK使用手册

短视频SDK下载

客户端	说明文档	下载地址
iOS	基础版SDK使用说明 标准版SDK使用说明 专业版SDK使用说明	iOS基础版SDK3.3.4， 示例代码 iOS标准版SDK3.3.4， 示例代码 iOS专业版SDK3.3.4， 示例代码
Android	基础版SDK使用说明 标准版SDK使用说明 专业版SDK使用说明	Android基础版3.3.4， 示例代码 Android标准版3.3.4， 示例代码 Android专业版3.3.4， 示例代码

上传SDK下载

客户端	说明文档	下载地址
JavaScript	JavaScript 上传SDK使用说明	JavaScript 上传SDK 1.1.0， 示例代码
iOS	iOS 上传SDK使用说明	iOS 上传SDK 1.1.0， 示例代码
Android	Android 上传SDK使用说明	Android 上传SDK 1.1.0， 示例代码
服务端	**说明文档**	**下载地址**
Java	Java 上传SDK使用说明	Java 上传SDK 1.0.3及示例代码

播放器SDK下载

客户端	说明文档	下载地址
iOS	iOS 基础播放器使用说明 iOS 高级播放器使用说明 iOS UI播放器使用说明	iOS 播放器SDK 3.2.0， iOS 播放器SDK模拟器版本 3.2.0， 示例代码
Android	Android 基础播放器使用说明 Android 高级播放器使用说明 Android UI播放器使用说明	Android播放器SDK 3.2.0， 示例代码
Web播放器	Web播放器使用说明	详见文档引入即可

播放器SDK

Android播放器SDK

日期	版本	修改内容	历史版本
2017-11-15	v3.2.0	1.增加循环播放功能 2.增加截图功能 3.cache内Seek。 4.Demo优化,合并基础播放器和高级播放器	Android播放器SDK 3.2.0， 示例代码
2017-10-30	高级播放器(v3.1.0)	1.增加倍数播放功能 2.增加多语言配置 3.提供音频数据回调接口。 4.优化UI播放器功能，新增部分按钮触发事件	高级播放器SDK 3.1.0, 示例代码
2017-09-30	基础播放器(v3.1.0)	新增播放器场景demo 新增多语言配置 新增倍数播放功能 新增音频数据回调接口	基础播放器SDK3.1.0，播放器Demo
2017-09-19	高级播放器(v3.0.0)	1. 支持边播边缓存； 2. 接口和错误码完善 3. 修复一系列SDK的bug	
2017-08-31	基础播放器(v3.0.0)	重新规范和完善错误码定义 去掉初始化接口中中授权验证AccessKeyCallback参数 新增音量、亮度、获取加载时长等接口 将mts的vid播放接口移动到点播播放器接口	Android 播放器SDK3.0.0，播放器Demo
2017-08-09	基础播放器(V2.4.2)	修复拒绝权限导致的奔溃问题 修复直播奔溃和卡顿问题 优化网络切换导致的无法播放问题 修复Demo奔溃问题	基础播放器SDK2.4.2，播放器Demo
2017-07-31	高级播放器(v1.2.1)	1. 支持离线（加密）视频断点下载； 2. 支持多个视频同时下载 3. 支持PlayAuth方式下载	
2017-06-29	高级播放器(v1.2.0)	1. 支持离线（加密	

		）视频下载； 2. 支持离线播放	
2017-05-28	高级播放器(v1.1.1)	修复部分场景下无法获取视频地址的问题	
2017-04-28	高级播放器(v1.1.0)	增加新的播放参数设置方法，避免泄露风险	
2017-04-22	高级播放器(v1.0.1)	1. 修复1.0版本小问题 2. AliyunVodPlayerView 类添加加了4个方法即 start\pause\isPlaying\getPlayerState	
2017-03-28	高级播放器(v1.0)	满足视频播放的基本功能，含小窗、全屏播放两种模式； 提供标准UI和无UI两种播放器SDK，支持MP4、HLS两种流格式；	
2017-02-27	基础播放器(V2.3.1)	支持mp3格式播放	Android-player-2.3.0
2017-01-18	基础播放器(V2.2.0.20)	修复播放器由于surface传入为空导致的crash 修复缓冲判定bug 修复缓冲时暂停横竖屏切换变形问题 支持初始状态音量设置 完善声音静默接口 支持stop之后prepareAndPlay	Android-player-2.2.0.20
2016-12-25	基础播放器(V2.2.0)	支持带切边的视频渲染模式，支持多实例，支持https	Android-player-2.2.0

iOS播放器SDK

日期	版本	修改内容	历史版本
2017-11-15	v3.2.0	1.增加循环播放功能 2.增加截图功能 3.cache内Seek。 4.Demo优化，合并基础播放器和高级播放器 5.UI播放器增加全屏事件回调	iOS 播放器SDK 3.2.0，iOS 播放器SDK模拟器版本 3.2.0， 示例代码
2017-11-08	高级播放器(v3.1.1)	1.修复UI播放器移动网络情况播放错误的问题	高级播放器SDK 3.1.1，高级播放器SDK模拟器版本 3.1.1， 示例代码

2017-10-30	高级播放器(v3.1.0)	1.增加倍数播放功能 2.提供swift-Demo。 3.增加多语言配置,图片库和语言库为一个bundle。 4.提供音频数据回调接口。 5.优化下载功能。 6.优化UI播放器功能，新增部分按钮触发事件	
2017-10-09	高级播放器(v3.0.1)	1.修复UI播放器4G网络下无法播放问题 2.修复UI播放器下载demo本地地址播放锁屏失效问题。 3.修复UI播放器本地地址播放横屏模式下进度条显示问题。 4.新增可设置初始清晰度，如设置清晰度不存在，首选最高清晰度。	
2017-09-30	基础播放器(v3.1.0)	新增播放器场景demo 新增多语言配置 新增倍数播放功能 新增音频数据回调接口	基础播放器 SDK3.1.0，Demo
2017-09-19	高级播放器(v3.0.0)	1. 支持边播边缓存； 2. 接口和错误码完善 3. 修复一系列SDK的bug	
2017-08-31	基础播放器(v3.0.0)	重新规范和完善错误码定义 去掉授权验证接口setAccessKeyDelegate 新增音量、亮度、调试、打印日志等接口 将mts的vid播放接口移动到点播播放器接口 增加stop结束后的通知消息	基础播放器 SDK3.0.0，Demo
2017-08-09	基础播放器(V2.4.2)	修复频频切换crash和特定视频卡顿问题 修复进度条显示一点点快进效果 修复直播crash问题	基础播放器 SDK2.4.2，Demo
2017-07-31	高级播放器(v1.2.1)	1. 支持离线（加密）视频断点下载； 2. 支持多个视频同时下载 3. 支持PlayAuth方式下载	
2017-06-29	高级播放器(v1.2.0)	1. 支持离线（加密	

		）视频下载； 2. 支持离线播放	
2017-05-18	高级播放器(v1.1.2)	1.修复重播功能，提示信息问题。	
2017-05-10	高级播放器(v1.1.1)	1. 修复蓝牙耳机无声问题。	
2017-04-28	高级播放器(v1.1.0)	1. 支持”vid+ak+apikey “和“vid+播放凭证”两种播放方式； 2. 增加日志统计功能。	
2017-03-28	高级播放器(v1.0)	1. 满足视频播放的基本功能，含小窗、全屏播放两种模式； 2. 提供标准UI和无UI两种播放器SDK，支持MP4、HLS、FLV三种流格式；	
2017-02-27	基础播放器(V2.3.1)	支持mp3格式播放	基础播放器v2.3.1
2016-12-05	基础播放器(V2.2.1)	支持苹果 2017 年 1 月 1 号后强制使用 HTTPS 的要求	iOS-player-2.2.1

web播放器SDK

日期	版本	修改内容
2017-10-31	v2.2.0	1. vid+playauth播放方式使用新接口 ； 2. 增强Flash加密播放； 3. 提供全屏和取消全屏事件； 4. Safari也可以启用hls插件播放；
2017-10-18	v2.1.0	1. 添加language属性，支持国际化； 2. H5支持倍速播放； 3. 支持延迟播放功能；参考 4. 支持截图功能；参考 5. H5支持mp3音频播放； 6. 错误和信息UI可以隐藏自定义； 7. 修复iOS11上的bugs
2017-09-22	v2.0.1	1. H5播放器添加format属性和qualitySort属性；参考 2. H5在PC端支持m3u8的播放参考； 3. H5支持同层播放，解决Android微信上弹全屏问题参考； 4. H5新增

		enableSystemMenu属性控制显示右键； 5. 自适应播放能力的增强； 6. h5清晰度切换的记忆功能 7. 播放过程中或切换地址时地址异常，自动切换到其它可播地址 9. 启用全新的名字Aliplayer 10. 修复一些重要bugs
2017-08-21	v1.9.9	1. H5播放器支持清晰度切换 ； 2. 添加controlBarVisibility属性控制播放面板的显示； 3. 去掉Flash播放器的加密图标； 4. 修复H5播放器loadbyurl的问题； 5. 修复Flash播放器ended事件无效的问题；

短视频SDK

iOS基础版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.适配iPhoneX； 2.新增自定义码率接口； 3.新增GPU裁剪接口，提升性能； 4.内部优化并修复了已知的bug；
2017-10-09	v3.3.3	1.适配iOS11上的特性，解决了导入系统相机高效模式下录制的视频闪退问题； 2.为了更好的完善SDK，对程序的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪界面底色接口； 2.修复已知的一些bug；
2017-08-11	v3.3.1	1.新增导入界面视频和照片显示接口； 2.修复了已知的bug;
2017-07-12	v3.3.0	1.优化了相关的性能；
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题； 3.增加视频数据回调接口，开放给第三方人脸SDK渲染； 4. 底层库提供编码格式错误返回；
2017-05-19	v3.1.1	1. 修复录制界面滤镜切换时美颜图标状态错误； 2. 修复裁剪后的视频再次裁剪

		循环播放首帧短暂黑屏； 3. 修复视频裁剪过程中锁屏，解屏后返回应用界面，界面预览卡屏后出现灰屏； 4. 优化SDK接口
2017-05-12	v3.1.0	1. 满足短视频录制和编辑基本功能； 2. 支持断点拍摄、回删、美颜、摄像头切换、多种分辨率拍摄、实时滤镜、导入裁剪等功能

Android基础版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.新增自定义码率接口； 2.新增GPU裁剪接口，提升性能； 3.内部优化并修复了已知的bug；
2017-09-29	v3.3.3	1.优化了录制时手动对焦功能； 2.为了更好的完善SDK，对程序的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪界面底色接口； 2.新增软编； 3.修复安卓8.0造成的闪退bug;
2017-08-11	v3.3.1	1.新增导入界面视频和照片显示接口； 2.优化视频录制的清晰度；
2017-07-12	v3.3.0	1.修复了录制时偶现的闪退问题； 2.修复了导入视频比例错误的问题； 3.优化了相关的性能;
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题； 调整下载接口中的bundleId为packageName; 4. 底层库提供编码格式错误返回；
2017-05-17	v3.1.1	1. 增加了混淆配置； 2. 修复了gop < 2s崩溃； 3. nexus录制闪退； 4. 裁剪crash； 5. 相册进入拍摄界面无法切换滤、滤镜切换造成美颜图标的状态不正确
2017-05-12	v3.1.0	1. 满足短视频录制和编辑基本功能； 2. 支持断点拍摄、回删、美颜、摄像头切换、多种分辨率拍摄

		、实时滤镜、导入裁剪等功能

iOS标准版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.适配iPhoneX； 2.新增自定义码率接口； 3.新增GPU裁剪接口，提升性能； 4.内部优化并修复了已知的bug；
2017-10-09	v3.3.3	1.适配iOS11上的特性，解决了导入系统相机高效模式下录制的视频闪退问题； 2.为了更好的完善SDK，对程序的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪界面底色接口； 2.修复已知的一些bug；
2017-08-11	v3.3.1	1.修复了已知的bug;
2017-07-21	v3.3.0	1.新增实时混音功能和变速功能； 2.优化了相关的性能;
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题； 3.增加视频数据回调接口，开放给第三方人脸SDK渲染； 4. 底层库提供编码格式错误返回；
2017-05-19	v3.1.1	1. 修复录制界面滤镜切换时美颜图标状态错误； 2. 修复裁剪后的视频再次裁剪循环播放首帧短暂黑屏； 3. 修复视频裁剪过程中锁屏，解屏后返回应用界面，界面预览卡屏后出现灰屏； 4. 优化SDK接口
2017-05-12	v3.1.0	1. 满足短视频录制和编辑基本功能； 2. 支持断点拍摄、回删、美颜、摄像头切换、多种分辨率拍摄、实时滤镜、导入裁剪等功能

Android标准版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.新增自定义码率接口； 2.新增GPU裁剪接口，提升性能； 3.内部优化并修复了已知的

		bug；
2017-09-29	v3.3.3	1.优化了录制时手动对焦功能； 2.为了更好的完善SDK，对程序的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪界面底色接口； 2.新增软编； 3.修复安卓8.0造成的闪退bug;
2017-08-11	v3.3.1	1.优化视频录制的清晰度；
2017-07-21	v3.3.0	1.新增实时混音功能和变速功能； 2.修复了录制时偶现的闪退问题； 3.修复了导入视频比例错误的问题； 4.优化了相关的性能;
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题；调整下载接口中的bundleId为packageName; 4. 底层库提供编码格式错误返回；
2017-05-17	v3.1.1	1. 增加了混淆配置； 2. 修复了gop < 2s崩溃； 3. nexus录制闪退； 4. 裁剪crash； 5. 相册进入拍摄界面无法切换滤、滤镜切换造成美颜图标的状态不正确
2017-05-12	v3.1.0	1. 满足短视频录制和编辑基本功能； 2. 支持断点拍摄、回删、美颜、摄像头切换、多种分辨率拍摄、实时滤镜、导入裁剪等功能

iOS专业版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.适配iPhoneX； 2.新增自定义码率接口； 3.新增GPU裁剪接口，提升性能； 4.内部优化并修复了已知的bug；
2017-10-09	v3.3.3	1.适配iOS11上的特性，解决了导入系统相机高效模式下录制的视频闪退问题； 2.优化了气泡动图文字清晰度； 3.新增视频裁剪后秒进到编辑的接口； 4.为了更好的完善SDK，对程序

		的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪和编辑界面底色接口； 2.修复已知的一些bug；
2017-08-10	v3.3.1	1.新增人脸识别功能； 2.优化了字幕渲染清晰度； 3.解决了播放MV无声音等bug
2017-07-11	v3.3.0	1.实时混音功能和变速功能； 2.新增视频和照片混合导入功能；
2017-06-23	v3.2.0	1.新增视频涂鸦功能； 2.修复MV移动、选择、缩放等动画无效的问题；
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题； 3.增加视频数据回调接口，开放给第三方人脸SDK渲染； 4. 底层库提供编码格式错误返回；

Android专业版SDK

日期	版本	修改内容
2017-10-25	v3.3.4	1.新增自定义码率接口； 2.新增GPU裁剪接口，提升性能； 3.内部优化并修复了已知的bug；
2017-09-29	v3.3.3	1.优化了录制时手动对焦功能； 2.新增视频裁剪后秒进到编辑的接口； 3.为了更好的完善SDK，对程序的闪退，报错等信息进行上报；
2017-08-31	v3.3.2	1.新增裁剪和编辑界面底色接口； 2.新增软编； 3.修复安卓8.0造成的闪退bug; 4.修复导入本地音乐无法播放的bug；
2017-08-11	v3.3.1	1.新增人脸识别功能； 2.优化视频录制的清晰度； 3.优化录制文件过大问题； 4. 修复录制到编辑界面视频画面黑屏卡死bug；
2017-07-11	v3.3.0	1.实时混音功能和变速功能； 2.新增视频和照片混合导入功能；
2017-06-23	v3.2.0	1.新增视频涂鸦功能；

		2.修复不能输出非方形的缩略图； 3.修复MV移动、选择、缩放等动画无效的问题； 4. 提供取非关键帧缩略图接口；
2017-06-16	v3.1.3	1.增加编码格式错误码及提醒； 2.修复裁剪区域不正确的问题； 调整下载接口中的bundleId为packageName 4. 底层库提供编码格式错误返回；

上传SDK

简介

上传SDK提供了视频的文件列表管理和上传控制等功能。其中，文件列表管理包括文件的增加、删除、取消、恢复、遍历、清空；上传控制包括开始、停止、暂停、恢复、设置上传凭证。SDK也提供了回调事件，用来监听上传过程中的状态和进度变化。

上传流程

用户选择文件 -> 添加文件到列表 -> 开始上传 -> 设置上传凭证和地址 -> 上传完成事件。

用户选择文件

用户选择本地要上传的文件。

添加文件到列表

把用户需要上传的所有文件通过addFile接口添加到列表中。

开始上传

调用start接口就会开始真正的上传进程。

设置上传凭证和地址

在文件上传开始事件OnUploadStarted，调用setUploadAuthAndAddress函数，设置点播上传服务

返回的上传凭证和上传地址。注意上传凭证有时效性，如果一次性返回所有文件的上传凭证，要正确处理失效的场景：上传过程中，如果超出了时效范围，需要刷新上传凭证后才能恢复上传。

上传完成事件

包含成功和失败两种事件OnUploadSucceed，OnUploadFailed。

备注：视频点播的上传配置仅要求提供上传凭证（uploadAuth）和上传地址（uploadAddress）即可。

概念和说明

上传地址和上传凭证

上传地址

上传地址是有服务指定的，用户不可修改。

上传凭证

每个上传凭证都绑定了上传地址，不同上传地址关联的上传凭证不能互换，否则无法正常上传文件。另外，上传凭证是有时效性的。

具体参数获取方法查看文档 获取上传凭证和地址 。

分片上传和状态

SDK内部采用的是分片上传机制，状态只在一次执行内有效，如果由于各种原因导致应用退出（例如：关机、关闭浏览器页面、关闭APP、APP异常退出等），需要重新上传。

移动端3G/4G<->Wifi切换

为了避免浪费3G/4G网络下的流量，应用判断切换到3G/4G网络时（需要应用自己实现判断），可以调用pause暂停上传。在切换回Wifi网络时（需要应用自己实现判断），调用resume恢复上传。

现提供了3种终端的SDK：

HTML5：可以集成到PC的浏览器中，开发语言JavaScript。

iOS：可以集成到iOS系统的APP中，开发语言Object-C。

Android：可以集成到Android系统的APP中，开发语言Java。

功能描述

文件列表管理

接口名称	描述
addFile	添加文件到列表中，文件是按照添加的顺序依次上传
deleteFile	从列表中删除文件
cancelFile	取消列表中的单个文件，但是不会从上传列表中删除，效果就是会跳过这个文件的上传（JavaScript版本不支持）
resumeFile	不是恢复上传,只是恢复之前列表中被取消单个文件的状态（JavaScript版本不支持）
listFiles	获取列表
clearFiles	清除列表，即使是上传中的文件，也会停止上传并清除（JavaScript版本不支持）

上传控制

接口名称	描述
start	开始上传
stop	停止上传
pause	暂停上传（JavaScript版本不支持）
resume	恢复上传（JavaScript版本不支持）
resumeUploadWithAuth	上传凭证超时后，使用新的上传凭证来恢复上传
setUploadAuthAndAddress	设置上传地址和上传凭证。每个上传地址和上传凭证是独立的，需要在OnUploadStarted事件中调用，更新当前的上传地址和上传凭证

回调事件

事件名称	描述
OnUploadStarted	每个文件开始上传时都会触发。需要在这里调用setUploadAuthAndAddress设置当前文件的上传地址和上传凭证
OnUploadSucceed	上传成功
OnUploadFailed	上传失败。可恢复型的错误会自动断点续传，例如：网络异常、超时等。不可恢复类型的错误会导致失败，例如：上传凭证错误、文件不存在等

OnUploadProgress	上传进度汇报。在分片上传成功时触发
OnUploadTokenExpired	上传凭证超时。需要从服务重新获取新的上传凭证，并调用resumeUploadWithAuth函数恢复上传
OnUploadRetry	上传过程中，状态由正常切换为异常时触发。例如：网络异常，超时等（JavaScript版本不支持）
OnUploadRetryResume	上传过程中，状态由异常中恢复时触发。（JavaScript版本不支持）

获取上传地址和凭证

简介

以下文档描述的是视频点播上传流程中在服务端（Java）获取上传地址和凭证的方法。

环境要求

Java 6+

Maven

安装

添加maven仓库

```
 <repositories>
<repository>
<id>sonatype-nexus-staging</id>
<name>Sonatype Nexus Staging</name>
<url>https://oss.sonatype.org/service/local/staging/deploy/maven2/</url>
<releases>
<enabled>true</enabled>
</releases>
<snapshots>
<enabled>true</enabled>
</snapshots>
</repository>
</repositories>
```

添加Jar包依赖

注：aliyun-java-sdk-core包版本号必须>=3.2.2

```
 <dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>3.2.2</version>
</dependency>
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-vod</artifactId>
<version>2.6.0</version>
</dependency>
```

接口说明

引用

```
 import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.vod.model.v20170321.CreateUploadVideoRequest;
import com.aliyuncs.vod.model.v20170321.CreateUploadVideoResponse;
import com.aliyuncs.vod.model.v20170321.RefreshUploadVideoRequest;
import com.aliyuncs.vod.model.v20170321.RefreshUploadVideoResponse;
```

初始化

```
   DefaultAcsClient aliyunClient;
aliyunClient = new DefaultAcsClient(
DefaultProfile.getProfile("cn-shanghai",accessKeyId,accessKeySecret));
```

注： accessKeyId和accessKeySecret是全局配置参数，需要按实际值填写。

```
 private static String accessKeyId = "";
private static String accessKeySecret = "";
```

函数

1. 获取视频上传凭证和地址

```
  private static String createUploadVideo(DefaultAcsClient client) {
```

```
CreateUploadVideoRequest request = new CreateUploadVideoRequest();
CreateUploadVideoResponse response = null;
try {
/*必选，视频源文件名称（必须带后缀, 支持 ".3gp", ".asf", ".avi", ".dat", ".dv", ".flv", ".f4v", ".gif", ".m2t",
".m3u8", ".m4v", ".mj2", ".mjpeg", ".mkv", ".mov", ".mp4", ".mpe", ".mpg", ".mpeg", ".mts", ".ogg", ".qt",
".rm", ".rmvb", ".swf", ".ts", ".vob", ".wmv", ".webm"".aac", ".ac3", ".acm", ".amr", ".ape", ".caf", ".flac",
".m4a", ".mp3", ".ra", ".wav", ".wma" ）*/
request.setFileName("源文件名称.mp4");
//必选，视频标题
request.setTitle("视频标题");
//可选，分类ID
request.setCateId(0);
//可选，视频标签，多个用逗号分隔
request.setTags("标签1,标签2");
//可选，视频描述
request.setDescription("视频描述");
response = client.getAcsResponse(request);
} catch (ServerException e) {
System.out.println("CreateUploadVideoRequest Server Exception:");
e.printStackTrace();
return null;
} catch (ClientException e) {
System.out.println("CreateUploadVideoRequest Client Exception:");
e.printStackTrace();
return null;
}
System.out.println("RequestId:"+response.getRequestId());
System.out.println("UploadAuth:"+response.getUploadAuth());
System.out.println("UploadAddress:"+response.getUploadAddress());
return response.getVideoId();
}
```

2. 刷新视频上传凭证

```
  private static void refreshUploadVideo(DefaultAcsClient client, String videoId) {
RefreshUploadVideoRequest request = new RefreshUploadVideoRequest();
RefreshUploadVideoResponse response = null;
try {
request.setVideoId(videoId);
response = client.getAcsResponse(request);
} catch (ServerException e) {
System.out.println("RefreshUploadVideoRequest Server Exception:");
e.printStackTrace();
return;
} catch (ClientException e) {
System.out.println("RefreshUploadVideoRequest Client Exception:");
e.printStackTrace();
return;
}
System.out.println("RequestId:" + response.getRequestId());
System.out.println("UploadAuth:" + response.getUploadAuth());
}
```

函数调用示例

```
  String videoId = createUploadVideo(aliyunClient);
System.out.println("VideoId:" + videoId);
//当网络异常导致文件上传失败时,可刷新上传凭证后再次执行上传操作
refreshUploadVideo(aliyunClient, videoId);
```

简介

以下文档描述的是视频播放流程中在服务端（.NET）获取播放凭证的方法。

环境要求

适用于.NET 2.0 及以上版本

适用于Visual Studio 2010及以上版本

适用于Mono 3.12 及以上版本

安装

GitHub安装

- 如果没有安装git, 请先安装 git
- git clone, 请下载最新的源码 aliyun-net-sdk-core 和 aliyun-net-sdk-vod
- 下载好源码后，按照项目引入方式安装即可

项目引入方式安装

- 如果是下载了SDK包或者从GitHub上下载了源码，希望源码安装，可以右键解决方案，在弹出的菜单中单击添加 > 现有项目
- 在弹出的对话框中选择aliyun-net-sdk-core.csproj 和 aliyun-net-sdk-vod.csproj文件，单击打开
- 接下来右键您的项目 > 引用，选择添加引用，在弹出的对话框选择项目选项卡后选中aliyun-net-sdk-core 和 aliyun-net-sdk-vod项目，单击确定即可

接口说明

引用

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
```

```
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using System;
```

初始化

```
IClientProfile clientProfile = DefaultProfile.GetProfile(regionId, accessKeyId, accessKeySecret);
DefaultAcsClient client = new DefaultAcsClient(clientProfile);
```

注：目前仅支持上海区域，故 regionId 请填写 “cn-shanghai”，accessKeyId 和 accessKeySecret是全局配置参数，需要按实际值填写。

- 函数

获取视频上传凭证和地址

```
private void CreateUploadVideo(DefaultAcsClient client) {
CreateUploadVideoRequest request = new CreateUploadVideoRequest();
request.Title = "视频标题";
request.FileName = "文件名称.mov";
request.Description = "视频描述";
request.CoverURL = "http://cover.sample.com/sample.jpg";
request.Tags = "标签1,标签2";
request.CateId = 0;
try {
CreateUploadVideoResponse response = client.GetAcsResponse(request);
Console.WriteLine("RequestId = " + response.RequestId);
Console.WriteLine("VideoId = " + response.VideoId);
} catch (ServerException e) {
Console.WriteLine(e.ErrorCode);
Console.WriteLine(e.ErrorMessage);
} catch (ClientException e) {
Console.WriteLine(e.ErrorCode);
Console.WriteLine(e.ErrorMessage);
}
}
```

- API调用示例

```
CreateUploadVideo(client);
```

简介

以下文档描述的是视频点播上传流程中在服务端（PHP）获取上传地址和凭证的方法。

环境要求

PHP 5.3+

安装

从Github上下载PHP SDK的源代码，拷贝当前最新的aliyun-php-sdk-core文件夹和aliyun-php-sdk-vod的文件夹到您的项目中（下载地址见：aliyun-openapi-php-sdk），并且放置在同一个目录下。

编辑aliyun-php-sdk-core/Config.php，找到“//config sdk auto load path.”，在这行下面添加:

```
Autoloader::addAutoloadPath("aliyun-php-sdk-vod");
```

接口说明

- 引用

```
include_once 'aliyun-php-sdk-core/Config.php';
use vod\Request\V20170321 as vod;
```

- 初始化

```
include_once 'aliyun-php-sdk-core/Config.php';
$regionId = 'cn-shanghai';
$profile = DefaultProfile::getProfile($regionId, $access_key_id, $access_key_secret);
$client = new DefaultAcsClient($profile);
```

注：accessKeyId和accessKeySecret是全局配置参数，需要按实际值填写，$regionId 为’ cn-shanghai’ ，不需要修改。

- 函数

1. 获取视频上传凭证和地址（create_upload_video）

```
function create_upload_video($client, $regionId) {
$request = new vod\CreateUploadVideoRequest();
//视频源文件标题(必选)
$request->setTitle("视频标题");
//视频源文件名称，必须包含扩展名(必选)
$request->setFileName("文件名称.mov");
//视频源文件字节数(可选)
$request->setFileSize(0);
//视频源文件描述(可选)
$request->setDescription("视频描述");
//自定义视频封面URL地址(可选)
$request->setCoverURL("http://cover.sample.com/sample.jpg");
//上传所在区域IP地址(可选)
```

```
$request->setIP("127.0.0.1");
//视频标签，多个用逗号分隔(可选)
$request->setTags("标签1,标签2");
//视频分类ID(可选)
$request->setCateId(0);
$response = $client->getAcsResponse($request);
return $response;
}
```

2. 刷新视频上传凭证（refresh_upload_video）

```
function refresh_upload_video($client, $regionId) {
$request = new vod\RefreshUploadVideoRequest();
//视频ID(必选)
$request->setVideoId("视频ID");
$response = $client->getAcsResponse($request);
return $response;
}
```

- API调用示例

```
$createResponse = create_upload_video($client, $regionId);
//上传凭证
echo "UploadAuth=".$createResponse->UploadAuth."\n";
//上传地址
echo "UploadAddress=".$createResponse->UploadAddress."\n";
//视频ID
echo "VideoId=".$createResponse->VideoId."\n";
//请求ID
echo "RequestId=".$createResponse->RequestId."\n";

$refreshResponse = refresh_upload_video($client, $regionId);
//视频ID
echo "VideoId=".$refreshResponse->VideoId."\n";
//请求ID
echo "RequestId=".$refreshResponse->RequestId."\n";
```

简介

以下文档描述的是视频点播上传流程中在服务端（Python）获取上传凭证和地址的方法。

环境要求

Python 2.7 + pip

安装

```
pip install aliyun-python-sdk-core
pip install aliyun-python-sdk-vod
```

接口说明

- 引用

```
import json
from aliyunsdkvod.request.v20170321 import CreateUploadVideoRequest
```

- 初始化

```
from aliyunsdkcore import client
clt = client.AcsClient(access_key_id, access_key_secret, 'cn-shanghai')
```

注：accessKeyId和accessKeySecret是全局配置参数，需要按实际值填写。

- 函数

1. 创建视频上传任务（create_upload_video）

```
def create_upload_video(clt):
request = CreateUploadVideoRequest.CreateUploadVideoRequest()
request.set_accept_format('JSON')
request.set_Title("test_upload") //视频标题
request.set_FileName("test_upload.mov") //视频源文件名称
request.set_FileSize(10***L) //视频文件大小
request.set_Description("test_decription") //视频描述
request.set_CoverURL("test_url") //自定义视频封面URL地址
request.set_Privilege("public") //视频观看权限
request.set_IP("127.0.0.1") //上传所在IP地址
response = json.loads(client.do_action(request))
return response
```

2. 刷新视频上传凭证（refresh_upload_token）

```
def refresh_upload_token(clt):
request = RefreshUploadTokenRequest.RefreshUploadTokenRequest()
request.set_accept_format('JSON')
request.set_VideoId("test_vid") //视频ID
response = json.loads(client.do_action(request))
return response
```

- API调用示例

```
get_video_playinfo(clt)
```

```
create_upload_video(clt)
refresh_upload_token(clt)
```

安装

在页面上引入下面两个JS脚本，见 视频上传SDK下载 。

```
<script src="aliyun-sdk.min.js"></script>
<script src="vod-sdk-upload-1.1.0.min.js"></script>
```

创建VODUpload实例

在这里需要设置回调函数。

```
var uploader = new VODUpload({
// 开始上传
'onUploadstarted': function (uploadInfo) {
log("onUploadStarted:" + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:" +
uploadInfo.bucket + ", object:" + uploadInfo.object);
// uploader.setUploadAuthAndAddress(uploadInfo, uploadAuth, uploadAddress);
}
// 文件上传成功
'onUploadSucceed': function (uploadInfo) {
log("onUploadSucceed: " + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:" +
uploadInfo.bucket + ", object:" + uploadInfo.object);
},
// 文件上传失败
'onUploadFailed': function (uploadInfo, code, message) {
log("onUploadFailed: file:" + uploadInfo.file.name + ",code:" + code + ", message:" + message);
},
// 文件上传进度，单位：字节
'onUploadProgress': function (uploadInfo, totalSize, uploadedSize) {
log("onUploadProgress:file:" + uploadInfo.file.name + ", fileSize:" + totalSize + ", percent:" +
Math.ceil(uploadedSize * 100 / totalSize) + "%");
},
// 上传凭证超时
'onUploadTokenExpired': function () {
console.log("onUploadTokenExpired");
// uploader.resumeUploadWithAuth(uploadAuth);
}
});
```

列表管理

添加上传文件

注意：支持的文件大小<=10G。

需要使用标准的input方式让用户选择文件。

```
<form action="">
<input type="file" name="file" id="files" multiple/>
</form>
userData = '';
document.getElementById("files")
.addEventListener('change', function (event) {
for(var i=0; i<event.target.files.length; i++) {
// 逻辑代码
}
});
```

获取到用户选择的文件后，添加到上传列表中。

```
uploader.addFile(event.target.files[i], null, null, null, userData);
```

上传时，可以选择是否启用水印和优先级，userData是一个json对象。第一级的Vod是必须的，Vod下面有2个属性，示例如下：

```
var userData = '{"Vod":{"UserData":"{"IsShowWaterMark":"false","Priority":"7"}"}}';
```

删除上传文件

index，对应listFiles接口返回列表中元素的索引。

```
uploader.deleteFile(index);
```

取消单个文件上传

```
uploader.cancelFile(index);
```

恢复单个文件上传

```
uploader.resumeFile(index);
```

获取上传文件列表

```
uploader.listFiles();
```

```
var list = uploader.listFiles();
for (var i=0; i<list.length; i++) {
log("file:" + list[i].file.name + ", status:" + list[i].state + ", endpoint:" + list[i].endpoint + ", bucket:"
+ list[i].bucket + ", object:" + list[i].object);
}
```

清理上传文件列表

```
uploader.cleanList();
```

上传控制

开始上传

```
uploader.startUpload();
```

停止上传

```
uploader.stopUpload();
```

上传凭证失效后恢复上传

```
uploader.resumeUploadWithAuth(uploadAuth);
```

设置上传地址和上传凭证

```
uploader.setUploadAuthAndAddress(uploadAuth, uploadAddress);
```

环境要求

iOS系统版本：iOS 7.0以上

安装

pod方式引入Framework

（1）在Podfile文件中添加VODUpload库依赖

```
pod 'VODUpload'
```

（2）更新pod repo

```
pod repo update
```

(3) 安装VODUpload库

```
pod install
```

直接引入Framework

（1）VODUpload iOS SDK framework，见 视频上传SDK下载 。

（2）AliyunVideoCore iOS SDK framework, 见 视频上传SDK下载 。

（3）OSS iOS SDK framework，具体SDK说明和下载地址请查看 OSS产品的iOS-SDK 。

在Xcode中，直接把framework拖入您对应的Target下即可，在弹出框勾选Copy items if needed。

工程中引入头文件

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

注意，引入Framework后，需要在工程Build Settings的Other Linker Flags中加入-ObjC。如果工程此前已经设置过-force_load选项，那么，需要加入-force_load <framework path>/AliyunOSSiOS。

兼容IPv6-Only网络

OSS移动端SDK为了解决无线网络下域名解析容易遭到劫持的问题，已经引入了HTTPDNS进行域名解析，直接使用IP请求OSS服务端。在IPv6-Only的网络下，可能会遇到兼容性问题。而APP官方近期发布了关于IPv6-only网络环境兼容的APP审核要求，为此，SDK从2.5.0版本开始已经做了兼容性处理。在新版本中，除了-ObjC的设置，还需要引入两个系统库

```
libresolv.tbd
SystemConfiguration.framework
```

创建VODUpload实例

在这里需要设置回调函数。

注：在onUploadstarted回调中需设置好上传凭证和上传地址才可以正常上传。

```
 OnUploadStartedListener testUploadStartedCallbackFunc = ^(UploadFileInfo* fileInfo) {
NSLog(@"upload started .");
};

OnUploadSucceedListener testSuccessCallbackFunc = ^(NSString* filePath){
NSLog(@"file:%@ upload success!", filePath);
};

OnUploadFailedListener testFailedCallbackFunc = ^(NSString* filePath, NSString* code, NSString*
message){
NSLog(@"failed code = %@, error message = %@", code, message);
};

// 单位：字节
OnUploadProgressListener testProgressCallbackFunc = ^(NSString* filePath, long uploadedSize, long
totalSize) {
NSLog(@"progress uploadedSize : %li, totalSize : %li", uploadedSize, totalSize);
};

OnUploadTokenExpiredListener testTokenExpiredCallbackFunc = ^{
NSLog(@"*token expired.");
// get token and call resmeUploadWithAuth.
};

OnUploadRertyListener testUploadRertyListener = ^{
NSLog(@"retry begin.");
};

OnUploadRertyResumeListener testUploadRertyResumeListener = ^{
NSLog(@"retry resume.");
};

VODUploadListener *listener;
listener = [[VODUploadListener alloc] init];
listener.started = testUploadStartedCallbackFunc;
listener.success = testSuccessCallbackFunc;
listener.failure = testFailedCallbackFunc;
listener.progress = testProgressCallbackFunc;
listener.expire = testTokenExpiredCallbackFunc;
listener.retry = testUploadRertyListener;
listener.retryResume = testUploadRertyResumeListener;
```

列表管理

添加上传文件

注意：支持的文件大小<=4G。

```
[uploader addFile:<uploadFilePath>
vodInfo:<vodInfo>];
```

上传时，可以使用VodInfo对象来设置是否启用水印和优先级。定义如下：

```
@interface VodInfo : NSObject

@property (nonatomic, assign) BOOL isShowWaterMark;
@property (nonatomic, assign) NSNumber* priority;
```

删除上传文件

```
[uploader deleteFile:<index>];
```

取消列表中的单个文件上传

```
[uploader cancelFile:<index>];
```

恢复列表中的单个文件上传

```
[uploader resumeFile:<index>];
```

获取上传文件列表

```
[uploader listFiles];
```

清理上传文件列表

```
[uploader clearFiles];
```

上传控制

开始上传

```
[uploader start];
```

停止上传

```
[uploader stop];
```

暂停上传

```
[uploader pause];
```

恢复上传

```
[uploader resume];
```

上传凭证失效后恢复上传

```
[uploader resumeUploadWithAuth:<uploadAuth>];
```

设置上传地址和上传凭证

```
 [uploader setUploadAuthAndAddress:uploadAuth
uploadAddress:uploadAddress];
```

环境要求

Android系统版本：2.3.3（Api 10）及以上

通过jar包的形式导入. 分别引入以下两个jar包，

（1）VODUpload Android SDK，见 视频上传SDK下载 。

（2）OSS Android SDK ，具体SDK说明和下载地址请查看 OSS产品的Android-SDK 。 SDK下载之后，进行以下步骤（对Android studio或者Eclipse都适用）：

- 解压后在libs目录下得到jar包，目前包括aliyun-vod-croe-android-sdk-xxx.jar、aliyun-vod-upload-android-sdk-xxx.jar、gson-xxx.jar、jsr305-xxx.jar、okhttp-xxx.jar、okio-xxx.jar、oss-android-sdk-xxx.jar，将以上7个jar包导入工程的libs目录。

权限设置

以下是VODUpload Android SDK所需要的Android权限，请确保您的AndroidManifest.xml文件中

已经配置了这些权限，否则，SDK将无法正常工作。

注意：Android 6.0之后读写权限需要用户动态申请.请保证申请权限之后再读写文件不然会出现上传失败，文件找不到的异常.

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

创建VODUpload实例

在这里需要设置回调函数。

```
VODUploadCallback callback = new VODUploadCallback() {
/**
* 文件开始上传时触发
*/
void onUploadStarted() {;}

/**
* 上传成功回调
*/
void onUploadSucceed(UploadFileInfo info) {;}

/**
* 上传失败
*/
void onUploadFailed(UploadFileInfo info, String code, String message) {;}

/**
* 回调上传进度
* @param uploadedSize 已上传字节数
* @param totalSize 总共需要上传字节数
*/
void onUploadProgress(UploadFileInfo info, long uploadedSize, long totalSize) {;}

/**
* 上传凭证过期后，会回调这个接口
* 可在这个回调中获取新的上传，然后调用resumeUploadWithAuth继续上传
*/
void onUploadTokenExpired() {;}

/**
* 上传过程中，状态由正常切换为异常时触发
*/
void onUploadRetry(String code, String message) {;}

/**
* 上传过程中，从异常中恢复时触发
*/
```

```
void onUploadRetryResume() {;}
};

VODUploadClient uploader = new VODUploadClientImpl(getContext());
```

列表管理

添加上传文件

注意：支持的文件大小<=4G。

```
uploader.addFile("<uploadFilePath>",
"<videoInfo>");
```

上传时，可以使用VodInfo对象来设置是否启用水印和优先级。定义如下：

```
private Boolean isShowWaterMark;
private Integer priority;
```

删除上传文件

index，对应listFiles接口返回列表中元素的索引。

```
uploader.deleteFile(index);
```

取消列表中的单个文件上传

```
uploader.cancelFile(index);
```

恢复列表中的单个文件上传

```
uploader.resumeFile(index);
```

获取上传文件列表

```
List<UploadFileInfo> list = uploader.listFiles();
```

清除上传文件列表

```
upload.clearFiles();
```

上传控制

开始上传

```
uploader.start();
```

停止上传

```
uploader.stop();
```

暂停上传

```
uploader.pause();
```

恢复上传

```
uploader.resume();
```

上传凭证失效后恢复上传

```
uploader.resumeUploadWithAuth(uploadAuth);
```

设置上传地址和上传凭证

```
uploader.setUploadAuthAndAddress(uploadAuth, uploadAddress);
```

环境要求

Java 6+

安装

以 1.0.3 版本为例，步骤如下:

1.下载Java示例代码VODUploadDemo-java-1.0.3.zip开发包(包含示例代码和所需jar包），见 视频上传SDK下载

2.将解压后lib目录下的所有jar文件拷贝至您的项目中;

3.在Eclipse中选择您的工程，右击 -> Properties -> Java Build Path -> Add JARs;

4.选中您在第一步拷贝的所有jar文件;

经过以上几步，您就可以在Eclipse项目中使用VODUpload Java SDK。

示例程序

将VODUploadDemo-java-1.0.3.zip开发包解压后, 在sample目录下的UploadVideoDemo.java为示例程序, 如下：

```
import com.aliyun.vod.upload.impl.UploadVideoImpl;
import com.aliyun.vod.upload.req.UploadVideoRequest;
import com.aliyun.vod.upload.resp.UploadVideoResponse;

public class UploadVideoDemo {
public static void main(String[] args) {
//帐号AKID(必选)
String accessKeyId = "accessKeyId";
//帐号AK密钥(必选)
String accessKeySecret = "accessKeySecret";
//指定上传文件绝对路径,文件名称必须包含扩展名(必选)
String fileName = "/*/*/文件名称.mp4";
//视频标题(必选)
String title = "视频标题";
UploadVideoRequest request = new UploadVideoRequest(accessKeyId, accessKeySecret, title, fileName);
//视频分类ID(可选)
request.setCateId(0);
//视频标签,多个用逗号分隔(可选)
request.setTags("标签1,标签2"); //视频描述(可选)
request.setDescription("视频描述"); //视频自定义封面URL(可选)
request.setCoverURL("http://cover.sample.com/sample.jpg");
//设置上传完成后的回调URL(可选)
request.setCallback("http://callback.sample.com"); //可指定分片上传时每个分片的大小，默认为10M字节
request.setPartSize(10 * 1024 * 1024L);
//可指定分片上传时的并发线程数，默认为1，(注：该配置会占用服务器CPU资源，需根据服务器情况指定）
request.setTaskNum(1);
//是否使用默认水印
request.setIsShowWaterMark(true);

try {
UploadVideoImpl uploader = new UploadVideoImpl();
UploadVideoResponse response = uploader.uploadVideo(request);
//上传成功后返回视频ID
System.out.print("VideoId=" + response.getVideoId());
} catch (Exception e) {
```

```
e.printStackTrace();
System.out.print(e.getCause());
System.out.print(e.getMessage());
return;
}
}
}
```

播放器SDK

Android使用说明

产品介绍

阿里云播放器SDK（ApsaraVideo for Player SDK）是阿里视频云端到云到端服务的重要一环，除了支持点播和直播的基础播放功能外，深度融合视频云业务，如支持视频的加密播放、安全下载、清晰度切换等业务场景，为用户提供简单、快速、安全、稳定的视频播放服务。

特色功能

功能	功能描述
倍数播放	支持0.5~2倍的实时变速功能，实现在变速的情况下声音变速不变调。
加密播放	支持在云端转码为加密流，加密流仅能通过播放器SDK解密，保证视频安全。
安全下载	支持对下载的视频进行二次加密，保证被下载的视频仅能通过唯一的应用播放，高级别的防盗措施。
视频缓存	提供视频边播边缓存功能，满足短视频场景下的循环播放，节约用户流量。
视频截图	支持截取播放画面的任意帧，可用户视频封面选取或分享精彩画面给朋友。

核心优势

1.简单、易集成

Android和iOS提供统一接口和错误码，接近系统API的接口设计保证每位开发者能快速集成。

2.分层架构设计

基础功能、业务功能、UI组件等分层架构，保证最精简的包大小，根据业务需求组合选用。

3.云和端一体化

云端加密、客户端解密，端和云的联动保证视频安全。端上采集数据、云端分析，为业务运营提供支持。

4.多层安全保护

从防盗链、URL鉴权到加密播放和安全下载，全方位保护视频安全，满足不同场景的安全需求。

功能对比

阿里云播放器SDK提供基础播放器、高级播放器和UI播放器三层框架满足不同用户、不同业务场景需求，开发者可根据自己的业务需求来选用。具体区别如下：

基础播放器（AliyunPlayer/AlivcPlayer）：提供播放视频的基础能力，仅支持URL的方式播放，建议使用阿里云CDN+OSS存储或使用其他第三方服务的用户使用。

高级播放器（AliyunVodPlayer）：提供播放视频的高级能力，如视频加密、安全下载、边播边下缓存功能等，建议使用阿里视频云点播和直播业务的用户使用（最推荐）。

UI播放器（AliyunVodPlayerView）：提供多套播放器皮肤，建议对播放器个性化要求比较低并且想最快速度实现播放功能的用户使用。

> 备注：以上三个播放器SDK层层依赖，UI播放器依赖高级播放器，高级播放器依赖基础播放器。因此使用UI播放器的用户需同时集成基础播放器和高级播放器SDK，使用高级播放器的用户需同时集成基础播放器SDK。反之，使用基础播放器的用户可以不需要集成高级播放器和UI播放器SDK。

功能点	功能说明	基础播放器	高级播放器	UI播放器
完整UI	SDK包含多套完整UI，用户可以根据自己的应用风格选用	×	×	√
播放控制	支持开始、结束、暂停、恢复、重播和循环播放等播放控制功能	√	√	√
填充模式	支持画面填充和画面裁剪两种填充模式	√	√	√
静音	支持开启和关闭静音功能	√	√	√
音量调节	支持实时调节系统音量（UI版支持手势）	√	√	√

亮度调节	支持系统的亮度调节（UI版支持手势）	√	√	√
纯音频播放	支持AAC编码的MP3音频文件播放	√	√	√
多实例	支持在一个界面添加多个播放器同时播放	√	√	√
点播/直播支持	可以同时支持点播和直播功能	√	√	√
URL播放	支持本地视频和网络视频的URL方式播放	√	√	√
vid播放	支持点播提供的vid方式播放	×	√	√
自动播放	支持视频prepare后自动播放	√	√	√
Seek	支持seek到指定位置（UI版支持手势）	√	√	√
锁屏	支持锁屏功能，包含锁定旋转和隐藏界面元素	×	×	√
清晰度切换	支持点播和转码的多路清晰度流切换	×	√	√
加密播放	支持点播转码的加密流播放	×	√	√
安全下载	支持通过唯一应用下载视频并进行加密	√	√	√
边播边下缓存	支持视频边播边缓存功能（当前仅支持vid播放方式），适合短视频的循环播放场景	×	√	√
倍数播放	支持0.5~2倍的变速播放，支持音频变速不变调	√	√	√
后台播放	支持界面切到后台后继续播放音频	√	√	√
首屏秒开	支持点播和直播的首屏秒开功能	√	√	√

动态追帧	支持直播的动态追帧，降低延时	√	√	√
自动重连	支持直播的自动重连功能	√	√	√
视频截图	支持截取播放画面的任意一帧	√	√	√
cache内seek	支撑已经缓冲的视频内容在seek时不清除缓冲内容并快速seek	√	√	√

使用场景

场景一：短视频列表滑动及循环播放

【场景描述】在短视频应用中，往往采用全屏滑动播放的方式展示精彩内容，阿里云播放器SDK提供视图大小自定义功能可以简单实现全屏播放的需求，同时提供多实例、自动播放和预加载能力，可轻松实现多视频全屏滑动播放功能。由于视频内容短而精，短视频应用通常采用循环播放的方式让用户反复观看，为了节约用户流量和无缝的循环播放，阿里云播放器SDK提供了边播边缓存和循环播放接口，只需简单设置即可满足应用场景。

【使用流程】使用阿里云短视频SDK录制视频，然后使用上传SDK上传至点播系统，最后使用阿里云播放器SDK的高级框架，开启边播边缓存和循环播放功能实现短视频列表的全屏滑动播放和循环播放功能。

场景二：视频版权保护

【场景描述】现在对于视频版权的保护意识和要求越来越高，例如用户要做一个教育类的视频网站，由教师提供视频课程，只有购买课程的用户才能观看，那么如何保护视频不被盗播和盗版？阿里云播放器SDK提供多层级保护：一、提供防盗链功能仅允许配置了白名单的用户访问；二、提供URL鉴权功能，保护视频仅能在鉴权有效期内播放器；三、提供加密流播放功能，保障视频仅能使用阿里云播放器SDK才能播放；四、提供安全下载功能，保证下载的视频仅能通过在控制台配置的唯一应用（bundleID或签名）播放。

【使用流程】首先在点播控制台安全设置中开启防盗链和URL鉴权功能，然后在转码设置中配置的转码流中包含加密流，再在下载设置中开启安全下载并生成加密文件，最后把加密文件集成到播放器SDK，使用播放器SDK的高级框架进行播放和下载。

场景三：数据化运营

【场景描述】数据是反应用户和产品最真实情况的一种方式，通过数据分析可以优化产品、指导业务发展、提升转化率、为决策提供依据。阿里云播放器采用最接近用户的使用端收集数据，实时反馈用户的真实使用情况。通过数据隔离和保密措施保证每个用户的数据仅为产生数据的用户服务。

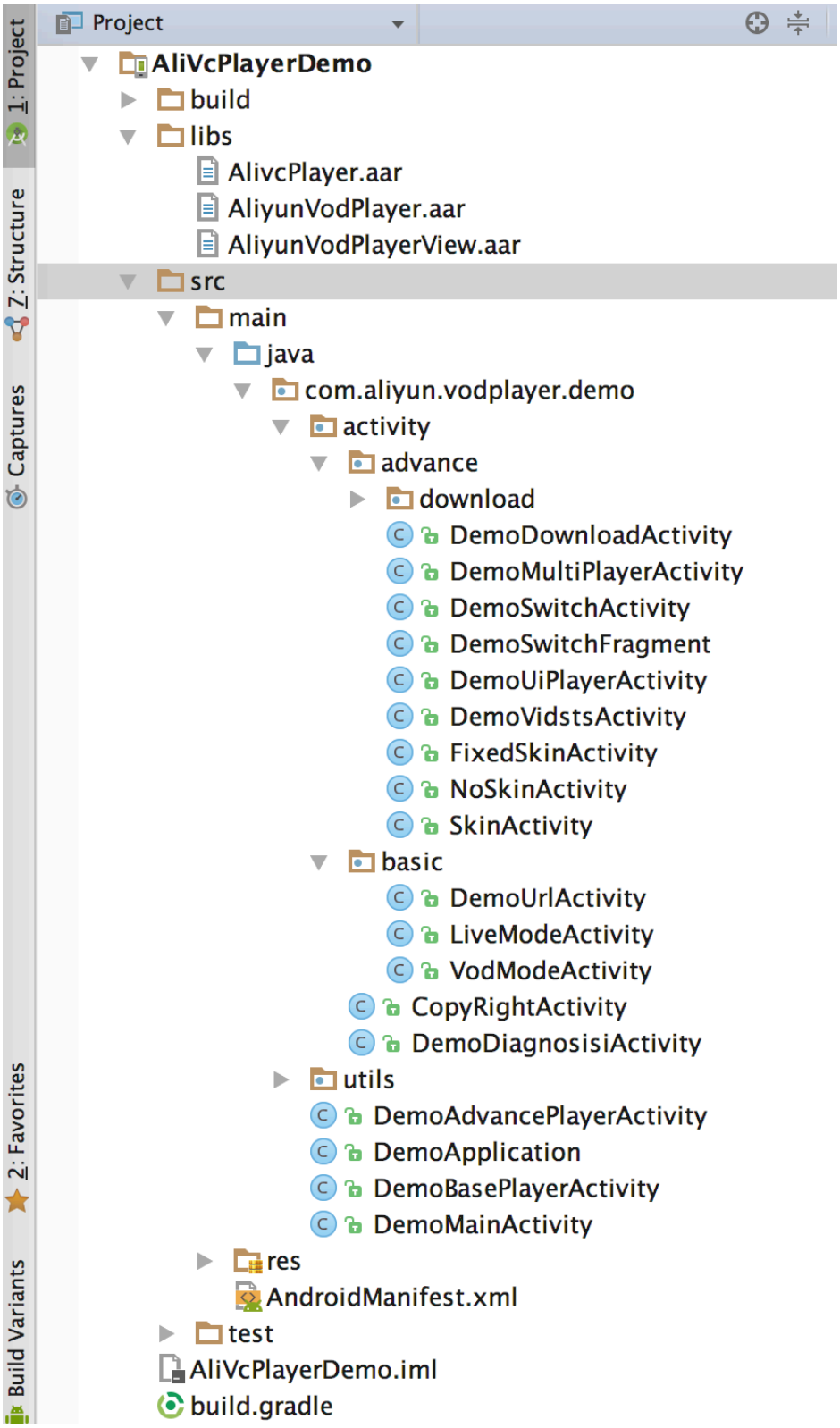
【使用流程】使用阿里云播放器SDK的高级框架或UI框架播放已经上传至点播系统中的视频，在点播控制台的数据分析中查看播放器数据和Top数据。

导入并运行

1. 在Android Studio环境中，使用File->Open，然后选中demo工程所在路径；

2. 导入工程后，运行demo，页面输入vid和对应的ak，token等信息，即可进行播放；

代码结构



Demo界面说明：

- DemoApplication：播放器配置页面。

- DemoMainActivity：选择播放方式的页面。
- DemoBasePlayerActivity：基础播放功能选择界面。
- DemoAdvancePlayerActivity：高级播放功能选择界面。
- DemoDiagnosisiActivity：网络诊断页面。
- CopyRightActivity:版权声明页面。

基础播放器部分：

- DemoUrlActivity: 基本播放器输入地址页面。
- VodModeAcitivity:基本播放器-点播播放页面
- LiveModeAcitivity:基本播放器-直播播放页面

高级播放器部分：

- SkinActivity：有皮肤播放页面，使用默认皮肤进行播放的示例。
- NoSkinActivity：无皮肤播放页面，提供了简单的播放操作和信息展示。
- FixedSkinActivity：有皮肤播放页面，固定竖屏播放。
- DemoVidStsActivity：填写VidSts的界面，使用无皮肤播放器播放。
- DemoUiPlayerActivity：填写VidSts的界面，使用有皮肤播放器播放。
- DemoSwitchAcitivity:切换播放器播放实例。
- DemoMultiPlayerActivity: 多播放器播放页面。
- DemoDownloadActivity: 下载功能的示例。

播放器SDK使用

1.概述

出于安全性的考虑，我们点播支持对视频进行加密。视频加密之后使用我们的播放器进行解密,于是就需要一个加解密的流程，于是播放器下载就需要先拿到一个安全文件.以下文档将说明怎么样拿到这个安全文件.

2.文档适用人群

对于客户端播放器实现下载需要的”安全图片“的开发者.

3.开启安全下载并生成安全文件

- 1.保证开启点播服务之后，如何开启阿里云点播服务见：

https://help.aliyun.com/document_detail/51512.html?spm=5176.doc51236.6.556.3OyQ7H

- 2.在阿里云控制台选择云计算基础服务—->视频点播——>全局设置——->下载设置—->开启下载功能——>选择安全下载
- 3.需要提供两个值来获取密钥: app唯一标识如何获取？
    - 安卓：需要提供一个sha1的密钥.如果觉得使用命令行比较麻烦这边提供一段代码段直接生成这个密钥.

```
//这个是获取SHA1的方法
public static String getCertificateSHA1Fingerprint(Context context) {
//获取包管理器
PackageManager pm = context.getPackageManager();
//获取当前要获取SHA1值的包名，也可以用其他的包名，但需要注意，
//在用其他包名的前提是，此方法传递的参数Context应该是对应包的上下文。
String packageName = context.getPackageName();
//返回包括在包中的签名信息
int flags = PackageManager.GET_SIGNATURES;
PackageInfo packageInfo = null;
try {
//获得包的所有内容信息类
packageInfo = pm.getPackageInfo(packageName, flags);
} catch (PackageManager.NameNotFoundException e) {
e.printStackTrace();
}
//签名信息
Signature[] signatures = packageInfo.signatures;
byte[] cert = signatures[0].toByteArray();
//将签名转换为字节数组流
InputStream input = new ByteArrayInputStream(cert);
//证书工厂类，这个类实现了出厂合格证算法的功能
CertificateFactory cf = null;
try {
cf = CertificateFactory.getInstance("X509");
} catch (CertificateException e) {
e.printStackTrace();
}
//X509证书，X.509是一种非常通用的证书格式
X509Certificate c = null;
try {
c = (X509Certificate) cf.generateCertificate(input);
} catch (CertificateException e) {
e.printStackTrace();
}
String hexString = null;
try {
//加密算法的类，这里的参数可以使MD4,MD5等加密算法
MessageDigest md = MessageDigest.getInstance("SHA1");
//获得公钥
byte[] publicKey = md.digest(c.getEncoded());
//字节到十六进制的格式转换
hexString = byte2HexFormatted(publicKey);
} catch (NoSuchAlgorithmException e1) {
e1.printStackTrace();
} catch (CertificateEncodingException e) {
```

```
e.printStackTrace();
}
return hexString;
}
//这里是将获取到得编码进行16进制转换
private static String byte2HexFormatted(byte[] arr) {
StringBuilder str = new StringBuilder(arr.length * 2);
for (int i = 0; i < arr.length; i++) {
String h = Integer.toHexString(arr[i]);
int l = h.length();
if (l == 1)
h = "0" + h;
if (l > 2)
h = h.substring(l - 2, l);
str.append(h.toUpperCase());
if (i < (arr.length - 1))
str.append(':');
}
return str.toString();
}
```

- iOS：直接填bundleID即可

- 4.需要提供两个值来获取密钥: 离线解密私钥怎么填写？填写16-32位的数字即可，其他没有限制.
- 5.填写完成之后点击“生成密钥并并下载” 这样就拿到了你想要的安全图片了.

注意:提供获取sha1的算法的前提是获取的默认的keystore的签名的sha1. 如果你最终release包是另外的keystore,建议直接在gradle文件中直接配置release keystore然后通过上面获取sha1的方法获取.

```
    //这里演示如何配置keystore,下面的keystore文件开发者可以自由替换, 一下配置的前提是开发者将keystore文件放在项目的根目录下.
signingConfigs {
debug {
storeFile file("$rootDir/debug.keystore")
storePassword "android"
keyAlias "androiddebugkey"
keyPassword "android"
}

release {
storeFile file("$rootDir/debug.keystore")
storePassword "android"
keyAlias "androiddebugkey"
keyPassword "android"
}
}

buildTypes {

debug {
multiDexEnabled true
signingConfig signingConfigs.debug
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
```

```
release {
minifyEnabled true
multiDexEnabled true
signingConfig signingConfigs.release
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
```

一、概述

阿里云视频点播总是会出现这几个值VideoId、AccessKeyId、AccessKeySecret、playKey(apikey)、playauth，这些值到底是什么？为什么会需要这些值？这些值到底从哪里拿到？有什么区别呢？本文将试图将这件事情讲清楚.并推荐用户使用正确的模式来完成点播的上传、下载、播放.

二、获取videoID

2.1 videoID是什么？为什么需要videoID?

上传点播之后的视频ID.主要还是安全性考虑，用户拿到的都是一个个的视频ID，而不是视频URL，避免被爬数据.
（当然也可以拿到视频URL，通过点播open api获取：
https://help.aliyun.com/document_detail/56124.html?spm=5176.doc54832.6.626.ej5DVw）

2.2 这个值从哪里来？

使用点播上传完成之后都会得到一个videoId.在阿里云点播控制台里面的视频列表里面会有一个“视频ID”即为videoID.可以拿到控制台的视频ID来做下载和播放的测试.
如何实现点播上传见：
https://help.aliyun.com/document_detail/52200.html?spm=5176.doc52858.6.667.bjm8cC

三、获取AccessKeyId和AccessKeySecret

3.1 AccessKeyId和AccessKeySecret是什么？

阿里云 access key ID 和 access key secret 是您访问阿里云API的唯一凭证。Access key ID 是类似身份的标识，而 access key secret 的作用是签名您的访问参数，以防被篡改。Access key secret 类似您的登录密码，不要向任何人泄漏。

3.2 这个值从哪里来？

1.登录阿里云官方网站。
2.单击页面上方菜单控制台。
3.鼠标放在右上方的用户名区域，在弹出的快捷菜单中单击AccessKeys。
4.系统弹出安全提示对话框，单击 继续使用AccessKey。页面显示 Access Key ID 和 Access Key Secret 。

四、获取playKey(同apikey)

4.1 playKey是什么？

playkey(apikey):播放密钥,用于播放器SDK获取视频播放地址时验证身份，播放鉴权是视频点播在阿里云AK安全认证基础上的二次鉴权机制。播放密钥用于播放器SDK获取视频播放地址时验证身份，可有效防止盗链。根据用户播放时可能使用的平台，默认提供Flash、H5、iOS、Android四个平台的播放密钥。

为保证密钥安全，查看播放密钥时需要输入手机号验证码确认身份。

4.2 这个值从哪里来？

1.登录阿里云官方网站。
2.单击页面上方菜单控制台。
3.选择云计算基础服务
4.选择视频点播
4.选择安全设置
5.选择播放鉴权
6.选择对应的平台点击显示即可

五、获取playauth

5.1 playauth是什么？

目前播放器播放视频分为三种模式,三种模式有着不同的使用场景.playauth就是最安全的方法，也就是setAuthInfo的方式来实现.

播放模式	适用场景	优劣	是否建议使用
setDataSource	适用于测试时为了方便测试使用	危险，需要将自己的ak写死在客户端，客户端如果被破解将有泄露风险	不建议商用时上线使用
setAuthInfo	适用于正式商用	安全，所有的视频地址和链接都不暴露	建议商用使用
播放本地和网络URL	可以播放本地视频且能够播放视频URL	简单，可以播放其他平台的视频	在有播放本地视频需求和播放网络视频需求时

			使用.

5.2 这个值从哪里来？

playauth可以理解为点播服务将所有的信息(VideoId、AccessKeyId、AccessKeySecret)都做了一个混合然后做了一个简单的加密。这样用户拿到的就是一串包含多个信息的数据.这样播放器就可以播放了.如何获取见一下流程：

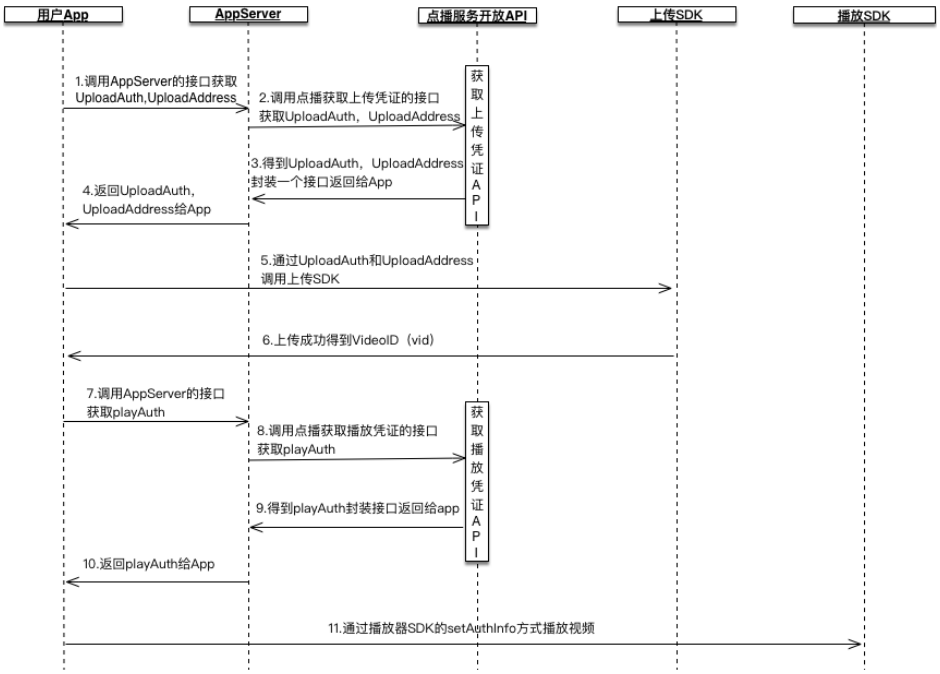
流程：服务端获取播放凭证 -> 将播放凭证下发给客户端 -> 完成视频播放。

1. 获取播放凭证
客户在服务器侧通过调用播放鉴权SDK(server端的SDK)，向视频点播服务获取播放凭证。

2. 完成视频播放
播放器SDK根据视频ID和播放凭证向视频点播服务获取视频的播放地址，从而加载视频流并解码完成播放。

注意：
(1)播放凭证时效为100秒，只能用于获取指定视频的播放地址，不能混用或重复使用；如果凭证过期则无法获取播放地址，请重新获取凭证。
(2) 播放器SDK根据播放凭证会自动获取播放地址进行解码播放，播放地址的时效为30分钟，若失效请重新获取播放凭证回传给播放器SDK用于刷新播放地址。
(3)为保障主账号安全，建议使用子账号的Access Key，尤其是Web播放场景下。

5.3 上传、播放最安全且最推荐的做法流程（重要）



Aliplayer Web播放器SDK，同时支持Flash和Html5两种播放技术。

支持格式

Flash 模式：

- 视频格式： mp4、flv、m3u8、rtmp、mp3
- 视频编码： H.264
- 音频编码： AAC、MP3

HTML5 模式：

- 视频格式： mp4、m3u8
- 视频编码： H.264
- 音频编码： AAC
- 音频格式： mp3

flash支持加密播放点播加密说明

m3u8格式播放依赖调用端浏览器支持情况：

iOS全系列支持，Android 4.0及以上版本支持；

PC端支持的浏览器：

- Chrome for Desktop 34+
- Firefox for Desktop 42+
- IE11+ for Windows 8.1+
- Edge for Windows 10+
- Opera for Desktop
- Safari for Mac 8+ (beta)

播放器功能说明

如何正确选择播放器

如何连续播放视频

如何在PC上播放hls视频

如何处理Flash跨域访问

如何启用H5的同层播放

清晰度切换的使用

诊断工具的使用

skinLayout属性的配置

H5自定义错误UI

直播出错恢复处理

如何使用截图

如何使用延迟播放

常见问题

更多demo和说明请参见文档Aliplayer播放器

扫描体验

资源文件

不依赖于任何的前端js库，只需要在页面中引用如下js文件，就可以进行播放器的初始化：

```
https://g.alicdn.com/de/prismplayer/2.2.0/aliplayer-min.js
```

这个文件同时包含了Flash和Html5跨终端自适应的逻辑。如果您只是想使用其中一种播放技术，也可以只引用

对应技术的js文件，从而获得更小的文件体积：

Flash版本:

```
https://g.alicdn.com/de/prismplayer/2.2.0/aliplayer-flash-min.js
```

Html5版本:

```
https://g.alicdn.com/de/prismplayer/2.2.0/aliplayer-h5-min.js
```

如果您的使用场景需要用到html5播放器，请额外引用css文件：
https://g.alicdn.com/de/prismplayer/2.2.0/skins/default/aliplayer-min.css

使用说明

Web播放器SDK，同时支持两种播放方式，即：

- 方式一：通过播放地址播放；
- 方式二：通过获取播放凭证完成视频播放；

默认和推荐使用方式一。详细描述请参见文档 播放SDK-使用说明。

一个简单的使用实例

提供一个最简单的使用实例，初始化播放器，监听某个dom元素的点击事件来触发调用播放器的接口方法，同时对播放器暴露的事件进行监听。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1, maximum-scale=1,
minimum-scale=1, user-scalable=no"/>
<title>用户测试用例</title>
<link rel="stylesheet" href="//g.alicdn.com/de/prismplayer/2.2.0/skins/default/aliplayer-min.css" />
<script type="text/javascript" src="//g.alicdn.com/de/prismplayer/2.2.0/aliplayer-min.js"></script>
</head>
<body>
<div class="prism-player" id="J_prismPlayer" style="position: absolute"></div>
<script>
var player = new Aliplayer({
id: 'J_prismPlayer',
width: '100%',
autoplay: false,

//支持播放地址播放,此播放优先级最高
source : '播放url',

//播放方式二：推荐
vid : '1e067a2831b641db90d570b6480fbc40',
```

```
playauth : '',
cover: 'http://liveroom-img.oss-cn-qingdao.aliyuncs.com/logo.png'
});
</script>
</body>
</html>
```

播放器配置属性

id，{String}，播放器外层容器的dom元素id

source, {String}, 视频播放地址url

vid，{String}，视频id。

playauth，{String}，播放凭证。

height，{String}，播放器高度，可形如’100%’或者’100px’

width，{String}，播放器宽度，可形如’100%’或者’100px’

preload，{Boolean}，播放器自动加载，目前仅h5可用

cover, {String}, 播放器默认封面图片，请填写正确的图片url地址，Flash播放器封面也需要开启允许跨域访问

isLive，{Boolean}，播放内容是否为直播，直播时会禁止用户拖动进度条

autoplay，{Boolean}，播放器是否自动播放，html5受浏览器限制，大多数情况下该配置会失效

rePlay，{Boolean}，播放器自动循环播放

useH5Prism，{Boolean}，指定使用H5播放器

useFlashPrism，{Boolean}，指定使用Flash播放器

playsinline, {Boolean}, H5是否内置播放，有的Android浏览器不起作用

showBuffer，{Boolean}，显示播放时缓冲图标，默认true

trackLog, {Boolean}，是否需要进行用户行为日志打点，默认为true

skinRes，{Url}，皮肤图片，不建议随意修改该字段,如要修改，请参照Web播放器皮肤定制

skinLayout，{Array | Boolean}，功能组件布局配置，不传该字段使用默认布局，传false隐藏所有功能组件，功能与皮肤定制方法请看下一节

showBarTime，{String}，控制栏自动隐藏时间（ms）

controlBarVisibility:，{String}，控制控制面板的现实，默认为’click’，可选的值为：’click’、’hover’、’always’

waterMark，{url|pos|size|alpha},添加水印，目前仅支持flash，url：水印图片（jpg/png）；pos：位置（TL/TR/BL/BR）;size: logo宽度占播放器比例（0-1，默认0.2）；alpha：透明度（0-1，默认1）。示例：waterMark:"logo.jpg|TL|0.15|0.5"

extraInfo，{“a_key”:”a_value”,”b_key”:”b_value”}，类型为json串，定制性接口参数，目前支持

```
 "fullTitle":"测试页面"，                  //全屏时显示视频标题(目前仅flash支持)；
"m3u8BufferLength":"30"， //播放m3u8时加载缓存ts文件长度(目前仅flash支持)，单位（秒）；
"liveStartTime":"2016/08/17 12:00:00", //直播开始时间，用于提示直播未开始(目前仅flash支持)；
"liveOverTime":"2016/08/17 14:00:00", //直播结束时间，用于提示直播结束(目前仅flash支持)；
"liveRetry":1, //直播流中断是否重试；
```

enableSystemMenu，Boolean，是否允许系统右键菜单显示，默认为false

format，Sting，指定播放地址格式，只有使用vid+plauth播放方式时支持，可选值为’mp4’和’m3u8’,默认为’mp4’

qualitySort，Sting，指定排序方式，只有使用vid+plauth播放方式时支持，‘desc’表示按倒序排序(即：从大到小排序)，‘asc’表示按正序排序(即：从小到大排序)，默认值为’asc’

x5_type，String，声明启用同层H5播放器，启用时设置的值为’h5’，具体参考同层播放

x5_fullscreen，Boolean，声明视频播放时是否进入到TBS的全屏模式，默认为false，具体参考同层播放

x5_orientation，Boolean，声明TBS播放器支持的方向，可选值：0：landscape 横屏，1：portraint竖屏，2：landscape | portrait跟随手机自动旋转，具体参考同层播放

autoPlayDelay，Number，延迟播放时间，单位为秒，具体参考延迟播放

autoPlayDelayDisplayText，String，延迟播放提示文本,具体参考延迟播放

language, String, 国际化，默认为’ zh-cn’，如果未设置，则采用浏览器语言,可选值为’ zh-cn’ or ‘en-us’

snapshot, Boolean, flash启用截图功能, 可参考如何使用截图

useHlsPluginForSafari，Boolean，Safari浏览器可以启用Hls插件播放，Safari 11除外

播放器API

play()，播放视频

pause()，暂停视频

replay()，重播视频

seek(time)，跳转到某个时刻进行播放，time的单位为秒

getCurrentTime()，获取当前的播放时刻，返回的单位为秒

getDuration()，获取视频总时长，返回的单位为秒

getVolume()，获取当前的音量，返回值为0-1的实数，ios和部分android会失效

setVolume(vol)，设置音量，vol为0-1的实数，ios和部分android会失效

loadByUrl(url,time)，直接播放视频url，time为可选值（单位秒）目前只支持同种格式（mp4/flv/m3u8）之间切换，暂不支持直播rtmp流切换

reloaduserPlayInfoAndVidRequestMts(vid,playauth)，vid，{String}，视频id；playauth,{String},播放凭证。目前只支持HTML5界面上的重载功能，暂不支持直播rtmp流切换；注意参数顺序。

setPlayerSize(w,h)，设置播放器大小，w,h可分别为400px像素或60%百分比，chrome浏览器下flash播放器分别不能小于397x297

setSpeed，speed， 设置倍速播放，移动端可能会失效，比如android 微信

setSanpshotProperties，参数：width 宽度 ，height 高度 ，rate 截图质量，设置截图参数，可参考如何使用截图

播放器对外事件

ready，播放器初始化完毕可以播放视频时触发，播放器UI初始设置需要此事件后触发，避免UI被初始化所覆盖，播放器提供的方法需要在此事件发生后才可以调用

play，视频由暂停恢复为播放时触发

pause，视频暂停时触发

ended，当前视频播放完毕时触发

liveStreamStop，直播流中断时触发,m3u8/flv/rtmp在重试5次未成功后触发，提示上层流中断或需要重新加载视频

m3u8Retry，m3u8直播流中断后重试事件，每次断流只触发一次

hideBar，控制栏自动隐藏事件

waiting，数据缓冲事件

snapshoted，截图完成时间，可参考如何使用截图

requestFullScreen，全屏事件

- cancelFullScreen，取消全屏事件

如果问题还未能解决，请联系 售后技术支持。

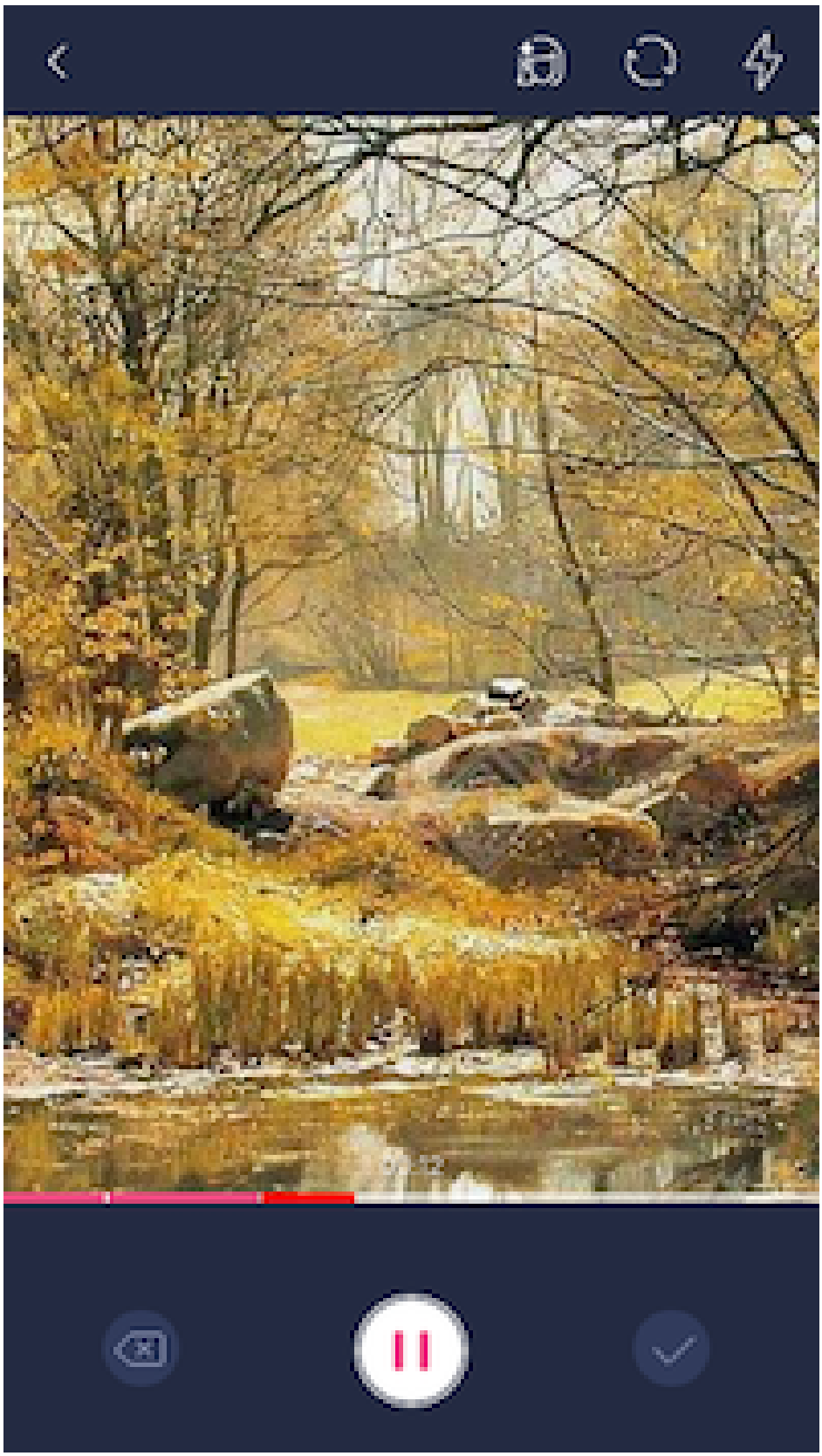
短视频SDK

一、简介

短视频SDK提供短视频录制、导入和编辑的高级功能，支持多种分辨率选择、实时美颜、实时滤镜、摄像头切换、闪光灯切换、对接人脸识别SDK实现人脸贴图等多样的录制功能、支持视频画面和时长裁剪、多视频拼接、添加滤镜、动图、音乐、MV、字幕、涂鸦等短视频高级编辑。提供产品级的UI开源界面，方便用户根据自己的业务定制界面，提供易用、稳定、统一的视频录制、导入裁剪和编辑高级接口，实现真正的二次开发、做到真正个性化。

二、功能说明

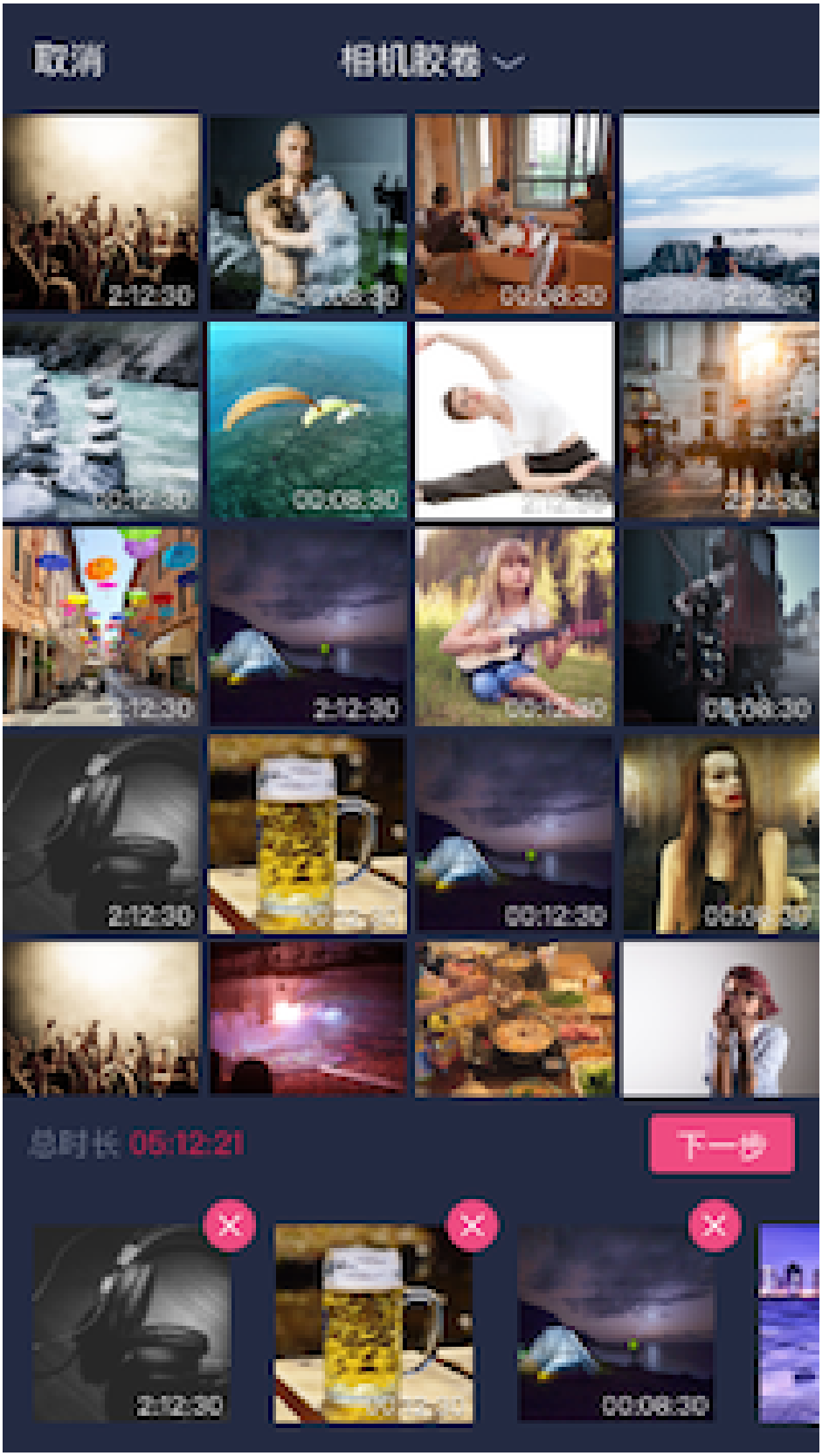
录制 支持断点录制、回删、点击拍摄、长按拍摄、美颜、实时滤镜、闪光灯、实时水印、摄像头切换、分辨率设定、对接第三方人脸库实现人脸贴图、实时混音和变速等功能

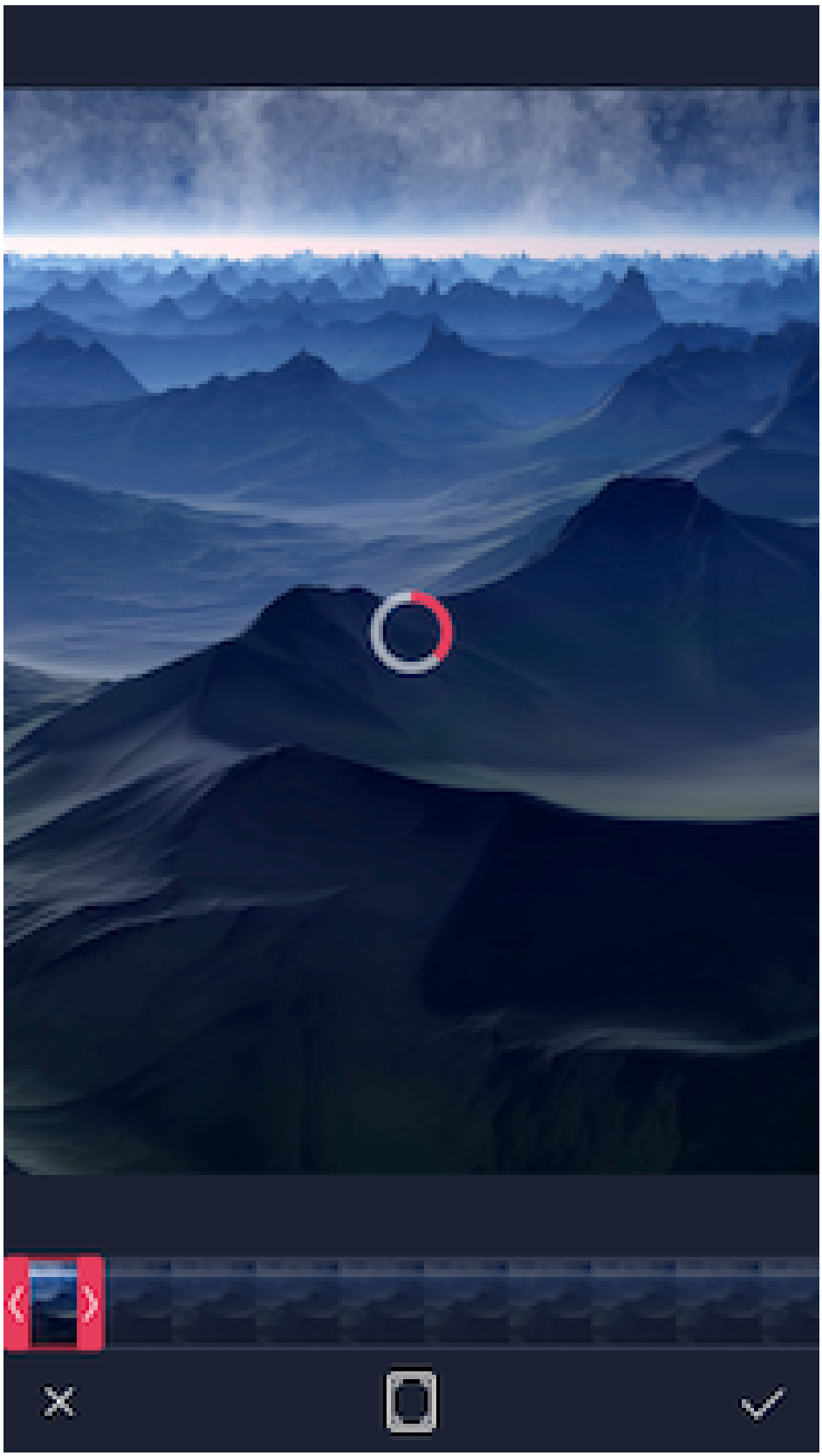



茨曲

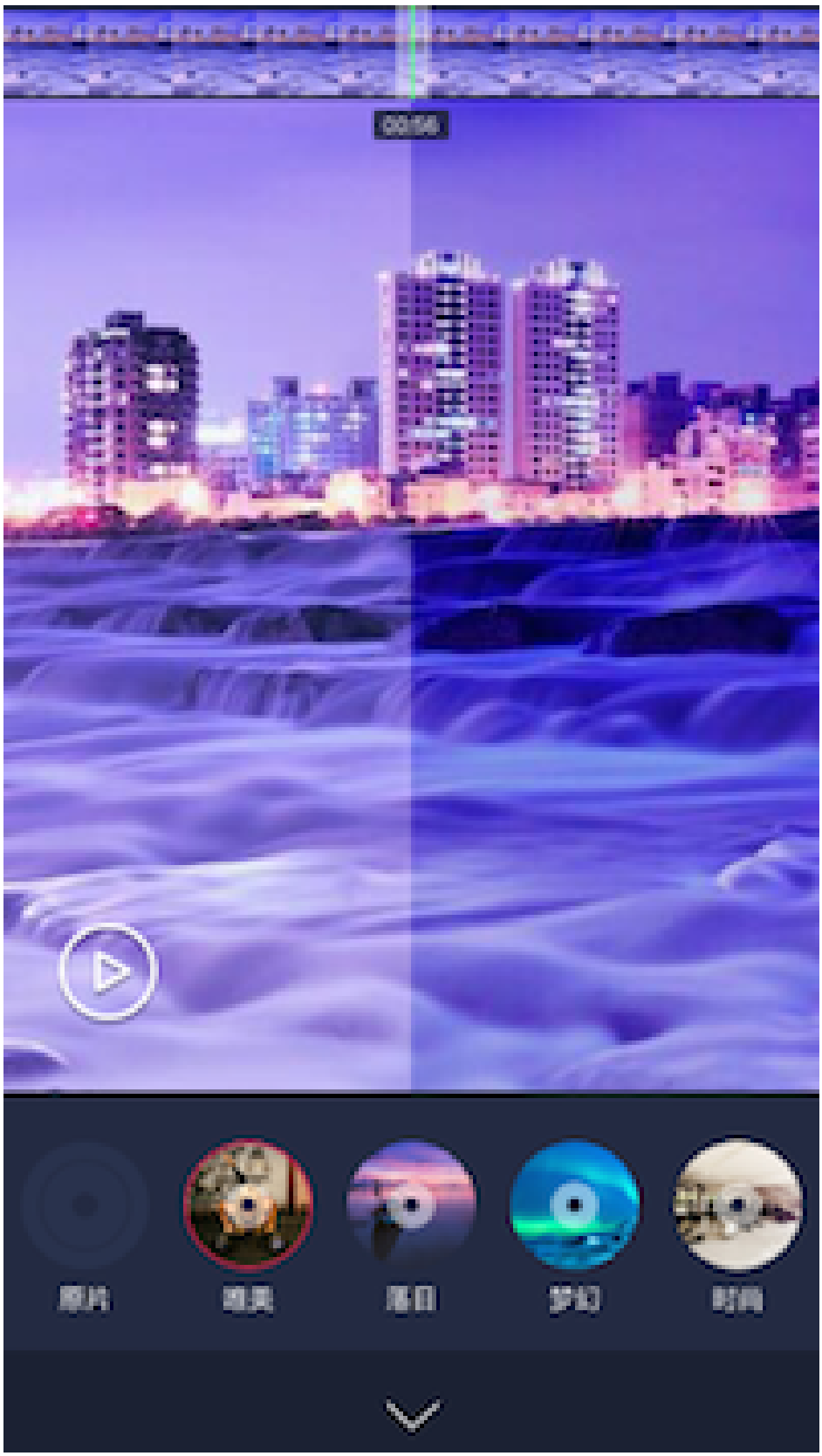

肥美

导入 支持从相册选择视频、按视频时长和画面进行裁剪。支持多视频/多照片/照片和视频混合导入拼接、并可设置转场模式和转场持续时间


取消
相机胶卷
总时长 05:12:21
下一步



编辑 支持在编辑界面添加滤镜、字幕（含普通字幕和气泡字幕）、贴图（支持动态和静态）、MV、音乐，涂鸦（支持画笔粗细调整，颜色调整和撤销）等功能。


00:06
原片
唯美
落日
梦幻
时间


编辑
走你走过路口
完成
填充
描边


元气少女

4. **demo体验**

扫一扫下载体验

三、核心优势

- **快速接入 成本经济**
  提供产品级SDK，最快2小时接入，节省自行开发耗费的人力物力，助你快速实现APP 短视频功能
- **接口简单，开放强性**
  接口简单易用，开放性强，专业版（UI开源）可以根据业务自由定制UI
- **功能齐备 应用广泛**
  录制功能自带断点录制、实时滤镜、高效美颜、人脸贴图接口功能，支持本地视频导入压缩裁剪，对视频添加MV、动图、字幕、音乐等高级功能。
- **迭代打磨 稳定可靠**
  视频技术经钉钉、美柚、梨视频、迅雷、贝贝网、宝宝树、蚂蜂窝等1000+应用商用验证，稳定可靠

四、基础版、标准版、专业版功能比较

功能点	功能说明	基础版	标准版	专业版
自定义UI	SDK包含一套默认的UI，布局、交互、界面可二次开发，基础版支持图标和背景	×	√	√

	颜色替换、标准版UI完全自定义			
UI开源	提供完整的UI交互源码，用户可定制UI界面	×	√	√
多段录制	支持断点拍摄和连续拍摄	√	√	√
自定义时长	自定义最长和最短拍摄时长	√	√	√
摄像头切换	选择使用前后摄像头进行录制	√	√	√
闪光灯	支持打开、关闭、自动闪光灯模式	√	√	√
实时水印	支持在录制时添加水印	√	√	√
焦距调节	录制中可调节画面焦距进行放大缩小	√	√	√
自定义分辨率及质量	可设定拍摄的画面尺寸、比例和质量，基础版仅支持9:16、3:4、1:1三种比例，标准版、专业版任意分辨率	√	√	√
美颜	录制实时美颜，平滑无极调整强度	√	√	√
实时滤镜	拍摄预览界面实时切换滤镜	√	√	√
人脸识别	内置人脸识别功能，进行人脸贴纸操作	×	×	√
人脸识别接口	支持外接人脸识别点，进行人脸贴纸操作	×	×	√
实时混音和变速	支持录制界面添加音乐，变速功能。变速支持调整倍数	×	√	√
相册选择	支持从相册过滤视频，也支持视频时长过滤	√	√	√
照片裁剪	支持照片画面大小的裁剪，同时支持画面填充和	√	√	√

	画面裁剪			
视频裁剪	支持视频画面大小和时长裁剪，同时支持画面填充和画面裁剪	√	√	√
原比例裁剪	支持保持原始视频比例裁剪视频时长	×	√	√
单视频导入	支持单视频导入，跳转进入用户定义的页面	√	√	√
多照片导入	支持多张照片导入，跳转进入用户定义的页面	×	×	√
多视频导入	支持多视频导入，进入编辑界面	×	×	√
视频和照片导入	支持多个视频多张照片混合导入，进入编辑界面	×	×	√
滤镜	在编辑界面添加滤镜，切换滤镜	×	×	√
动图	在编辑界面添加动图，可在任意时间点添加并支持时间调整	×	×	√
MV	在编辑界面添加MV效果，切换MV	×	×	√
音乐	支持将网络音乐和本地音乐合成到视频中	×	×	√
静音	支持消除当前视频的原音和音乐声音	×	×	√
字幕	支持普通文字字幕和气泡效果字幕，并且可更换字体	×	×	√
片尾	支持在视频末尾添加片尾水印效果，可定义持续时间	×	×	√
涂鸦	支持画笔尺寸和颜色调整	×	×	√
SDK下载	短视频SDK下载	基础版	标准版	专业版

五、license申请说明（请仔细查看）

1. 购买点播SaaS预付费指定套餐即送短视频SDK一年license。套餐一、二、三对应基础版SDK、套餐四对应标准版SDK。【立刻购买】
2. 购买专业版无需绑定云服务，按功能模块进行售卖。【立刻购买】
3. 购买套餐后请提供应用名、订单号、bundleID、包名和签名（MD5格式小写无冒号），并发送至videosdk@service.aliyun.com，以便为您开通短视频SDK License。
4. license不需要集成到SDK里面，只需确保提交申请的bundleID、包名、签名和自己工程中的完全一致。测试时可以直接使用demo提供的bundleID、包名、签名来体验。
5. 如果成功获取了license，在运行app时还显示license无效，请先清除下工程再运行app。
6. 基础版和标准版购买1个套餐最多支持1个app（最多支持10个马甲包），专业版1次购买最多支持3个app（最多支持10个马甲包）。

六、关于SDK试用说明

1. 请发送公司名称、应用名称、申请试用的SDK版本、联系人、联系电话、应用bundleID、包名和签名信息（MD5格式小写无冒号）、阿里云的账号/UID（若没有账号请注册）至videosdk@service.aliyun.com，申请开通试用，或直接联系商务经理申请开通。
2. 试用期默认为一个月，需延长试用期续参考步骤一再次申请。

七、专业版如何购买

请将您的联系方式和需求发送邮件至videosdk@service.aliyun.com，商务人员会联系您详细沟通。点击SDK下载使用demo提供的bundleID、包名和签名进行测试。

Android短视频SDK

一、版本要求

Android支持4.3及以上

二、开发环境配置

本SDK开发环境为 JAVA1.7 | ANDROID SDK API LEVEL 18

三、使用说明

导入SDK

先下载SDK，详见SDK下载页面。

SDK包括QuSDK-RC.aar,libQuCore.so,libQuCore-ThirdParty.so 3个基本文件，使用时只需要将.aar文件放入项目libs文件夹中，.so文件放入libs/armeabi-v7a文件夹中即可。

接入

使用aar：aar 文件放入引用Module的libs目录下，gradle配置文件中把libs 目录放入依赖：

```
repositories{
flatDir{
dirs 'libs'
}
}
```

在gradle文件中使用依赖的方式引用aar：

```
compile(name:'xxx',ext:'aar')
```

使用so:

so 文件放入引用Module的libs/armeabi-v7a目录下，gradle配置文件中把libs 目录放入依赖：

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

在代码中加入:

```
   System.loadLibrary("QuCore-ThirdParty");
System.loadLibrary("QuCore");
```

SDK配置

设置AndroidManifest.xml文件，声明使用权限(必须)

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

使用本SDK需要依赖外部第三方包，目前包含（必须）：
注意：目前闭源support版本的包必须跟以下版本一致.

```java
compile 'com.android.support:appcompat-v7:24.2.1'
compile 'com.android.support:design:24.2.1'
compile 'com.google.code.findbugs:jsr305:3.0.0'
compile 'com.github.bumptech.glide:glide:3.7.0'
compile 'pub.devrel:easypermissions:0.2.1'
compile 'com.squareup.okhttp3:okhttp:3.2.0'
compile 'com.github.bumptech.glide:okhttp3-integration:1.4.0@aar'
compile 'com.squareup.okio:okio:1.12.0'
compile 'com.google.code.gson:gson:2.8.0'
```

如果使用了jackson，请在app中的build.gradle中添加：

```
    android{
packagingOptions {
exclude('META-INF/LICENSE')
}
}
```

在应用启动的时候，请做如下初始化工作(必须)：

```
    QupaiHttpFinal.getInstance().initOkHttpFinal();
```

UI配置

UI配置需要为Activity指定主题，定义如下(非必须指定，如不指定即为默认主题,如有疑义请参考demo配置)

```
  <style name="AliyunVideoUIStytle" >
<item name="qusnap_background_color">@color/color_bg</item> //背景主题色
<item name="qusnap_tint_color">@color/tint_color</item> //录制进度条颜色
<item name="qusnap_timeline_backgound_color">@color/timeline_backgound_color</item> //录制进度条背景色
<item name="qusnap_timeline_del_backgound_color">@color/timeline_background_del_color</item> //录制删除进度颜色
<item name="qusnap_back_icon">@mipmap/icon_back</item> // 返回按钮图标
<item name="qusnap_switch_light_icon">@drawable/snap_switch_light_selector</item> //闪光灯选择器
<item name="qusnap_switch_light_icon_disable">@mipmap/icon_light_dis</item>// 闪光灯禁用图标
<item name="qusnap_switch_light_icon_visibility">visible</item> // 闪光灯显隐属性
<item name="qusnap_switch_camera_icon">@drawable/snap_switch_camera</item> //摄像头选择器
<item name="qusnap_switch_camera_icon_visibility">visible</item> //摄像头显隐属性
<item name="qusnap_beauty_icon">@drawable/snap_switch_beauty</item> //美颜选择器
<item name="qusnap_beauty_icon_visibility">visible</item> // 美颜显隐属性
<item name="qusnap_record_icon">@drawable/snap_record_state_selector</item> //录制选择器
<item name="qusnap_delete_icon">@drawable/snap_icon_delete</item> //删除选择器
```

```
<item name="qusnap_complete_icon">@drawable/snap_icon_complete</item> //完成选择器
<item name="qusnap_gallery_icon">@mipmap/icon_default</item> //相册按钮
<item name="qusnap_gallery_icon_visibility">visible</item>//相册的显示隐藏
<item name="qusnap_time_txt_color">@android:color/white</item> //录制时间文字颜色
<item name="qusnap_time_txt_size">15dp</item> //录制时间文字大小
<item name="qusnap_time_txt_visibility">visible</item> //录制时间文字显隐属性
<item name="qusnap_time_line_pos_y">0dp</item> //录制进度条向上偏移量
<item name="qusnap_crop_sweep_left">@mipmap/icon_sweep_left</item> //裁剪滑动条左图标
<item name="qusnap_crop_sweep_right">@mipmap/icon_sweep_right</item> //裁剪滑动条右图标
<item name="qusnap_crop_seek_frame">@mipmap/icon_frame</item> //裁剪播放帧进度图标
<item name="qusnap_crop_seek_padding_color">@android:color/holo_red_dark</item> //裁剪滑动条上
下边框颜色
<item name="qusnap_crop_icon_transform">@drawable/snap_transform_selector</item> //裁剪模式切换
选择器
<item name="qusnap_crop_icon_transform_visibility">visible</item> //裁剪模式显隐属性
<item name="qusnap_crop_time_txt_color">@android:color/white</item> //裁剪视频时长文字颜色
<item name="qusnap_crop_time_txt_size">15dp</item> //裁剪视频时长文字大小
<item name="qusnap_crop_txt_visibility">visible</item> //裁剪视频时长文字显隐属性
</style>
```

录制

初始化参数使用录制功能，需要初始化AliyunSnapVideoParam对象，初始化方法如下：

```
AliyunSnapVideoParam recordParam = new AliyunSnapVideoParam.Builder()
//设置录制分辨率，目前支持360p，480p，540p，720p
.setResulutionMode(AliyunVideoRecorder.RESOLUTION_360P)
//设置视频比例，目前支持1:1,3:4,9:16
.setRatioMode(AliyunVideoRecorder.RATIO_MODE_9_16)
.setRecordMode(RecorderDemo.RECORD_MODE_AUTO) //设置录制模式，目前支持按录，点录和混合模式
.setFilterList(eff_dirs) //设置滤镜地址列表,具体滤镜接口接收的是一个滤镜数组
.setBeautyLevel(80) //设置美颜度
.setBeautyStatus(true) //设置美颜开关
.setCameraType(CameraType.FRONT) //设置前后置摄像头
.setFlashType(FlashType.ON) // 设置闪光灯模式
.setNeedClip(true) //设置是否需要支持片段录制
.setMaxDuration(max) //设置最大录制时长 单位毫秒
.setMinDuration(min) //设置最小录制时长 单位毫秒
.setVideQuality(videoQuality) //设置视频质量
.setGop(gop) //设置关键帧间隔
.setVideoBitrate(2000) //设置视频码率，如果不设置则使用视频质量videoQulity参数计算出码率
.setSortMode(AliyunSnapVideoParam.SORT_MODE_VIDEO)//设置导入相册过滤选择视频
.build();
```

拉起录制界面

```
AliyunVideoRecorder.startRecordForResult(this,REQUEST_RECORD,recordParam);
```

回调

需要回调请重写`onActivityResult`函数，其中有返回类型参数，代表是从裁剪返回还是从录制返回

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if(requestCode == REQUEST_RECORD){
if(resultCode == Activity.RESULT_OK && data!= null){
int type = data.getIntExtra(AliyunVideoRecorder.RESULT_TYPE,0);
if(type == AliyunVideoRecorder.RESULT_TYPE_CROP){
String path = data.getStringExtra(AliyunVideoCrop.RESULT_KEY_CROP_PATH);
Toast.makeText(this,"文件路径为 "+ path + " 时长为 " +
data.getLongExtra(AliyunVideoCrop.RESULT_KEY_DURATION,0),Toast.LENGTH_SHORT).show();
}else if(type == AliyunVideoRecorder.RESULT_TYPE_RECORD){
Toast.makeText(this,"文件路径为 "+
data.getStringExtra(AliyunVideoRecorder.OUTPUT_PATH),Toast.LENGTH_SHORT).show();
}
}else if(resultCode == Activity.RESULT_CANCELED){
Toast.makeText(this,"用户取消录制",Toast.LENGTH_SHORT).show();
}
}
}
```

导入裁剪

初始化参数 使用裁剪功能，需要初始化AliyunSnapVideoParam对象，初始化方法如下：

```
AliyunSnapVideoParam mCropParam = new AliyunSnapVideoParam.Builder()
.setFrameRate(frameRate) //设置帧率
.setGop(gop) //设置关键帧间隔
.setCropMode(cropMode) //设置裁剪模式，目前支持有黑边和无黑边两种
.setVideQuality(videoQulity) //设置裁剪质量
.setVideoBitrate(2000) //设置视频码率，如果不设置则使用视频质量videoQulity参数计算出码率
.setCropUseGPU(true) //设置裁剪方式，是否使用gpu进行裁剪，不设置则默认使用cpu来裁剪
.setResulutionMode(resolutionMode) //设置分辨率，目前支持360p，480p，540p，720p
.setRatioMode(ratioMode)//设置裁剪比例 目前支持1:1,3:4,9:16
.setNeedRecord(true)//设置是否需要开放录制入口
.setMinVideoDuration(4000) //设置过滤的视频最小长度 单位毫秒
.setMaxVideoDuration(29 * 1000) //设置过滤的视频最大长度 单位毫秒
.setMinCropDuration(3000) //设置视频最小裁剪时间 单位毫秒
.build();
```

- **拉起裁剪页面**

```
AliyunVideoCrop.startCropForResult(this,REQUEST_CROP,mCropParam);
```

回调

需要回调请重写onActivityResult函数，其中有返回类型参数，代表是从裁剪返回还是从录制返回

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if(requestCode == REQUEST_CROP){
if(resultCode == Activity.RESULT_OK && data!= null){
int type = data.getIntExtra(MediaActivity.RESULT_TYPE,0);
if(type == MediaActivity.RESULT_TYPE_CROP){
String path = data.getStringExtra(AliyunVideoCrop.RESULT_KEY_CROP_PATH);
Toast.makeText(this,"文件路径为 "+ path + " 时长为 " +
data.getLongExtra(AliyunVideoCrop.RESULT_KEY_DURATION,0),Toast.LENGTH_SHORT).show();
}else if(type == MediaActivity.RESULT_TYPE_RECORD){
Toast.makeText(this,"文件路径为 "+
data.getStringExtra(AliyunVideoRecorder.OUTPUT_PATH),Toast.LENGTH_SHORT).show();
}
}else if(resultCode == Activity.RESULT_CANCELED){
Toast.makeText(this,"用户取消裁剪",Toast.LENGTH_SHORT).show();
}
}
}
```

四、上传

SDK生成的mp4文件，可接入阿里云上传SDK上传文件，可跳转至阿里云上传SDK下载。点击Android上传SDK获取使用文档。

一、系统版本

1.1 版本要求

Android支持4.3及以上

二、快速开始

2.1 开发环境配置

本SDK开发环境为 JAVA1.7 | ANDROID SDK API LEVEL 18

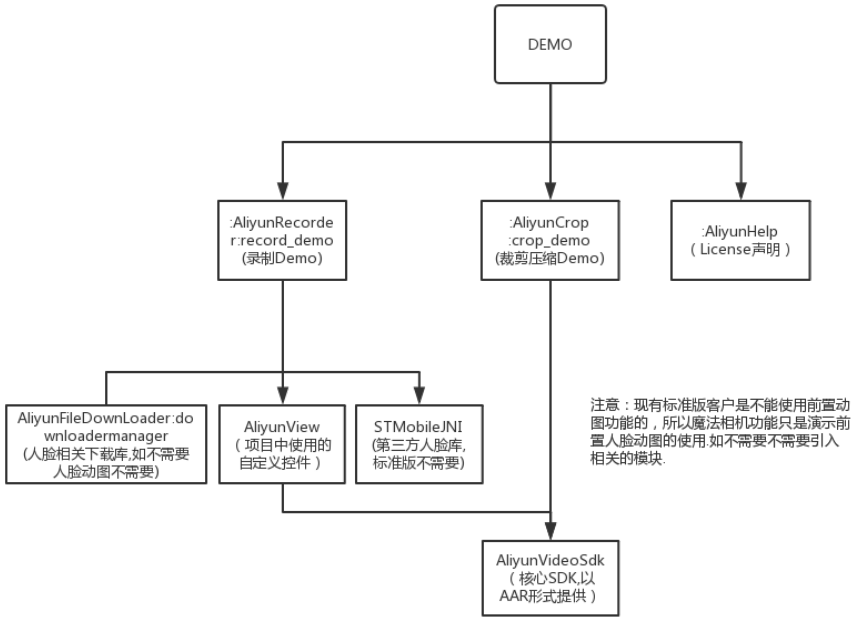
2.2 导入SDK

2.2.1 简介

SDK示例包括
DEMO,:AliyunRecorder:record_demo,:AliyunCrop:crop_demo,:AliyunHelp,AliyunView,AliyunFileDownLoader:downloadermanager,:AliyunVideoSdk,STMobileJNI等8个基本moudle，使用Demo时只需要将整个根文件夹作为ANDROID STUDIO的项目导入即可。
其中DEMO是作为整个SDK的模块选择界面
AliyunCrop:crop_demo：是裁剪模块的示例代码
AliyunRecorder:record_demo：是录制模块的示例代码
AliyunVideoSdk：核心SDK,以AAR形式提供
AliyunHelp：阿里云License声明
AliyunFileDownLoader:downloadermanager：人脸动图需要的下载管理模块,如无人脸动图需求不需要导入
STMobileJNI：第三方人脸库，目前仅仅做了代码演示，引导用户接入人脸识别库，但实际上因为License授权问题，该人脸库在本Demo中不可用
AliyunView：项目中使用的自定义控件

2.2.2 SDK的文件结构如图所示：

2.2.3 SDK模块依赖关系：



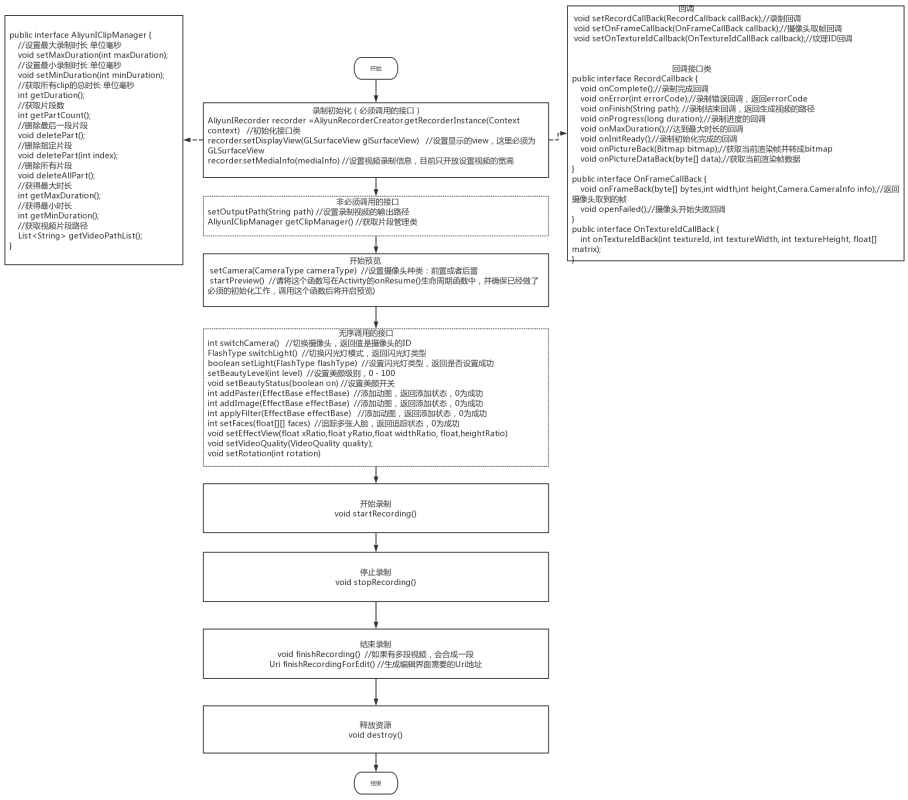
2.3 权限要求

```
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

三、功能使用

3.1 录制

录制模块功能接口调用顺序图：



3.1.1初始化参数

使用录制功能，需要初始化AliyunIRecorder对象，初始化方法如下：

```
AliyunIRecorder recorder = AliyunRecorderCreator.getRecorderInstance(Context context);//参数context为当前页面的上下文


recorder.setDisplayView(GLSurfaceView glSurfaceView);//参数glsurfaceView为用户自己定义的GLSurfaceView及其子类对象
```

设置视频参数信息：

```
MediaInfo mediaInfo = new MediaInfo();
mediaInfo.setVideoWidth(TEST_VIDEO_WIDTH);
mediaInfo.setVideoHeight(TEST_VIDEO_HEIGHT);
mediaInfo.setAutoAdjustHWEncoder(true);//硬编时自适应宽高为16的倍数
recorder.setMediaInfo(mediaInfo);
```

注意：录制时的分辨率是按照GLSurfaceView的实际宽高比，以设置的分辨率的宽为基准生成的，比如用户设置的分辨率是480x480，如果GLSurfaceView的宽高比为9:16，则录制的视频分辨率为480:854，如果想要保持480x480的分辨率，则GLSurfaceView的宽高比要保持1:1。硬编自适应开关如果设为true，则使用硬编（目前只支持硬编）时，如果宽 / 高不是16的倍数，则会强制转换为16的倍数，这样有一个影响就是传入的分辨率与实际输出分辨率会有一点出入。而如果设置为false，则如果宽 / 高不是16的倍数，在调用setMediaInfo时将抛出IllegalArgumentException。

3.1.2开始预览&结束预览开始预览如下方法：

```
void setCamera(CameraType camera); //设置预览的摄像头，前置或者后置
void startPreview();//请将这个函数写在Activity的onResume生命周期中，调用这个函数后将开始预览
```

结束预览如下方法：

```
void stopPreview();//请将这个函数写在Activity的onPause生命周期中，调用这个函数后将结束预览
```

3.1.3开始录制&结束录制

在录制之前，确保已设置录制文件的输出路径，通过AliyunIRecorder的outputPath属性进行设置。

设置传感器角度值，这个必须设置，否则录制的视频角度是错的。

```
void setRotation(int rotation);
```

开始录制接口：

```
void startRecording();
```

结束录制接口：

```
void stopRecording();
```

完成录制接口：

```
void finishRecording(); //调用完成录制的接口会合并录制的多段视频
```

生成录制片段信息路径接口（录制的多段视频会序列化到一个json描述文件中，该接口可以返回描述文件的地址，用于编辑等操作）

```
void finishRecordingForEdit(); //调用接口会生成一个配置文件的Uri
```

取消录制接口：

```
void cancelRecording();
```

3.1.4拍照

```
/**
* 拍照（从GPU中抓取数据，可以包含特效）
* @param needBitmap 是否需要生成Bitmap
*/
void takePhoto(boolean needBitmap);
```

```
/**
* 使用系统的拍照接口（不能包含特效）
* @param needBitmap 是否需要生成Bitmap
*/
void takePicture(boolean needBitmap);
```

3.1.5录制变速

```
void setRate(float rate);//录制变速，速率在0.5-2.0之间
```

3.1.6录制配乐

```
void setMusic(String path,long startTime,long duration);//path:音乐文件路径(建议使用aac格式的音频) startTime:音乐开始时间 duration:音乐时长
```

3.1.7录制静音

```
void setMute(boolean isMute);//是否需要录制静音，只对麦克风音频有效，请在录制开始前调用
```

3.1.8释放资源

在Activity销毁时，我们也需要释放录制资源，在destroy中调用：

```
void destroy();
AliyunRecorderCreator.destroyRecorderInstance();
```

3.1.9回调接口

录制功能提供了一些回调接口：录制回调

```
public interface RecordCallback {
/**
* 录制完毕的回调
*/
void onComplete(boolean validClip,long clipDuration);

/**
* 合成完毕的回调
* @param outputPath
*/
void onFinish(String outputPath);

/**
* 录制进度回调
* @param duration 当前已录制时间
*/
void onProgress(long duration);

/**
* 达到最大时长
```

```
*/
void onMaxDuration();

/**
* 录制错误回调
* @param errorCode
*/
void onError(int errorCode);

/**
* 录制初始化回调
*/
void onInitReady();

/**
* 该回调后可以调用{@link AliyunIRecorder#startRecording()}接口
*/
void onDrawReady();

/**
* 获取当前渲染帧并转成bitmap
* @param bitmap
*/
void onPictureBack(Bitmap bitmap);

/**
* 获取当前渲染帧数据
* @param data
*/
void onPictureDataBack(byte[] data);
}
```

采集帧数据回调

```
public interface OnFrameCallBack {
/**
* 采集帧回调，每采集一帧会通过该接口返回帧数据
* @param bytes
* @param width
* @param height
* @param info
*/
void onFrameBack(byte[] bytes,int width,int height,Camera.CameraInfo info);

/**
* 选择预览分辨率的回调
* @param supportedPreviewSizes
* @return
*/
Camera.Size onChoosePreviewSize(List<Camera.Size> supportedPreviewSizes, Camera.Size
preferredPreviewSizeForVideo);

/**
* 摄像头开启失败
*/
```

```
void openFailed();
}
```

纹理ID回调

```
public interface OnTextureIdCallBack {
/**
* 采集后，渲染处理前的纹理回调
* @param textureId
* @param textureWidth
* @param textureHeight
* @param matrix
* @return
*/
int onTextureIdBack(int textureId, int textureWidth, int textureHeight, float[] matrix);

/**
* 经过旋转、裁剪、缩放处理后的纹理回调
* @param scaledId
* @param textureWidth
* @param textureHeight
* @param matrix
* @return
*/
int onScaledIdBack(int scaledId,int textureWidth,int textureHeight,float[] matrix);
}
```

3.1.10前置特效

3.1.10.1 特效操作

录制过程中可以添加效果,这些效果目前包含滤镜，水印，人脸动图三种类型。效果的基类是EffectBase类。三种类型具体对应如下类：EffectFilter—滤镜EffectPaster—贴图EffectImage —静态图片（水印）添加效果的接口：

```
int addPaster(EffectBase effectBase); // 添加动图效果
int addImage(EffectBase effectBase); // 添加水印效果
int applyFilter(EffectBase effectBase);//添加滤镜效果,如果不需要滤镜，可以将参数置空传入
```

设置特效信息

```
void setEffectView(float xRatio,float yRatio,float widthRatio,float heightRatio,EffectBase effectBase);//设置图片的信息（位置，尺寸），其中xRatio，yRatio表示其起始坐标在屏幕中的相对位置百分比，widthRatio，heightRatio表示图片宽高和屏幕宽高的比例值
```

设置人脸坐标接口：

```
int setFaces(float[][] faces);//faces参数是包含多个face参数（含义参上）的二维数组，主要用于多人脸追踪，该接口可在采集帧数据回调中对帧数据做人脸识别，识别出人脸坐标后通过该接口传入到SDK内部。
```

移除效果的接口：

```
void removePaster(EffectBase effectBase); //删除动图效果
void removeImage(EffectBase effectBase); //删除水印效果
```

3.1.10.2 控制接口

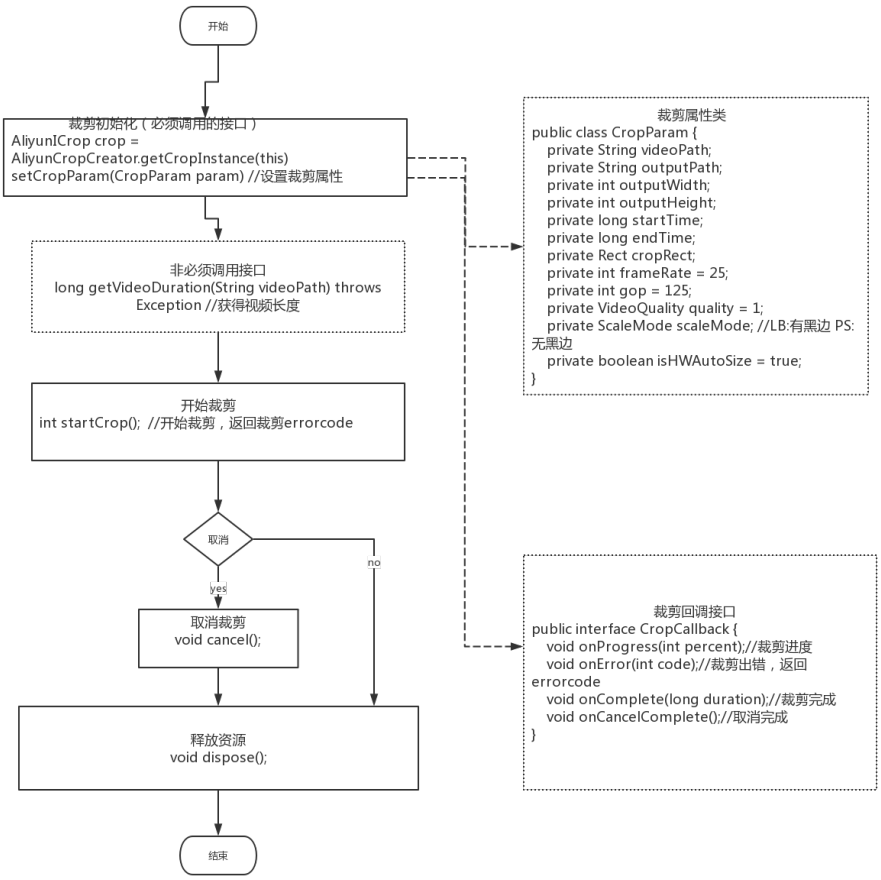
```
int switchCamera();//切换摄像头
FlashType switchLight();//循环切换闪光灯模式
boolean setLight(FlashType flashType);//切换为指定闪光灯模式
void setBeautyLevel(int level);//设置美颜度
void setBeautyStatus(boolean on);//设置美颜开 / 关，true：表示开，false：表示关
void setRotation(int rotation);//设置视频旋转度
void setVideoQuality(VideoQuality quality);//设置视频录制质量
void setVideoBitrate(int bitrate);//设置视频码率，如果不设置则使用视频质量videoQulity参数计算出码率
```

3.1.10.3 其他接口

```
int getBeautyLevel();//获取当前美颜级别
int getCameraCount();//获取摄像头数量
float getCurrentExposureCompensationRatio();//获取当前曝光度的比例值
String version();//获取接口版本号
```

3.2 导入裁剪

裁剪模块功能接口调用顺序图：



3.2.1 初始化参数

使用裁剪功能，需要初始化AliyunICrop对象，初始化方法如下：

```
AliyunICrop crop = AliyunCropCreator.getCropInstance(Context context);//其中context为当前页面的上下文
```

需要设置裁剪参数：

```
void setCropParam(CropParam param);
```

CropParam为裁剪参数类，如下：

```
public class CropParam {
private String videoPath; //源视频地址
private String outputPath; //输出视频地址
private int outputWidth; //输出宽
private int outputHeight; //输出高
private long startTime; //开始时间
private long endTime; //结束时间
private Rect cropRect; //裁剪区域（cropMode 为CropMode.LB时该参数无效）
private int frameRate = 25; //视频帧率
private int gop = 125; //关键帧间隔
private int mVideoBitrate; //视频码率
private VideoQuality quality = VideoQuality.HD; //视频质量 VideoQuality.SSD:极高 VideoQuality.HD:高
VideoQuality.SD:中 VideoQuality.LD:低 VideoQuality.PD:非常低 VideoQuality.EPD:极低
```

```
private ScaleMode scaleMode; //裁剪模式 ScaleMode.LB:有黑边 ScaleMode.PS:无黑边
private boolean isHWAutoSize = true; //这个变量如果为true（默认为true）则在硬编模式（目前仅支持硬编）下，如果宽/高不是16的倍数，则会自动矫正为16的倍数，这样会导致设置的宽/高与视频实际输出宽/高会有出入，表现出来的效果就是有黑边；如果false，在宽/高不是16的倍数时，则调用setCropParam时将抛出IllegalArgumentException，需要开发者自行处理该异常。
private MediaType mMediaType = MediaType.ANY_VIDEO_TYPE;//媒体文件类型，默认为视频，该参数必须填写，目前支持视频：ANY_VIDEO_TYPE 图片：ANY_IMAGE_TYPE 和 音频：ANY_AUDIO_TYPE
private int mFilmFillColorlColor = Color.BLACK;//填充模式时的填充底色
private boolean isUseGPU = false;//是否使用GPU
}
```

3.2.2 开始裁剪

开始裁剪，接口如下：

```
int startCrop();
```

3.2.3 取消裁剪

```
void cancel();
```

3.2.4 释放资源

```
void dispose();
```

3.2.5 获取视频时长

```
long getVideoDuration(String videoPath) throws Exception;
```

3.2.6裁剪音频文件

```
int startCropAudio(String inputPath,String outputPath,long startTime,long endTime);//inputPath: 音频文件地址 outputPath:裁剪输出地址 startTime:音频开始时间 endTime:音频结束时间
```

3.2.7 获取接口版本号

```
String version();
```

四、上传

SDK生成的mp4文件，可接入阿里云上传SDK上传文件，可跳转至阿里云上传SDK下载。点击Android上传SDK获取使用文档。

一、快速开始

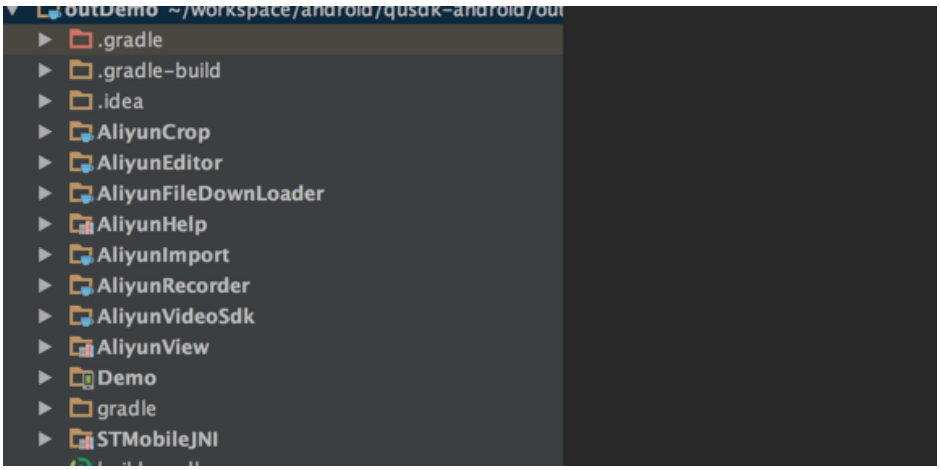
1.1 开发环境配置

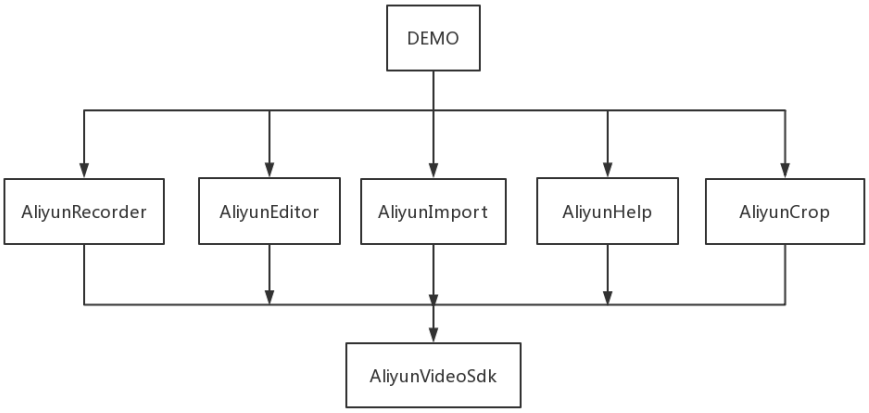
本SDK开发环境为 JAVA1.7 | ANDROID SDK API LEVEL 18

1.2 导入SDK

SDK包括DEMO,AliyunCrop,AliyunRecorder,AliyunEditor,AliyunImport,AliyunVideoSdk,AliyunHelp7个基本moudle，还可能包含其他人脸识别的第三方库，使用时只需要将整个根文件夹作为ANDROID STUDIO的项目导入即可。其中DEMO是作为整个SDK的模块选择界面，AliyunCrop是裁剪模块的示例代码，AliyunRecorder是魔法相机模块的示例代码，AliyunEditor是视频编辑模块的示例代码，AliyunImport是导入模块的示例代码，AliyunHelp是其他模块的示例代码，AliyunVideoSdk是SDK提供的接口。依赖关系为：DEMO依赖AliyunCrop，AliyunRecorder，AliyunEditor,AliyunImport和AliyunHelp。AliyunCrop,AliyunRecorder,AliyunEditor和AliyunImport依赖AliyunVideoSdk。

SDK的文件结构如图所示：





SDK的依赖结构如图所示：

1.3 权限要求

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

```
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Android 6.0以上系统需要做动态权限请求。

1.4 加载动态链接库

请将SDK解压包中的*jniLibs*目录拷贝到app module的*main*目录中，并且在app module的build.gradle文件中声明jniLibs的路径，如下:

```
android {
sourceSets.main {
jni.srcDirs = []
jniLibs.srcDir "src/main/jniLibs"
}
}
```

在app的Application文件中加载动态链接库，库文件说明如下：

```
libaliface_jni.so---------------人脸识别相关的库（不需要人脸识别功能可不加载）
libAliFaceAlignmentModule.so---人脸识别相关的库（不需要人脸识别功能可不加载）
libencoder.so------------------编码相关的库
libopenh264.so-----------------编码相关的库
libQuCore.so-------------------SDK核心库
libQuCore-ThirdParty.so--------SDK依赖的第三方库
```

在App自定义Application类的onCreate方法里面执行动态库加载：

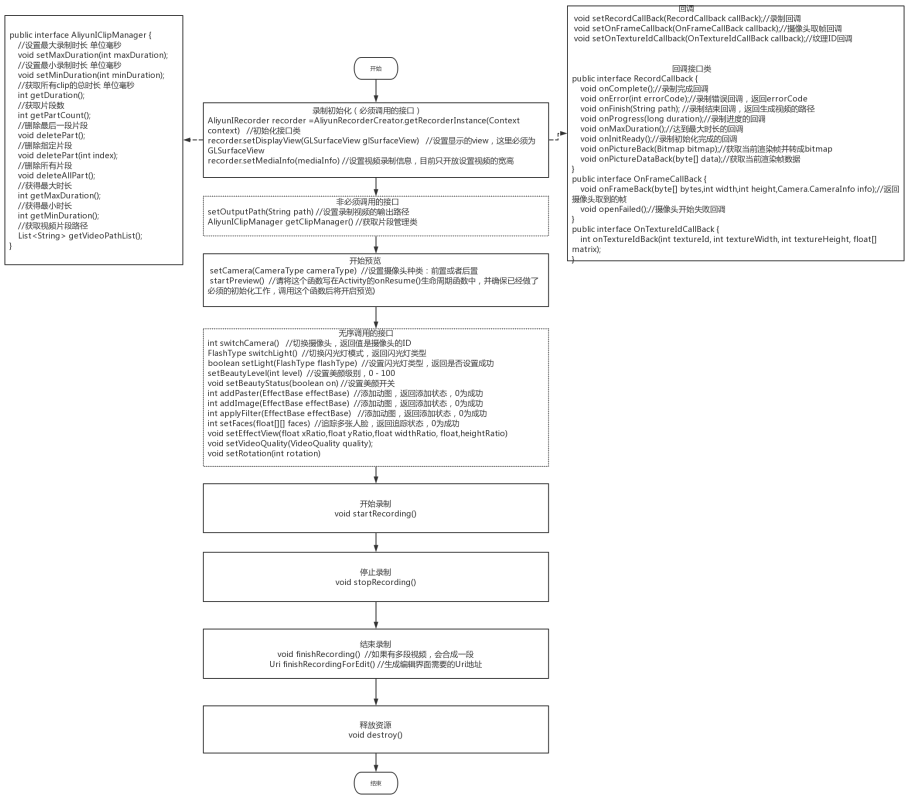
```
System.loadLibrary("openh264");
System.loadLibrary("encoder");
System.loadLibrary("AliFaceAlignmentModule");
System.loadLibrary("aliface_jni");
System.loadLibrary("QuCore-ThirdParty");
System.loadLibrary("QuCore");
```

二、功能使用

2.1 功能分布图

2.2 录制

录制模块功能接口调用顺序图：



2.2.1初始化参数

使用录制功能，需要初始化AliyunIRecorder对象，初始化方法如下：

```
AliyunIRecorder recorder = AliyunRecorderCreator.getRecorderInstance(Context context);//参数context为当前页面的上下文


recorder.setDisplayView(GLSurfaceView glSurfaceView);//参数glsurfaceView为用户自己定义的GLSurfaceView及其子类对象
```

设置视频参数信息：

```
MediaInfo mediaInfo = new MediaInfo();
mediaInfo.setVideoWidth(TEST_VIDEO_WIDTH);
mediaInfo.setVideoHeight(TEST_VIDEO_HEIGHT);
mediaInfo.setAutoAdjustHWEncoder(true);//硬编时自适应宽高为16的倍数
recorder.setMediaInfo(mediaInfo);
```

注意：录制时的分辨率是按照GLSurfaceView的实际宽高比，以设置的分辨率的宽为基准生成的，比如用户设置的分辨率是480x480，如果GLSurfaceView的宽高比为9:16，则录制的视频分辨率为480:854，如果想要保持480x480的分辨率，则GLSurfaceView的宽高比要保持1:1。硬编自适应开关如果设为true，则使用硬编（目前只支持硬编）时，如果宽 / 高不是16的倍数，则会强制转换为16的倍数，这样有一个影响就是传入的分辨率与实际输出分辨率会有一点出入。而如果设置为false，则如果宽 / 高不是16的倍数，在调用setMediaInfo时将抛出IllegalArgumentException。

2.2.2开始预览&结束预览开始预览如下方法：

```
void setCamera(CameraType camera); //设置预览的摄像头，前置或者后置
void startPreview();//请将这个函数写在Activity的onResume生命周期中，调用这个函数后将开始预览
```

结束预览如下方法：

```
void stopPreview();//请将这个函数写在Activity的onPause生命周期中，调用这个函数后将结束预览
```

2.2.3开始录制&结束录制

在录制之前，确保已设置录制文件的输出路径，通过AliyunIRecorder的outputPath属性进行设置。

设置传感器角度值，这个必须设置，否则录制的视频角度是错的。

```
void setRotation(int rotation);
```

开始录制接口：

```
void startRecording();
```

结束录制接口：

```
void stopRecording();
```

完成录制接口：

```
void finishRecording(); //调用完成录制的接口会合并录制的多段视频
```

生成录制片段信息路径接口（录制的多段视频会序列化到一个json描述文件中，该接口可以返回描述文件的地址，用于编辑等操作）

```
void finishRecordingForEdit(); //调用接口会生成一个配置文件的Uri
```

取消录制接口：

```
void cancelRecording();
```

2.2.4拍照

```
/**
* 拍照（从GPU中抓取数据，可以包含特效）
* @param needBitmap 是否需要生成Bitmap
*/
void takePhoto(boolean needBitmap);
```

```
/**
* 使用系统的拍照接口（不能包含特效）
* @param needBitmap 是否需要生成Bitmap
*/
void takePicture(boolean needBitmap);
```

2.2.5录制变速

```
void setRate(float rate);//录制变速，速率在0.5-2.0之间
```

2.2.6录制配乐

```
void setMusic(String path,long startTime,long duration);//path:音乐文件路径(建议使用aac格式的音频) startTime:音乐开始时间 duration:音乐时长
```

2.2.7录制静音

```
void setMute(boolean isMute);//是否需要录制静音，只对麦克风音频有效，请在录制开始前调用
```

2.2.8开启内置人脸追踪

```
void needFaceTrackInternal(boolean need);//开启内置人脸
```

2.2.9设置人脸模型文件夹路径

```
void setFaceTrackInternalModelPath(String path);//设置内置人脸模型文件夹路径，请将所有人脸模型文件放在同一个文件夹中，文件模型在录制模块demo的assets文件夹中
```

2.2.10释放资源

在Activity销毁时，我们也需要释放录制资源，在destroy中调用：

```
void destroy();
AliyunRecorderCreator.destroyRecorderInstance();
```

2.2.11回调接口

录制功能提供了一些回调接口：录制回调

```
public interface RecordCallback {
/**
* 录制完毕的回调
*/
void onComplete(boolean validClip,long clipDuration);

/**
* 合成完毕的回调
```

```
* @param outputPath
*/
void onFinish(String outputPath);

/**
* 录制进度回调
* @param duration 当前已录制时间
*/
void onProgress(long duration);

/**
* 达到最大时长
*/
void onMaxDuration();

/**
* 录制错误回调
* @param errorCode
*/
void onError(int errorCode);

/**
* 录制初始化回调
*/
void onInitReady();

/**
* 该回调后可以调用{@link AliyunIRecorder#startRecording()}接口
*/
void onDrawReady();

/**
* 获取当前渲染帧并转成bitmap
* @param bitmap
*/
void onPictureBack(Bitmap bitmap);

/**
* 获取当前渲染帧数据
* @param data
*/
void onPictureDataBack(byte[] data);
}
```

采集帧数据回调

```
public interface OnFrameCallBack {
/**
* 采集帧回调，每采集一帧会通过该接口返回帧数据
* @param bytes
* @param width
* @param height
* @param info
*/
void onFrameBack(byte[] bytes,int width,int height,Camera.CameraInfo info);
```

```
/**
* 选择预览分辨率的回调
* @param supportedPreviewSizes
* @return
*/
Camera.Size onChoosePreviewSize(List<Camera.Size> supportedPreviewSizes, Camera.Size
preferredPreviewSizeForVideo);

/**
* 摄像头开启失败
*/
void openFailed();
}
```

纹理ID回调

```
public interface OnTextureIdCallBack {
/**
* 采集后，渲染处理前的纹理回调
* @param textureId
* @param textureWidth
* @param textureHeight
* @param matrix
* @return
*/
int onTextureIdBack(int textureId, int textureWidth, int textureHeight, float[] matrix);

/**
* 经过旋转、裁剪、缩放处理后的纹理回调
* @param scaledId
* @param textureWidth
* @param textureHeight
* @param matrix
* @return
*/
int onScaledIdBack(int scaledId,int textureWidth,int textureHeight,float[] matrix);
}
```

2.2.12前置特效

2.2.12.1 特效操作

录制过程中可以添加效果,这些效果目前包含滤镜，水印，人脸动图三种类型。效果的基类是EffectBase类。三种类型具体对应如下类：EffectFilter—滤镜EffectPaster—贴图EffectImage —静态图片（水印）添加效果的接口：

```
int addPaster(EffectBase effectBase); // 添加动图效果
int addImage(EffectBase effectBase); // 添加水印效果
int applyFilter(EffectBase effectBase);//添加滤镜效果,如果不需要滤镜，可以将参数置空传入
```

设置特效信息

```
void setEffectView(float xRatio,float yRatio,float widthRatio,float heightRatio,EffectBase effectBase);//设置图片的信息（位置，尺寸），其中xRatio，yRatio表示其起始坐标在屏幕中的相对位置百分比，widthRatio，heightRatio表示图片宽高和屏幕宽高的比例值
```

设置人脸坐标接口：

```
int setFaces(float[][] faces);//faces参数是包含多个face参数（含义参上）的二维数组，主要用于多人脸追踪，该接口可在采集帧数据回调中对帧数据做人脸识别，识别出人脸坐标后通过该接口传入到SDK内部。
```

移除效果的接口：

```
void removePaster(EffectBase effectBase); //删除动图效果
void removeImage(EffectBase effectBase); //删除水印效果
```

2.2.12.2 控制接口

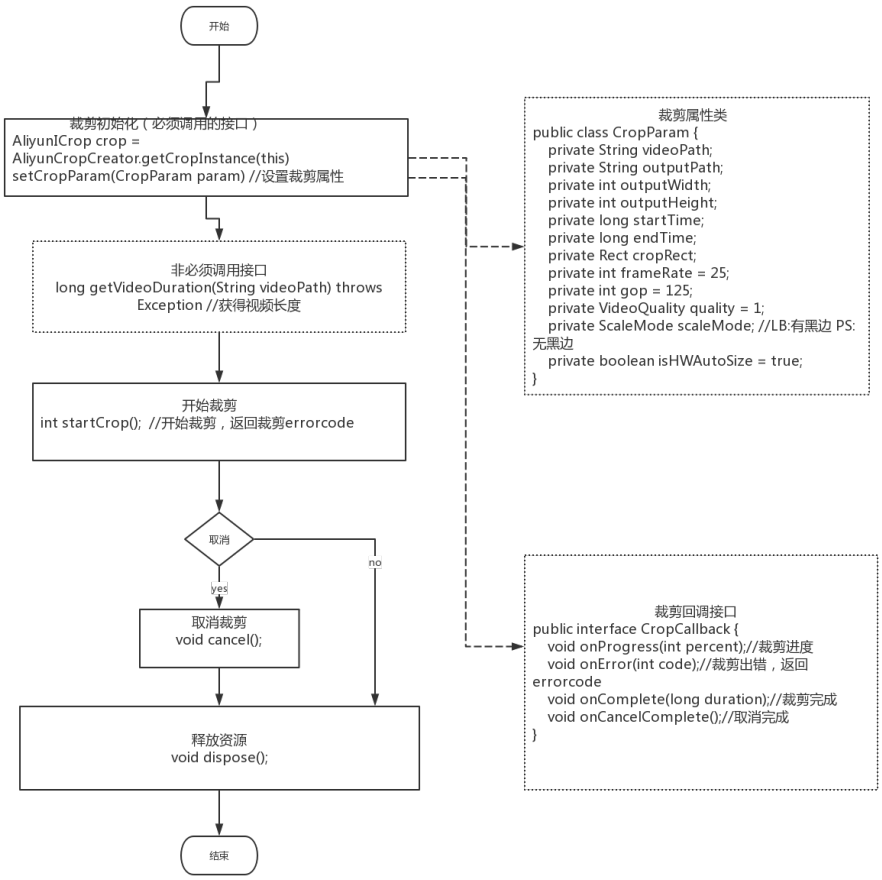
```
int switchCamera();//切换摄像头
FlashType switchLight();//循环切换闪光灯模式
boolean setLight(FlashType flashType);//切换为指定闪光灯模式
void setBeautyLevel(int level);//设置美颜度
void setBeautyStatus(boolean on);//设置美颜开 / 关，true：表示开，false：表示关
void setRotation(int rotation);//设置视频旋转度
void setVideoQuality(VideoQuality quality);//设置视频录制质量
void setVideoBitrate(int bitrate);//设置视频码率，如果不设置则使用视频质量videoQulity参数计算出码率
void setFocusMode(int mode);//设置对焦模式
void setFocus(float xRatio, float yRatio);//设置对焦点
```

2.4.12.3 其他接口

```
int getBeautyLevel();//获取当前美颜级别
int getCameraCount();//获取摄像头数量
float getCurrentExposureCompensationRatio();//获取当前曝光度的比例值
String version();//获取接口版本号
```

2.3 导入裁剪

裁剪模块功能接口调用顺序图：



2.3.1 初始化参数

使用裁剪功能，需要初始化AliyunICrop对象，初始化方法如下：

```
AliyunICrop crop = AliyunCropCreator.getCropInstance(Context context);//其中context为当前页面的上下文
```

需要设置裁剪参数：

```
void setCropParam(CropParam param);
```

CropParam为裁剪参数类，如下：

```
public class CropParam {
private String videoPath; //源视频地址
private String outputPath; //输出视频地址
private int outputWidth; //输出宽
private int outputHeight; //输出高
private long startTime; //开始时间
private long endTime; //结束时间
private Rect cropRect; //裁剪区域（cropMode 为CropMode.LB时该参数无效）
private int frameRate = 25; //视频帧率
private int gop = 125; //关键帧间隔
private int mVideoBitrate; //视频码率
private VideoQuality quality = VideoQuality.HD; //视频质量 VideoQuality.SSD:极高 VideoQuality.HD:高
VideoQuality.SD:中 VideoQuality.LD:低 VideoQuality.PD:非常低 VideoQuality.EPD:极低
```

```
private ScaleMode scaleMode; //裁剪模式 ScaleMode.LB:有黑边 ScaleMode.PS:无黑边
private boolean isHWAutoSize = true; //这个变量如果为true（默认为true）则在硬编模式（目前仅支持硬编）下，如果宽/高不是16的倍数，则会自动矫正为16的倍数，这样会导致设置的宽/高与视频实际输出宽/高会有出入，表现出来的效果就是有黑边；如果false，在宽/高不是16的倍数时，则调用setCropParam时将抛出IllegalArgumentException，需要开发者自行处理该异常。
private MediaType mMediaType = MediaType.ANY_VIDEO_TYPE;//媒体文件类型，默认为视频，该参数必须填写，目前支持视频：ANY_VIDEO_TYPE 图片：ANY_IMAGE_TYPE 和 音频：ANY_AUDIO_TYPE
private int mFilmFillColorlColor = Color.BLACK;//填充模式时的填充底色
private boolean isUseGPU = false;//是否使用GPU
}
```

2.3.2 开始裁剪

开始裁剪，接口如下：

```
int startCrop();
```

2.3.3 取消裁剪

```
void cancel();
```

2.3.4 释放资源

```
void dispose();
```

2.3.5 获取视频时长

```
long getVideoDuration(String videoPath) throws Exception;
```

2.3.6裁剪音频文件

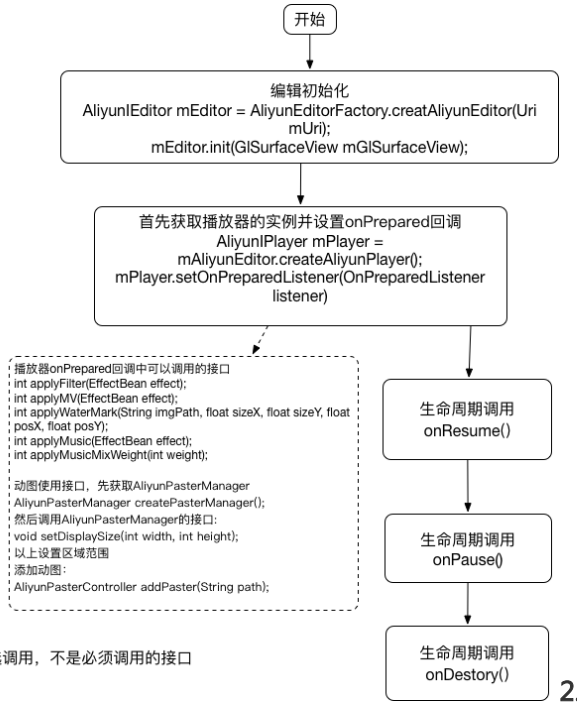
```
int startCropAudio(String inputPath,String outputPath,long startTime,long endTime);//inputPath: 音频文件地址 outputPath:裁剪输出地址 startTime:音频开始时间 endTime:音频结束时间
```

2.3.7 获取接口版本号

```
String version();
```

2.4 编辑

```
导出调用，首先获取导出接口实例
AliyunIExporter exporter = mEditor.getExporter();
exporter.setTailWatermark("/sdcard/QUEditorDemo/tail/
qupai-logo.png",
                0.5f, 0.5f, 0, 0);
exporter.setOutputVideoParam(mVideoParam);
final long totalDuration = mQuiPlayer.getDuration();
exporter.startCompose("/mnt/sdcard/outputVideo.mp4", new
OnComposeCallback() {
    @Override
    public void onError() {
        mIsComposing = false;
        dialog.dismiss();
        Toast.makeText(getApplicationContext(), "合成失
败",Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onComplete() {
        dialog.dismiss();
        if(mIsComposing) {
            Toast.makeText(getApplicationContext(), "合成完
成", Toast.LENGTH_SHORT).show();
        }
        mPlayer.start();
        mTimelineBar.resume();
        mIsComposing = false;
    }

    @Override
    public void onProgress(int progress) {
        dialog.setProgress(progress);
    }
});
```



2.4.1初始化参数

使用编辑功能，需要初始化AliyunIEditor对象，初始化方法如下：

```
AliyunIEditor mAliyunEditor = AliyunEditorFactory.creatAliyunEditor(uri);//参数uri为传入的视频片段描述文件地址（该文件可以从录制或者多段导入接口获取）
boolean init(SurfaceView surfaceView);//初始化编辑器
```

2.4.2视频预览播放

视频预览需要获取播放器，调用如下接口获取一个播放器的实例（注意【视频预览播放】模块的接口如没有特殊说明都是AliyunIPlayer中的接口）：

2.4.2.1开始播放

```
AliyunIPlayer createAliyunPlayer();//创建播放器实例（播放器实例为非线程安全，如需多线程环境使用，需要开发者自己保证线程安全。该接口来自于AliyunIEditor）
void setOnPreparedListener(OnPreparedListener listener);//设置播放器prepare的回调，在播放器onPrepared的回调中调用start()方法播放视频
void start();//开始播视频，该方法务必在onPrepared回调中调用。
```

关于onPrepared回调，编辑界面特效使用都必须在该回调之后才能使用，在该回调之前使用会有异常。

2.4.2.2停止播放

停止播放调用如下方法：

```
void stop();//调用这个函数后将停止视频播放
```

2.4.2.3 seek

```
/**
* seek到某个时间点
* @param time 时间，单位：毫秒
* @param resumeRequest seek后是否播放 true：播放，false：不播放
*/
void seek(long time, boolean resumeRequest);
```

2.4.2.4 设置播放回调

```
void setOnPlayCallbackListener(OnPlayCallback listener);
interface OnPlayCallback {
void onPlayStarted();//播放开始时回调
void onError();//播放出错时回调
void onPlayCompleted();//播放完成时回调
}
```

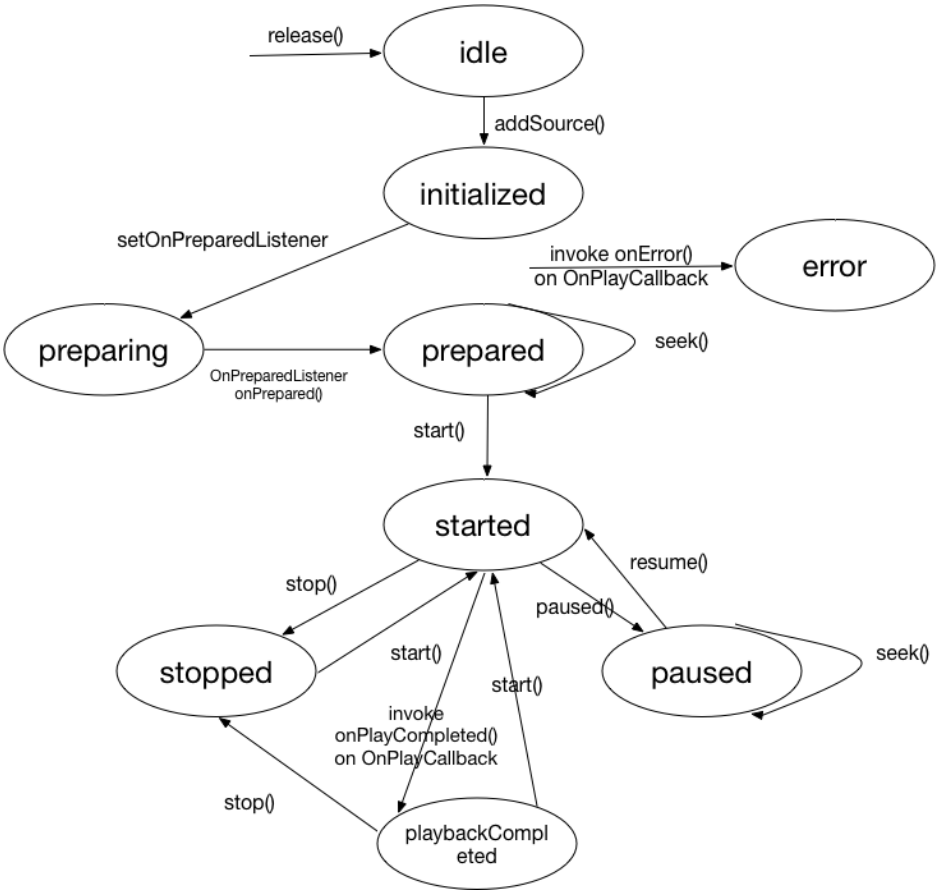
播放器回调都会post到主线程处理，播放器接口最好也在主线程调用。

2.4.2.5 暂停&恢复

```
void pause();//暂停播放
void resume();//恢复播放
```

2.4.2.6 播放器其他功能

```
long getCurrentPosition();//获取当前视频进度
long getDuration();//获取视频总时长
boolean isPlaying();//获取视频状态，是否在播放中
int getVideoWidth();//获取原始视频的宽度，如果有多段视频则取第一段视频的宽度
int getVideoHeight();//获取原始视频的高度，如果有多段视频则取第一段视频的高度
boolean isAudioSilence();//判断视频是否处于静音状态
void setAudioSilence(boolean silence);//设置视频静音或解除静音
void setVolume(int volume);//设置视频音量
void setDisplayMode(VideoDisplayMode mode);//设置视频填充模式 mode取值VideoDisplayMode.FILL 截断填充，铺满展示区域，多出的部分裁掉，V ideoDisplayMode.SCALE 黑边填充按视频比例全部展示到展示区域，无法铺满的 部分填充黑边
void setFillBackgroundColor(int color);//设置填充颜色，只有在{@link AliyunIPlayer#setDisplayMode(VideoDisplayMode)}为{@link VideoDisplayMode#FILL}时才有效
```



播放器状态及生命周期图：

2.4.3 特效编辑

使用特效之前确保播放器已经就绪，处于onPrepared的状态。可以使用的特效主要有滤镜，贴纸，字幕，音乐，MV，水印。其中贴纸，字幕有交互需求，SDK提供了两种使用方式。除贴纸，字幕外的其他特效均使用AliyunIEditor接口添加滤镜，音乐，MV三种特效均使用特效类EffectBean

2.4.3.1 应用滤镜特效

```
/**
* 使用滤镜效果
* @param effect 特效资源的路径及ID
* @return EFFECTNOTPAY,
* EFFECTUSEFAILED，
* EFFECTUSESUCCESS
*/
int applyFilter(EffectBean effect);
```

2.4.3.2 应用MV特效

```
/**
* 使用mv效果
* @param effect 特效资源的路径及ID
* @return EFFECTNOTPAY,
* EFFECTUSEFAILED，
* EFFECTUSESUCCESS
```

```
*/
int applyMV(EffectBean effect);
```

2.4.3.3 应用音乐特效

```
/**
* 使用music效果
* @param effect 特效资源的路径及ID
* @return EFFECTNOTPAY,
* EFFECTUSEFAILED，
* EFFECTUSESUCCESS
*/
int applyMusic(EffectBean effect);
```

2.4.3.2 应用水印特效

```
int applyWaterMark(String imgPath, float sizeX, float sizeY, float posX, float posY);//传入水印图片的路径，sizeX 水平方向上水印相对展示区域的比例，sizeY竖直方向上水印相对展示区域的比例，posX 水印位置在水平方向上相对展示区域的比例， posY 水印位置在竖直方向上相对展示区域的比例
```

水印接口调用示例

```
File waterMark = new File("/sdcard/QUEditorDemo/tail/qupai-logo2.png");
if (waterMark.exists()) {
Bitmap wmBitmap = BitmapFactory.decodeFile("/sdcard/QUEditorDemo/tail/qupai-logo2.png");
if (wmBitmap != null) {
//水印例子 水印的大小为 ：水印图片的宽高和显示区域的宽高比，注意保持图片的比例，不然显示不完全 水印的位置为 ：以水印图片中心点为基准，显示区域宽高的比例为偏移量，0,0为左上角，1,1为右下角
mAliyunEditor.applyWaterMark("/sdcard/QUEditorDemo/tail/qupai-logo2.png", (float)wmBitmap.getWidth() / mGlSurfaceView.getWidth(), (float) wmBitmap.getHeight() / mGlSurfaceView.getHeight(), 1f - (float) wmBitmap.getWidth() / mGlSurfaceView.getWidth() / 2, 0f + (float) wmBitmap.getHeight() / mGlSurfaceView.getHeight() / 2);
}
}
```

2.4.3.3 设置混音权重

```
int applyMusicMixWeight(int weight);
```

一般在使用了音乐或MV后调用，如果没有音乐资源则调用该接口无效。

2.4.3.4 贴纸&字幕&文字

贴纸和字幕拥有编辑可选项，与其他特效有所不同，可以有两种方法来操作：方法一：带交互逻辑的贴纸和字幕编辑操作使用AliyunPasterManager和AliyunPasterController完成。方法二：不带交互逻辑的贴纸和字幕编辑操作使用AliyunPasterRender接口完成。（实际上方法二内部也是使用的AliyunPasterRender）如果开发者想要使用默认编辑交互行为，则可以使用第一种带交互逻辑的接口来进行；如果第一种方法的交互行为无法满足开发者的需要，则可以使用第二种方法，使用AliyunPasterRender来进行贴纸和字幕的编辑，渲染工作。而实际的交互行为由开发者自己定义开发。

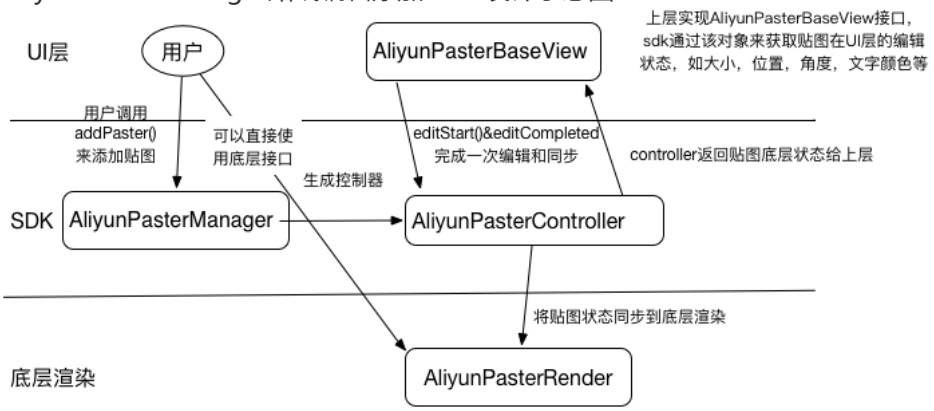
使用方法一：编辑需求的贴纸，字幕添加，首先获取AliyunPasterManager的实例：

```
AliyunPasterManager createPasterManager();//通过AliyunIEditor对象调用该方法获取
```

AliyunPasterManager的主要接口：

```
AliyunPasterController addPaster(String path);//添加普通贴纸
AliyunPasterController addPasterWithStartTime(String path, long startTime, long duration);//添加贴纸并传入时长和开始时间
AliyunPasterController addSubtitle(String text, String font);//添加文字贴纸，并传入文字所使用的字体文件路径
AliyunPasterController addSubtitleWithStartTime(String text, String font, long sta tTime, long duration);//添加文字贴纸，传入文字所使用的字体文件的路径，并传入开始时间和时长
void setDisplaySize(int width, int height);//设置贴纸展示区域的大小
void setOnPasterRestoreListener(OnPasterRestored listener);//设置贴纸恢复的回调
```

获取AliyunPasterManager对象后，需要设置动图展示区域的大小，如果不设置，默认为屏幕的宽为边长的正方形区域，设置后才能正确的计算出贴纸在展示区域内正确的大小和位置。使用贴纸后会返回AliyunPasterController对象，AliyunPasterController定义了贴纸的位置，大小等相关信息，可以通过get方法获取，它也提供了贴纸编辑的一些接口，比如移除贴纸，设置贴纸时间，通知贴纸开始编辑及结束编辑。
AliyunPasterManager贴纸编辑添加MVC设计示意图：



从上面的示意图中我们看到AliyunPasterManager负责创建控制器AliyunPasterController对象，控制器中定义了AliyunPasterBaseView接口来获取上层UI的状态，开发者可以实现AliyunPasterBaseView接口，并且通过该接口将UI交互行为产生的数据变动传递给控制器，控制器再同步UI状态到渲染层，从而完成了贴纸的编辑渲染，通过这种方式，开发者可以不需要关心内部业务细节而只关心UI交互。

使用方法二：无编辑需求的贴图使用，首先获取AliyunPasterRender接口：

```
AliyunPasterRender getPasterRender();//通过AliyunIEditor接口获取
```

获取到AliyunPasterRender对象后同样的需要设置展示区域的大小和贴图恢复和保存的回调：

```
void setDisplaySize(int width, int height);
void setOnPasterResumeAndSave(OnPasterResumeAndSave listener);
```

AlivunPasterRender的主要接口：

```
int addSubtitle(Bitmap bmp, EffectText subtitle);
int addEffectPaster(EffectPaster paster);
int addCaptionPaster(Bitmap bmp, EffectCaption caption);
```

EffectPaster为贴图的基类，EffectText扩展了EffectPaster，添加了文字相关的属性，EffectCaption又扩展了EffectPaster，添加了文字与贴图的关系描述。使用接口时需要自己构造贴图描述对象，主要描述有贴图的宽高，中心点位置，旋转角度，是否镜像，开始结束时间，名称，id等

删除隐藏及显示

```
void removePaster(EffectPaster paster);
void hidePaster(EffectPaster paster);
void showPaster(EffectPaster paster);
void showTextPaster(Bitmap bmp, EffectText text);
void showCaptionPaster(Bitmap bmp, EffectCaption caption);
```

2.4.3.5 涂鸦

涂鸦功能提供了涂鸦控制器*AliyunICanvasController*,通过涂鸦控制器，开发者可以拿到画板（View）和画笔，画板是一个View，开发者可以将其动态添加到布局中，画笔用来设置各种属性。

获取涂鸦控制器*AliyunICanvasController*

```
AliyunICanvasController obtainCanvasController(Context context, int w, int h);
```

以下涂鸦部分接口均来自*AliyunICanvasController*

```
View getCanvas();//获取涂鸦板
void removeCanvas();//清除渲染到视频的涂鸦
void undo();//撤销上一笔操作
void clear();//清空画板
void setPaint(Paint paint);//设置自定义画笔
void setCurrentColor(int color);//设置当前画笔颜色
void setCurrentSize(float size);//设置当前画笔粗细
int applyPaintCanvas();//设置涂鸦
int resetPaintCanvas();//重新渲染涂鸦,这里与resetPaintCanvas不同的是不需要重新生成中间文件，而是直接渲染，该接口必须在执行过applyPaintCanvas之后调用
```

2.4.4视频合成导出

获取导出实例，在AliyunIEditor中调用：

```
AliyunIExporter getExporter();
```

获取AliyunIExporter对象后设置片尾水印（可选），然后启动合成，传入合成回调及输出地址，中途可以取消合成。主要接口如下：

```
<ol><li> /** </li><li> * 添加片尾水印 </li><li> * @param imgPath 水印图片地址 </li><li> * @param sizeX 水印图片
```

```
横向大小 </li><li> * @param sizeY 水印图片纵向大小 </li><li> * @param posX 水印图片横向位置（-1，1） </li><li>
* @param posY 水印图片纵向位置（-1，1） </li><li> */ </li><li> void setTailWatermark ( String imgPath , float
sizeX , float sizeY , float posX , float posY );</li><li></li><li> /** </li><li> * 清除片尾水印 </li><li> */ </li><li>
void clearTailWatermark (); </li><li></li><li> /**</li><li> * 开始导出合成</li><li> * @param outputPath</li><li>
* @param callback</li><li> */</li><li> void startCompose ( String outputPath , OnComposeCallback callback );
</li><li></li><li> /**</li><li> * 取消导出合成</li><li> */</li><li> void cancel ();</li></ol>
```

调用导出后播放器的一些功能将不能使用，如start，stop，seek，pause，resume等，必须在导出完成或者取消后才可以继续调用。导出视频的码率由上一个界面传入的参数决定。

2.4.5导出回调接口

导出功能提供了一些回调接口：

导出回调

```
public interface OnComposeCallback {
void onError();
void onComplete();
void onProgress(int progress);
}
```

2.4.6其他接口

AliyunIEditor还提供了获取视频缩略图的工具接口AliyunIThumbnailFetcher，用于自定义视频导航条，其实例可以通过AliyunIEditor.getAliyunThumbnailFetcher方法获取，AliyunIThumbnailFetcher主要方法有：

```
首先添加视频源（支持多端视频）
void addVideoSource(String path);//添加视频源， path表示视频路径
void fromConfigJson(String jsonPath);//从配置文件加载视频信息（适用于SDK内操作产生的project.json）
```

设置取帧参数

```
/**
* 设置输出大小，该方法必须在{@link AliyunIThumbnailFetcher#requestThumbnailImage(long[],
OnThumbnailCompletion, int)}
* 前面调用，否则会抛出{@link IllegalStateException}
* @param width 输出宽度
* @param height 输出高度
* @param mode 裁剪模式（目前可以忽略，填任意值，所有的都是从中间裁剪）
* @param scaleMode 缩放模式（目前可以忽略，填任意值，只支持裁剪模式，不支持填充模式）
* @param cacheSize 缓存大小，即缓存的缩略图数量，缓存的图片不需要重新解码取
* @return
*/
int setParameters(int width, int height,
CropMode mode, ScaleMode scaleMode,
int cacheSize);
/**
* 通过请求id取消该id所标识的请求，该方法不保证一定成功，会返回取消成功的标识
* @param id 要取消的缩略图请求的ID
* @return true 取消成功， false 取消失败
*/
```

```
boolean cancelThumbnailRequest(int id);
/**
* 传入时间点获取该时间点的视频缩略图，并通过回调返回结果
* @param time 时间点，单位毫秒
* @param callback 获取缩略图结果的回调
* @return 返回该次请求的ID
*/
int requestThumbnailImage(long time, int width, int height, OnThumbnailCompl etion callback, int cacheSize);
/**
* 获取总时长，单位是毫秒
* @return
*/
long getTotalDuration();
```

缩略图获取的回调：

```
interface OnThumbnailCompletion {
void onThumbnailReady(ShareableBitmap frameBitmap, long time);
}
```

2.5 多段导入

2.5.1 初始化参数

使用导入编辑功能，需要初始化AliyunIImport对象，初始化方法如下：

```
AliyunImportCreator.getImportInstance(Context context);//其中context建议传入Application Context
```

设置视频输出参数：

```
void setVideoParam(AliyunVideoParam param);
```

AliyunVideoParam为视频输出参数类，如下：

```
public class AliyunVideoParam {
private int mFrameRate; //输出帧率
private int mGop; //关键帧间隔（单位为帧数）
private int mBitrate; //视频码率
private int mOutputWidth; //输出视频宽
private int mOutputHeight;//输出视频高
private VideoQuality mVideoQuality = VideoQuality.HD; //输出视频质量，分为极高，高，中，低
private ScaleMode mScaleMode = ScaleMode.LB; //裁切模式，LB为视频显示完全，多出的部分填充黑边，PS为不填充黑边，假如视频放不下则裁切视频以适应区域
private boolean isHWAutoSize = true;//这个变量如果为true（默认为true）则在硬编模式（目前仅支持硬编）下，如果宽/高不是16的倍数，则会自动矫正为16的倍数，这样会导致设置的宽/高与视频实际输出宽/高会有出入，表现出来的效果就是有黑边；如果false，在宽/高不是16的倍数时，则调用setVideoParam时将抛出IllegalArgumentException，需要开发者自行处理该异常。
}
```

2.5.2 添加要合成的视频添加视频，接口如下：

```
int addVideo(String videoPath, long fadeDuration, AliyunDisplayMode mode);
```

新增选择添加视频的时间区间，接口如下：

```
int addVideo(String videoPath, long startTime,long endTime,long fadeDuration, AliyunDisplayMode mode);
```

2.5.3 添加要合成的图片添加图片，接口如下：

```
int addImage(String imagePath, long fadeDuration, long duration, AliyunDisplayMode mode);
```

2.5.4 移除视频

```
void removeVideo(String videoPath);
```

2.5.5 更换视频顺序

```
void swap(int pos1, int pos2);
```

2.5.6 生成可编辑的配置

```
String generateProjectConfigure();
```

2.5 视频取帧工具.可取非关键帧

开发者可以通过*AliyunThumbnailFetcherFactory.createThumbnailFetcher()*获取一个AliyunIThumbnailFetcher的实例。目前跟编辑界面取帧工具是是个接口类.首先添加视频源（支持多端视频）

```
void addVideoSource(String path);//添加视频源， path表示视频路径
void fromConfigJson(String jsonPath);//从配置文件加载视频信息（适用于SDK内操作产生的project.json）
```

新增选择视频时间的接口

```
void addVideoSource(String path,long startTime,long endTime);
```

设置取帧参数

```
/**
* 设置输出大小，该方法必须在{@link AliyunIThumbnailFetcher#requestThumbnailImage(long[],
OnThumbnailCompletion, int)}
* 前面调用，否则会抛出{@link IllegalStateException}
* @param width 输出宽度
```

```
* @param height 输出高度
* @param mode 裁剪模式（目前可以忽略，填任意值，所有的都是从中间裁剪）
* @param scaleMode 缩放模式（目前可以忽略，填任意值，只支持裁剪模式，不支持填充模式）
* @param cacheSize 缓存大小，即缓存的缩略图数量，缓存的图片不需要重新解码取
* @return
*/
int setParameters(int width, int height,
CropMode mode, ScaleMode scaleMode,
int cacheSize);
```

其他接口

```
/**
* @param time 一组时间点，单位毫秒
* @param callback 获取缩略图结果的回调
* @return 返回该次请求的ID
*/
int requestThumbnailImage(long[] time, OnThumbnailCompletion callback, int cacheSize);
void release();//释放资源
long getTotalDuration();//获取总时长，单位是毫秒
```

三、常见问题

录制时预览画面拉伸变形的问题？答：录制视频的输出分辨率宽高比设置必须保证跟预览所用的GLSurfaceView宽高比一致，否则会自适应GLSurfaceView的宽高比，而导致视频拉伸/压缩变形，这样设计是为了保证所见即所得。

系统相册和SDK都无法读取视频缩略图且播放页不显示内容？答：缩略图获取使用的是系统API，当视频的色彩编码是4:4:4时，安卓系统API无法无法获取缩略图，且系统播放器也无法正常播放（DEMO中使用的播放器是系统的播放器）。

系统相册中显示的缩略图跟视频内容不符？答：部分手机系统数据库提供的缩略图是如此的，系统相册显示的缩略图是从系统数据库中获取的。

视频导入裁剪，缩略图无法获取到？答：目前使用系统API获取缩略图在部分机型上是无法正常获取的，该问题会在后续的版本中优化解决（SDK内部自己实现缩略图的获取替代系统API）

录制完成后，进入系统相册无法找到录制的视频？答：部分手机的SD卡增加多媒体文件后需要重启手机才会生成新的缩略图信息，如果遇到该问题，只需重启手机即可。

使用Demo工程打包的Apk，魔法相机的人脸识别不起作用？答：目前工程中使用的人脸识别库是商汤的人脸识别库，由于License已经过期，因此无法正常使用，之所以保留代码是为了给开发者示范如何接入人脸识别库到我们的SDK中，如果用户想要使用人脸识别功能请自行更换人脸识别库。如果用户想要试用带人脸识别的魔法相机功能，请安装官网下载目录中携带的apk安装包。

四、 特殊参数限制说明

参数名称	对应接口或字段	限制说明
视频输出分辨率	录制：MediaInfo.setVideoHeight和MediaInfo.setVideoWidth 裁剪：CropParam.setOutputHeight和CropParam.setOutputWidth 导入合成：AliyunVideoParam.setOutputWidth和AliyunVideoParam.setOutputHeight	录制、裁剪、合成的输出分辨率宽、高不能小于128
视频输出GOP大小	录制：AliyunIRecorder.setGop 裁剪：CropParam.setGop 导入合成：AliyunVideoParam.setGop	Gop大小不能小于帧率的2倍，否则SDK将会默认改为帧率的2倍，也就是Gop最小为2秒
硬编分辨率16数适配	录制：MediaInfo.setHWAutoSize 裁剪：CropParam.setHWAutoSize 导入合成：AliyunVideoParam.setHWAutoSize	开启硬编模式（目前只有硬编）后，视频输出分辨率必须为16倍数，如果isHWAutoSize设置为true，则SDK会对于非16倍数的宽 / 高自动做矫正，这样会导致最终输出的视频与开发者设置的分辨率不会完全匹配，会有一点黑边，如果isHWAutoSize为false，则对于非16倍数的宽 / 高，SDK将在对应的参数设置接口抛出IllegalArgumentsException的异常，由开发者自行处理该问题。
码率计算	裁剪或合成：videoBitrate码率	如果不设置码率或设置码率为0时，码率由视频质量参数来计算得到，如果设置码率则使用该码率参数来编码，码率由用户控制。

iOS短视频SDK

一、版本要求

iOS支持iOS8.0以上

二、开发环境配置

SDK开发环境为macOS Sierra 10.12.2 XCode8.0及以上

三、使用说明

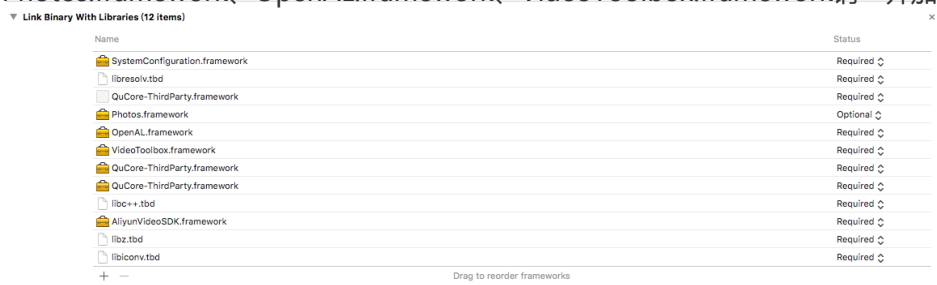
导入SDK

先下载SDK，详见SDK下载页面。
直接引入SDK的framework，名称为 AliyunVideoSDK

1. 开发者打开工程，选中目标target,依次选择”Build Phases”->”Link Binary With Libraries”,点击”+”号，点击“Add Other...”,导入AliyunVideoSDK.framework。

   开发者打开工程，选中目标target,依次选择”General”->”Embededed Binaries”,点击”+”号，点击“Add Other...”,导入QUCore-ThirdParty.framework。

   同时，还需依赖libz.tbd、libc++.tbd、libiconv.tbd、libresolv.tbd、ImageIO.framework、CoreMedia.framework、CoreVideo.framework、SystemConfiguration.framework、Photos.framework、OpenAL.framework、VideoToolbox.framework请一并加上。截图如下：



   配置 Build Setting -- Build Options -- Enable Bitcode 选项为NO

5. 配置 Build Setting -- Linking -- Other Linker Flags 添加 -ObjC选项

添加用户权限

打开工程的 info.Plist 文件,添加如下字段：

```
`Privacy - Camera Usage Description`、
`Privacy - Microphone Usage Description`、
`Privacy - Photo Library Usage Description`。
```

按需添加字段描述。

UI配置&整体配置

SDK的基础类为单例 AliyunVideoBase

- 4.1.1UI参数配置

UI配置类为 AliyunVideoUIConfig ，如果需要对UI界面的颜色和Button的隐藏等进行调整，只需要实例化一个AliyunVideoUIConfig类，并对如下属性进行赋值，各个属性参数如下

```
/* 背景颜色 */
@property (nonatomic, strong) UIColor *backgroundColor;
/* 录制进度条 已录制颜色 */
@property (nonatomic, strong) UIColor *timelineTintColor;
/* 录制进度条 背景颜色 */
@property (nonatomic, strong) UIColor *timelineBackgroundCollor;
/* 录制进度条 视频删除部分选中颜色 */
@property (nonatomic, strong) UIColor *timelineDeleteColor;
/* 录制时间提示框 字体颜色 */
@property (nonatomic, strong) UIColor *durationLabelTextColor;
/* 裁剪页裁剪条下边框颜色 */
@property (nonatomic, strong) UIColor *cutBottomLineColor;
/* 裁剪页裁剪条上边框颜色 */
@property (nonatomic, strong) UIColor *cutTopLineColor;


/* 隐藏已录制时间提示框 */
@property (nonatomic, assign) BOOL hiddenDurationLabel;
/* 隐藏美颜按钮 */
@property (nonatomic, assign) BOOL hiddenBeautyButton;
/* 隐藏录制按钮 */
@property (nonatomic, assign) BOOL hiddenCameraButton;
/* 隐藏闪光灯按钮 */
@property (nonatomic, assign) BOOL hiddenFlashButton;
/* 隐藏相册导入按钮 */
@property (nonatomic, assign) BOOL hiddenImportButton;
/* 隐藏删除视频片段按钮 */
@property (nonatomic, assign) BOOL hiddenDeleteButton;
/* 隐藏录制完成按钮 */
@property (nonatomic, assign) BOOL hiddenFinishButton;
/* 是否录制单段视频 */
@property (nonatomic, assign) BOOL recordOnePart;
/* 相册界面是否显示跳转相机按钮 */
@property (nonatomic, assign) BOOL showCameraButton;


/* 录制模式 */
@property (nonatomic, assign) AliyunVideoRecordType recordType;
/* 滤镜名称数组 注：请对应到滤镜bundle的各个滤镜文件名 */
@property (nonatomic, strong) NSArray<NSString *> *filterArray;


/* 录制切换为无滤镜选项时提示文字 */
@property (nonatomic, strong) NSString *noneFilterText;
```

```
/* 图片资源存放的BundleName */
@property (nonatomic, strong) NSString *imageBundleName;
/* 滤镜资源存放的BundleName */
@property (nonatomic, strong) NSString *filterBundleName;
```

对应表

属性	类型	释义
backgroundColor	UIColor	背景颜色
timelineTintColor	UIColor	录制进度条 已录制颜色
timelineBackgroundCollor	UIColor	录制进度条 背景颜色
timelineDeleteColor	UIColor	录制进度条 视频删除部分选中颜色
durationLabelTextColor	UIColor	录制时间提示框 字体颜色
hiddenDurationLabel	BOOL	隐藏已录制时间提示框
cutBottomLineColor	UIColor	裁剪页裁剪条下边框颜色
cutTopLineColor	UIColor	裁剪页裁剪条上边框颜色
hiddenCameraButton	BOOL	隐藏美颜按钮
hiddenFlashButton	BOOL	隐藏录制按钮
hiddenImportButton	BOOL	隐藏闪光灯按钮
hiddenDeleteButton	BOOL	隐藏相册导入按钮
hiddenFinishButton	BOOL	隐藏删除视频片段按钮
recordOnePart	BOOL	是否录制单段视频
showCameraButton	BOOL	相册界面是否显示跳转相机按钮
AliyunVideoRecordType	AliyunVideoRecordType	录制模式
filterArray	NSArray	滤镜名称数组
noneFilterText	NSString	录制切换为无滤镜选项时提示文字
imageBundleName	NSString	图片资源存放的BundleName
filterBundleName	NSString	滤镜资源存放的BundleName

注：filterArray中存放的滤镜名称必须与filterBundle中的文件名称对应。

UI配置注册

完成第一步设置好AliyunVideoUIConfig后需要调用AliyunVideoBase下述接口进行UI配置的注册

```
 /**
设置UI配置参数

@param videoUIConfig UI配置
*/
- (void)registerWithQUIConfig:(AliyunVideoUIConfig *)videoUIConfig;
```

示例：

```
 AliyunVideoUIConfig *config = [[AliyunVideoUIConfig alloc] init];
config.timelineDeleteColor = [UIColor redColor];
config.hiddenFlashButton = NO;
config.imageBundleName = @"image";
[[AliyunVideoBase shared] registerWithQUIConfig:config];
```

UI图片配置

如果开发者需要对SDK内部界面的UIButton等控件的图片进行修改，只需要将 上述AliyunVideoUIConfig的imageBundleName设置为开发者制作的Bundle即可。需要注意的是**bundle中各个image的名称需要与Demo中给出的image.bundle中的名称一一对应**

滤镜替换的原理相同，对应设置 AliyunVideoUIConfig的 filterBundleName即可。

录制

初始化参数

使用录制功能，需要初始化AliyunVideoRecordParam对象（录制视频参数配置类）

```
/* 摄像头方向 */
@property (nonatomic, assign) AliyunCameraPosition position;
/* 闪光灯模式 */
@property (nonatomic, assign) AliyunCameraTorchMode torchMode;
/* 美颜状态 */
@property (nonatomic, assign) BOOL beautifyStatus;
/* 设置美颜度 [0,100] */
@property (nonatomic, assign) int beautifyValue;
/* 输出路径 */
@property (nonatomic, strong) NSString *outputPath;
/* 视频分辨率 */
@property (nonatomic, assign) AliyunVideoSize size;
/* 视频比例 */
@property (nonatomic, assign) AliyunVideoRatio ratio;
/* 最小时长 */
@property (nonatomic, assign) CGFloat minDuration;
/* 最大时长 */
@property (nonatomic, assign) CGFloat maxDuration;
```

```
/* 视频质量 */
@property (nonatomic, assign) AliyunVideoQuality videoQuality;
/* 编码方式 */
@property (nonatomic, assign) AliyunVideoEncodeMode encodeMode;
/* 帧率 */
@property (nonatomic, assign) int fps;
/* 关键帧间隔 */
@property (nonatomic, assign) int gop;
/** 裁剪码率 */
@property (nonatomic, assign) int bitrate;
```

对应表

属性	类型	释义
position	AliyunCameraPosition	摄像头方向
torchMode	AliyunCameraTorchMode	闪光灯模式
beautifyStatus	BOOL	美颜状态
beautifyValue	int	美颜值
outputPath	NSString	输出路径
size	AliyunVideoSize	视频分辨率
ratio	AliyunVideoRatio	视频比例
minDuration	CGFloat	最小时长
maxDuration	CGFloat	最大时长
videoQuality	AliyunVideoQuality	视频质量
encodeMode	AliyunVideoEncodeMode	编码方式
fps	int	帧率
gop	int	关键帧间隔

可以通过调用初始化方法进行初始化

```
/**
创建AliyunVideoRecordParam
*/
+ (instancetype)recordConfigWithVideoRatio:(AliyunVideoRatio)ratio
videoSize:(AliyunVideoSize)size;


/**
创建AliyunVideoRecordParam
*/
+ (instancetype)recordConfigWithVideoRatio:(AliyunVideoRatio)ratio
videoSize:(AliyunVideoSize)size
position:(AliyunCameraPosition)position
trochMode:(AliyunCameraTorchMode)torchMode
beautifyStatus:(BOOL)beautifyStatus
```

```
beautifyValue:(int)beautifyValue
outputPath:(NSString *)outputPath
minDuration:(CGFloat)minDuration
maxDuration:(CGFloat)maxDuration
videoQuality:(AliyunVideoQuality)videoQuality
encodeMode:(AliyunVideoEncodeMode)encodeMode
fps:(int)fps
gop:(int)gop;
```

- **录制界面创建**

完成第一步设置好AliyunVideoRecordParam后需要调用AliyunVideoBase下述接口创建录制Controller

```
 /**
创建一个录制界面

@param recordParam 录制视频参数
@return 录制界面
*/
- (UIViewController *)createRecordViewControllerWithRecordParam:(AliyunVideoRecordParam *)recordParam;
```

示例：

```
 UIViewController *recordViewController = [[AliyunVideoBase shared]
createRecordViewControllerWithRecordParam:self.quVideo];
[AliyunVideoBase shared].delegate = self;
[self.navigationController pushViewController:recordViewController animated:YES];
```

注：由于SDK内部使用的push跳转，所以外部在present调起录制界面时，需要加一层UINavigationViewController在进行present。裁剪页面同理

录制界面回调

设置AliyunVideoBase的delegate即可接收 录制完成 退出录制 退出录制等回调

```
 /**
录制完成回调

@param base AliyunVideoBase
@param recordVC 录制界面VC
@param videoPath 视频保存路径
*/
- (void)videoBase:(AliyunVideoBase *)base recordCompeleteWithRecordViewController:(UIViewController
*)recordVC videoPath:(NSString *)videoPath;

/**
退出录制
*/
- (void)videoBaseRecordVideoExit;
```

```
/**
录制页跳转相册页

@param recordVC 当前录制页VC
@return 跳转相册页的视频配置，若返回空则沿用录制页的视频配置
*/
- (AliyunVideoCropParam *)videoBaseRecordViewShowLibrary:(UIViewController *)recordVC;
```

导入裁剪

- 初始化参数

使用裁剪功能，需要初始化AliyunVideoRecordParam对象（录制视频参数配置类）

```
 /* 输出路径 */
@property (nonatomic, strong) NSString *outputPath;
/* 输出视频分辨率 */
@property (nonatomic, assign) AliyunVideoSize size;
/* 输出视频比例 */
@property (nonatomic, assign) AliyunVideoRatio ratio;
/* 过滤相册视频最小时长 */
@property (nonatomic, assign) CGFloat minDuration;
/* 过滤相册视频最大时长 */
@property (nonatomic, assign) CGFloat maxDuration;
/* 裁剪模式 */
@property (nonatomic, assign) AliyunVideoCutMode cutMode;
/* 视频质量 */
@property (nonatomic, assign) AliyunVideoQuality videoQuality;
/* 编码方式 */
@property (nonatomic, assign) AliyunVideoEncodeMode encodeMode;
/* 帧率 */
@property (nonatomic, assign) int fps;
/* 关键帧间隔 */
@property (nonatomic, assign) int gop;
/* 视频asset */
@property (nonatomic, strong) AVAsset *avAsset;
/* 是否仅仅展示视频 */
@property (nonatomic, assign) BOOL videoOnly;
/* 填充的背景颜色 */
@property (nonatomic, strong) UIColor *fillBackgroundColor;
/* 是否使用gpu裁剪 */
@property (nonatomic, assign) BOOL gpuCrop;
/* 裁剪码率 */
@property (nonatomic, assign) int bitrate;
```

对应表

属性	类型	释义
outputPath	NSString	输出路径
size	AliyunVideoSize	视频分辨率
ratio	AliyunVideoRatio	视频比例

minDuration	CGFloat	过滤相册视频最小时长
maxDuration	CGFloat	过滤相册最大时长
cutMode	AliyunVideoCutMode	裁剪模式
videoQuality	AliyunVideoQuality	视频质量
encodeMode	AliyunVideoEncodeMode	编码方式
fps	int	帧率
gop	int	关键帧间隔
bitrate	int	码率

可以通过调用初始化方法进行初始化

```
/**
创建AliyunVideoCropParam
*/
+ (instancetype)recordConfigWithVideoRatio:(AliyunVideoRatio)ratio
videoSize:(AliyunVideoSize)size;


/**
创建AliyunVideoCropParam
*/
+ (instancetype)recordConfigWithVideoRatio:(AliyunVideoRatio)ratio
videoSize:(AliyunVideoSize)size
outputPath:(NSString *)outputPath
minDuration:(CGFloat)minDuration
maxDuration:(CGFloat)maxDuration
cutMode:(AliyunVideoCutMode)cutMode
videoQuality:(AliyunVideoQuality)videoQuality
encodeMode:(AliyunVideoEncodeMode)encodeMode
fps:(int)fps
gop:(int)gop;
```

- **导入裁剪界面创建**

完成第一步设置好AliyunVideoCropParam后需要调用AliyunVideoBase下述接口创建导入裁剪Controller

```
 /**
创建一个相册导入界面

@param cropParam 裁剪视频参数
@return 相册导入界面
*/
- (UIViewController *)createPhotoViewControllerCropParam:(AliyunVideoCropParam *)cropParam;
```

示例：

```
 UIViewController *photoViewController = [[AliyunVideoBase shared]
createPhotoViewControllerCropParam:self.mediaInfo];
[AliyunVideoBase shared].delegate = self;
[self.navigationController pushViewController:photoViewController animated:YES];
```

- **录制界面回调**

设置AliyunVideoBase的delegate即可接收 裁剪完成 退出相册页 相册页面跳转录制页 等回调

```
 /**
裁剪完成回调

@param base AliyunVideoBase
@param cropVC 裁剪页VC
@param videoPath 视频保存路径
*/
- (void)videoBase:(AliyunVideoBase *)base cutCompeleteWithCropViewController:(UIViewController *)cropVC
videoPath:(NSString *)videoPath;

/**
退出相册页
*/
- (void)videoBasePhotoExit;

/**
相册页面跳转录制页

@param photoVC 当前相册VC
@return 跳转视频页的视频配置，若返回空则沿用相册裁剪页的视频配置
*/
- (AliyunVideoRecordParam *)videoBasePhotoViewShowRecord:(UIViewController *)photoVC;
```

四、上传

SDK生成的mp4文件，可接入阿里云上传SDK上传文件，可跳转至阿里云上传SDK下载。点击iOS上传SDK获取使用文档。

一、版本要求

iOS支持iOS8.0以上

二、开发环境配置

SDK开发环境为macOS Sierra 10.12.2 XCode8.0及以上

三、使用说明

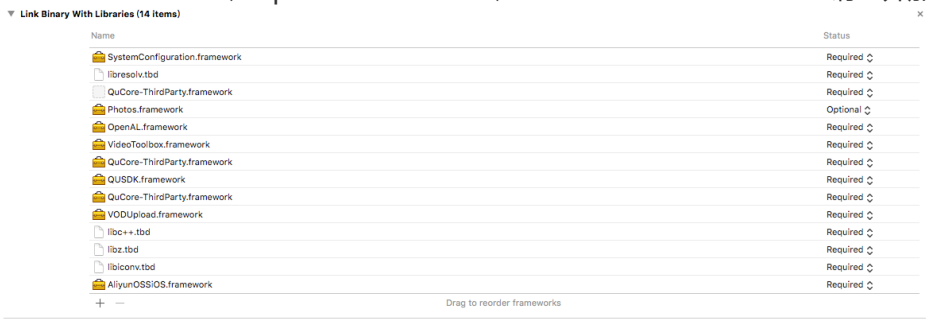
导入SDK

先下载SDK，详见SDK下载页面。
直接引入SDK的framework，名称为 AliyunVideoSDKPro

1. 开发者打开工程，选中目标target,依次选择”Build Phases”->”Link Binary With Libraries”,点击”+”号，点击“Add Other...”,导入AliyunOSSiOS.framework和VODUpload.framework。

   开发者打开工程，选中目标target,依次选择”General”->”Embededed Binaries”,点击”+”号，点击“Add Other...”,导入QUCore-ThirdParty.framework和AliyunVideoSDKPro.framework。

   同时，还需依赖libz.tbd、libc++.tbd、libiconv.tbd、libresolv.tbd、ImageIO.framework、CoreMedia.framework、CoreVideo.framework、SystemConfiguration.framework、Photos.framework、OpenAL.framework、VideoToolbox.framework请一并加上。截图如下：



   配置 Build Setting -- Build Options -- Enable Bitcode 选项为NO

5. 配置 Build Setting -- Linking -- Other Linker Flags 添加 -ObjC选项

添加用户权限

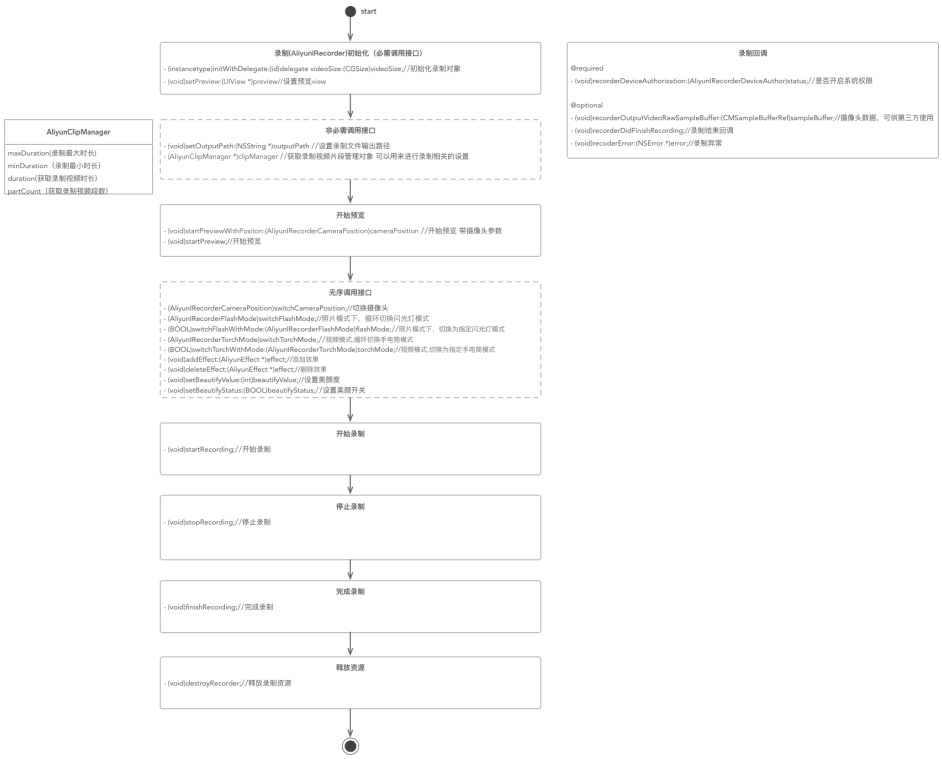
打开工程的 info.Plist 文件,添加如下字段：

```
`Privacy - Camera Usage Description`、
`Privacy - Microphone Usage Description`、
`Privacy - Photo Library Usage Description`。
```

按需添加字段描述。

录制

录制模块功能接口调用顺序图：



初始化参数

使用录制功能，需要初始化AliyunIRecorder对象，初始化方法如下：

```
- (instancetype)initWithDelegate:(id)delegate videoSize:(CGSize)videoSize;
```

参数说明，delegate传入代理对象，用于录制的回调信息，videoSize为设置的视频分辨率。

AliyunIRecorder对象提供了preview属性，为用于播放视频的预览view，必需进行设置，即

```
_recorder.preview = yourView;
```

录制视频管理

我们提供了视频录制片段的管理类—AliyunClipManager，该类可以对拍摄的视频进行设置，为AliyunIRecorder的属性，通过：

```
_recorder.clipManager;
```

获取。

该类提供如下方法：

```
    @property (nonatomic, assign) CGFloat maxDuration; //视频最大时长 默认8
@property (nonatomic, assign) CGFloat minDuration; //视频最小时长 默认0.5
@property (nonatomic, assign, readonly) CGFloat duration;//视频总时长
@property (nonatomic, assign, readonly) NSInteger partCount;//视频段数
- (void)deleteAllPart;//删除所有视频片段
- (void)deletePart;//删除最后一个视频片段
- (void)deletePart:(NSInteger)index; //删除某一个视频段
```

开始预览&结束预览

开始预览如下方法：

```
 - (void)startPreviewWithPositon:(AliyunIRecorderCameraPosition)cameraPosition;
- (void)startPreview;
```

建议在ViewController的- (void)viewWillAppear:(BOOL)animated时调用。

结束预览如下方法：

```
 - (void)stopPreview;
```

建议在ViewController的- (void)viewDidDisappear:(BOOL)animated时调用。

开始录制&结束录制&完成录制

在录制之前，确保已设置录制文件的输出路径，通过AliyunIRecorder的outputPath属性进行设置。

开始录制接口：

```
- (void)startRecording;
```

结束录制接口：

```
- (void)stopRecording;
```

完成录制接口：

```
- (void)finishRecording;
```

注:结束录制和完成录制的区别在于完成录制相当于录制流程的结束。这种场景可以出现在多段视频录制场景中，例如用户按下录制按钮，则开始录制，松开按钮，则结束录制，若不想再继续录制而想进入下一页，则调用完成录制。

释放资源

在ViewController销毁时，我们也需要释放录制资源，在dealloc中调用：

```
- (void)destroyRecorder;
```

回调接口

录制功能提供了一些回调函数：

```
@required
- (void)recorderDeviceAuthorization:(AliyunIRecorderDeviceAuthor)status;
```

该接口开发必须调用，用来回调系统级别的权限，检测是否具有拍摄权限等。

以下为非必需调用接口：

```
- (void)recorderOutputVideoRawSampleBuffer:(CMSampleBufferRef)sampleBuffer;
```

该接口回调视频的原始数据，开放这个接口的目的是让开发者可以使用摄像头数据来完成开发者自己的业务，如人脸识别。

```
- (void)recorderVideoDuration:(CGFloat)duration;
```

录制实时时长

```
- (NSInteger)recorderOutputVideoTextureName:(NSInteger)textureName textureSize:(CGSize)textureSie;
```

需要对视频纹理做渲染等二次加工时使用该回调。摄像头返回的原始视频纹理，摄像头数据格式为BGRA、YUV时都需实现

```
- (NSInteger)recorderOutputVideoUVTextureName:(NSInteger)textureName;
```

需要对视频纹理做渲染等二次加工时使用该回调。摄像头返回的原始视频纹理，摄像头数据格式仅为YUV时须实现，反之不实现

```
- (void)recorderDidStopWithMaxDuration;
```

录制到最大时长时的回调

```
- (void)recorderDidStopRecording;
```

停止录制回调

```
- (void)recorderDidFinishRecording;
```

录制结束的回调。

```
- (void)recoderError:(NSError *)error;
```

录制异常。

其他

设置效果

录制过程中可以添加效果,这些效果目前包含滤镜，水印，人脸动图三种类型。效果的基类是AliyunEffect类。三种类型具体对应如下类：

```
  `AliyunEffectFilter`--滤镜
`AliyunEffectPaster`--贴图
`AliyunEffectImage` --静态图片（水印）
```

添加效果的接口：

```
- (void)addEffect:(AliyunEffect *)effect;
```

移除效果的接口：

```
- (void)deleteEffect:(AliyunEffect *)effect;
```

录制配乐与调速接口：

```
AliyunEffectMusic *effectMusic = [[AliyunEffectMusic alloc] initWithFile:`设置音乐路径`];
effectMusic.startTime = 5; // 配乐开始时间点，从5秒开始
effectMusic.duration = 15; // 配乐持续时长，持续15秒
[_recorder applyMusic:effectMusic];
[_recorder setRate:2]; // 两倍速
```

配乐建议使用AAC编码格式。如果是MP3格式的音乐，会增加处理时间

音乐设置需要在录制前，开始录制后禁止调用配乐接口

录制速率大小在0.5~2之间，超过此区间无效

其他接口

切换摄像头

-(AliyunIRecorderCameraPosition)switchCameraPosition;

照片模式下，循环切换闪光灯

(AliyunIRecorderFlashMode)switchFlashMode;

照片模式下，切换为指定闪光灯模式

-(BOOL)switchFlashWithMode:(AliyunIRecorderFlashMode)flashMode;

视频模式下，循环切换手电筒模式

(AliyunIRecorderTorchMode)switchTorchMode;

视频模式下，切换为指定手电筒模式

-(BOOL)switchTorchWithMode:(AliyunIRecorderTorchMode)torchMode;

设置美颜开关

-(void)setBeautifyStatus:(BOOL)beautifyStatus;

设置美颜度

-(void)setBeautifyValue:(int)beautifyValue;

追踪识别的人脸点

-(void)faceTrack:(NSArray<AliyunFacePoint *> *)facePoints;

设置摄像头采集数据格式

```
  /**
提供三种格式：
kCVPixelFormatType_420YpCbCr8BiPlanarFullRange，
kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange，
kCVPixelFormatType_32BGRA，

默认kCVPixelFormatType_420YpCbCr8BiPlanarFullRange格式
*/
@property (nonatomic, assign) AliyunIRecorderVideoOutputPixelFormatType
```

```
outputType;
```

获取framework版本号

```
+ (NSString *)version;
```

导入裁剪

裁剪模块功能接口调用顺序图：

初始化参数

使用裁剪功能，需要初始化AliyunCrop对象，初始化方法如下：

```
- (instancetype)initWithDelegate:(id<AliyunCropDelegate>)delegate;
```

各实例变量说明：

```
  float startTime             截取时间起点         (单位：秒)
float endTime 截取时间终点 (单位：秒)
float fadeDuration 视频片段过渡动画时间 (单位：秒)
int cropMode 0 填充黑边 1 裁剪画面
NSString *videoPath 视频资源路径
NSString *outputPath 裁剪完成后的文件存放路径
CGSize outputSize 裁剪后的视频尺寸
int fps 帧率
int gop 关键帧间隔
AliyunVideoQuality videoQuality 视频质量
CGRect rect 保留的视频尺寸 (cropMode 为0时 无效)
```

开始裁剪

```
 - (int)startCrop;

返回值为0时表示配置正确
```

取消裁剪

```
 - (void)cancelCutVideo;
```

版本号

```
  + (NSString *)version;
```

- 其他接口

```
/**
删除文件
*/
- (BOOL)deleteFile:(int) index;

/**
清除上传列表
*/
- (BOOL)clearFiles;


/**
获取上传文件列表
*/
- (NSMutableArray<UploadFileInfo *> *)listFiles;
```

```
/**
取消单个文件上传，文件保留在上传列表中
*/
- (BOOL)cancelFile:(int)index;

/**
恢复已取消的上传文件
*/
- (BOOL)resumeFile:(int)index;

/**
停止上传
*/
- (BOOL)stop;

/**
暂停上传
*/
- (BOOL)pause;

/**
恢复上传
*/
- (BOOL)resume;

/**
使用Token恢复上传
*/
- (BOOL)resumeWithAuth:(NSString *)uploadAuth;

/**
使用Token恢复上传
*/
- (BOOL)resumeWithToken:(NSString *)accessKeyId
accessKeySecret:(NSString *)accessKeySecret
secretToken:(NSString *)secretToken
expireTime:(NSString *)expireTime;

/**
设置上传凭证
*/
- (BOOL)setUploadAuthAndAddress:(UploadFileInfo *)uploadFileInfo
uploadAuth:(NSString *)uploadAuth
uploadAddress:(NSString *)uploadAddress;
```

四、上传

SDK生成的mp4文件，可接入阿里云上传SDK上传文件，可跳转至阿里云上传SDK下载。点击iOS上传SDK获取使用文档。

一、快速开始

1.1 系统版本

iOS支持iOS8.0及以上

1.2 开发环境配置

SDK开发环境建议macOS Sierra 10.12.2 XCode9.0及以上

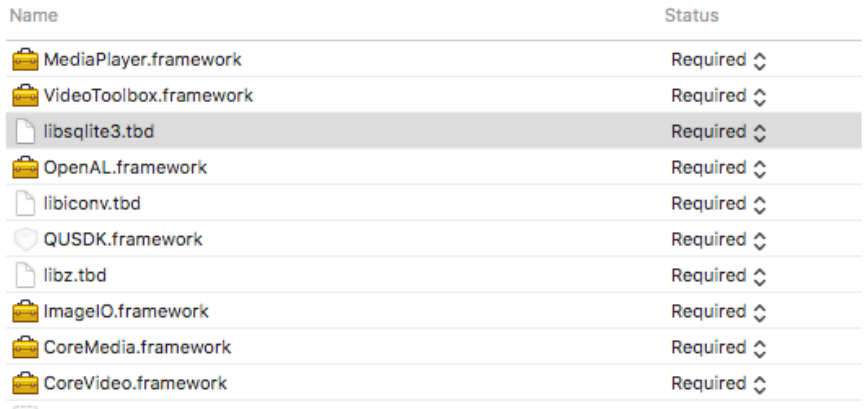
1.3 导入SDK

直接引入SDK的framework，名称为 AliyunVideoSDKPro

1. 开发者打开工程，选中目标target,依次选择”Build Phases”->”Link Binary With Libraries”,点击”+”号，点击“Add Other...”,导入AliyunVideoSDKPro.framework和QuCore-ThirdParty.framework。

   开发者打开工程，选中目标target,依次选择”General”->”Embededed Binaries”,点击”+”号，点击“Add Other...”,导入AliyunVideoSDKPro.framework和QuCore-ThirdParty.framework。

   同时，还需依赖libz.tbd、ImageIO.framework、CoreMedia.framework、CoreVideo.framework，VideoToolBox.framework、MediaPlayer.framework、OpenAL.framework，libc++.tbd,libsqlit3.tbd,libiconv.tbd请一并加上。截图如下：



1. 配置 Build Setting — Linking — Other Linker Flags 添加 -ObjC选项

1.4 添加用户权限

打开工程的 info.Plist 文件,添加如下字段

```
`Privacy - Camera Usage Description`、
`Privacy - Microphone Usage Description`、
`Privacy - Photo Library Usage Description`。
```

按需添加字段描述。

二、功能使用

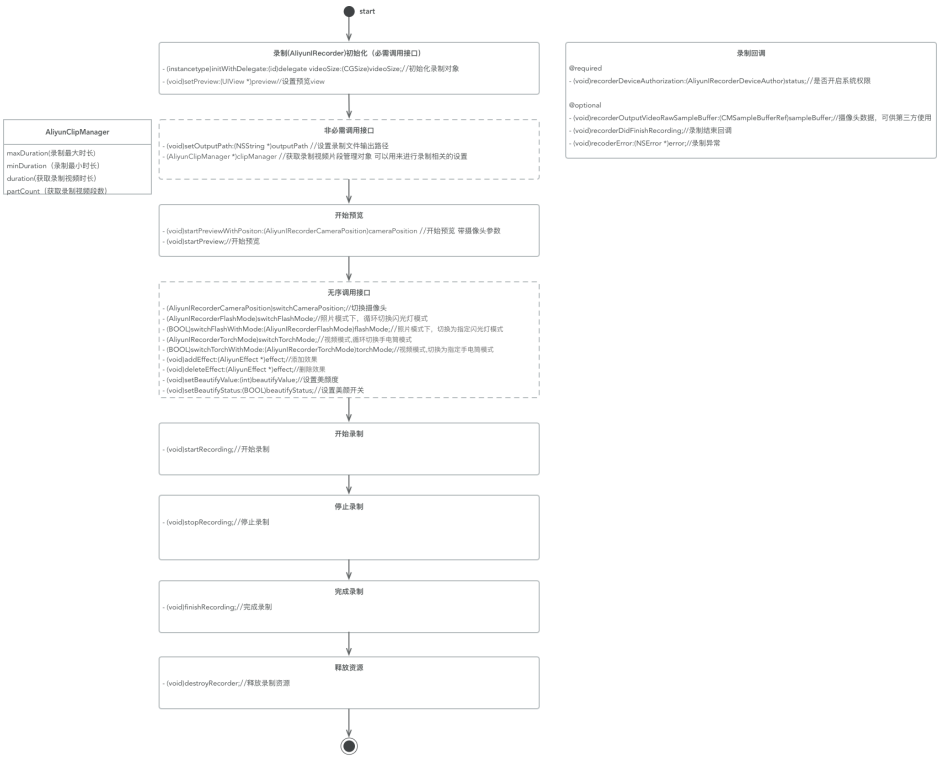
2.1 录制模块

2.1.1 接口说明

阿里视频云短视频SDK提供视频录制层核心接口，详细定义如下：

头文件	功能说明
AliyunIRecorder.h	视频录制核心类，通过这个类可以管理摄像头画面裁剪、管理录制流程、录制界面的特效管理、录制流程中的相关回调以及对接三方人脸库进行人脸贴图的接口。
AliyunClipManager.h	视频录制片段管理类，实现视频录制时长管理、片段数量管理、片段文件管理等功能
AliyunFacePoint.h	人脸识别特征点管理类，通过人脸识别SDK识别出特征点后，此类提供相关的接口传入特征点，SDK内部进行人脸贴图渲染
AliyunEffectFilter.h	添加滤镜的类，通过加载滤镜文件即可实现实时滤镜
AliyunEffectImage.h	添加水印的类，通过加载水印文件即可实现实时水印，支持水印位置设置
AliyunEffectPaster.h	实现贴图的类，此类和编辑模块的贴图类是公用的，当前在录制时这个类仅用于人脸贴图

2.1.2 接口调用顺序



初始化参数

引入<AliyunVideoSDKPro/AliyunIRecorder.h>头文件。

使用录制功能，需要初始化AliyunIRecorder对象，初始化方法如下：

```
- (instancetype)initWithDelegate:(id)delegate videoSize:(CGSize)videoSize;
```

参数说明，delegate传入代理对象，用于录制的回调信息，videoSize为设置的视频分辨率。

AliyunIRecorder对象提供了preview属性，为用于播放视频的预览view，必需进行设置并确保preview的比例和videoSize的比例一致，即

```
_recorder.preview = yourView;
```

taskPath属性，**必须设置，否则会导致无法输出文件**，即

```
_recorder.taskPath = yourTaskPath
```

开始预览&结束预览

开始预览如下方法：

```
- (void)startPreviewWithPositon:(AliyunIRecorderCameraPosition)cameraPosition;
- (void)startPreview;
```

建议在ViewController的- (void)viewWillAppear:(BOOL)animated时调用。

结束预览如下方法：

```
- (void)stopPreview;
```

建议在ViewController的- (void)viewDidDisappear:(BOOL)animated时调用。并且注意在进入其他界面时确保preview已经被stop掉，避免GL环境冲突。 启动拍摄拍摄界面，代码示例：

```
//在录制界面的viewControllerd的viewDidLoad里面add一个拍摄预览视图
UIView * preview = [[UIView alloc] init];
//设置preview的大小，需要和viewSize保持一致比例，比如是9:16全屏拍摄可设置为
preview.frame = CGRectMake(0,0,UIScreen.mainScreen.bounds.width,UIScreen.mainScreen.bounds.height);
[self.view.addSubview preview];
//创建一个AliyunIrecorder对象
AliyunIrecorder * recorder = [[AliyunIrecorder alloc] initWithDelegate:self videoSize:CGSizeMake(540,960)];
//设置recorder的preview
recorder.preview = preview

//在viewWillAppear里面调用开始预览的接口,需要指定摄像头请调用startPreviewWithPositon
[record startPreview];
```

开始录制&结束录制&完成录制

在录制之前，确保已设置录制文件的输出路径，通过AliyunIRecorder的outputPath属性进行设置，**注意该属性必须设置，否则会引起崩溃。**

开始录制接口：

```
- (void)startRecording;
```

结束录制接口：

```
- (void)stopRecording;
```

完成录制接口：

```
- (void)finishRecording;
```

注:结束录制和完成录制的区别在于完成录制相当于录制流程的结束。这种场景可以出现在多段视频录制场景中，例如用户按下录制按钮，则开始录制，松开按钮，则结束录制，若不想再继续录制而想进

入下一页，则调用完成录制。调用录制完成会把已经录制的多个片段合成为一个视频，如果用户用户使用我们的编辑模块，可以不用调用录制完成接口，编辑界面可以接受端端视频文件进行渲染，方便后期对多段视频再加工。 录制的代码示例：

```
 //设置输出路基
recorder.outputPath =  '自己设置录制完成后输出mp4的文件地址'
recorder.taskPath =  '自己设置存放中间文件、视频片段、配置文件的目录地址'

//设置最小和最大录制时长,默认值为0.5和8秒
recorder.clipManager.minDuration = 2;
recorder.clipManager.maxDuration = 30;

//在点击录制按钮里面实现开始录制
[recorder startRecording];

//在停止录制按钮里面实现停止录制
[recorder stopRecording];

//在录制的视频不需要经过编辑界面合成时，录制完成后调用finishRecording，这个接口会把录制的视频片段合成一个mp4输出到你设置的输出目录文件下面。如果需要对视频进行编辑的，建议不用掉此方法直接把当前的taskPath传给编辑界面。
[recorder finishRecording];

//在界面跳转前建议停止预览
[recorder stopPreview];
```

录制视频管理

我们提供了视频录制片段的管理类—AliyunClipManager，该类可以对拍摄的视频进行设置，为AliyunIRecorder的属性，通过：

```
 _recorder.clipManager;
```

获取。

该类提供如下方法：

```
@property (nonatomic, assign) CGFloat maxDuration; //视频最大时长 默认8
@property (nonatomic, assign) CGFloat minDuration; //视频最小时长 默认0.5
@property (nonatomic, assign, readonly) CGFloat duration;//视频总时长
@property (nonatomic, assign, readonly) NSInteger partCount;//视频段数
- (void)deleteAllPart;//删除所有视频片段
- (void)deletePart;//删除最后一个视频片段
- (void)deletePart:(NSInteger)index; //删除某一个视频段
```

释放资源

在ViewController销毁时，我们也需要释放录制资源，在dealloc中调用：

- (void)destroyRecorder;

回调接口

录制功能提供了一些回调函数：

```
@required
- (void)recorderDeviceAuthorization:(AliyunIRecorderDeviceAuthor)status;
```

该接口开发必须调用，用来回调系统级别的权限，检测是否具有拍摄权限等。

以下为非必需调用接口：

```
- (void)recorderOutputVideoRawSampleBuffer:(CMSampleBufferRef)sampleBuffer;
```

该接口回调视频的原始数据，开放这个接口的目的是让开发者可以使用摄像头数据来完成开发者自己的业务，主要场景是将数据传给第三方人脸识别SDK获取人脸识别后的特征点。如果第三方人脸识别SDK提供了人脸动图渲染能力，可以使用-(NSInteger)recorderOutputVideoPixelBuffer:(CVPixelBufferRef)pixelBuffer textureName:(NSInteger)textureName;接收渲染后的纹理。此接口的业务场景为：使用第三方人脸识别SDK人脸贴图渲染并使用阿里云视频SDK提供的录制能力。**注：如果使用这个接口就不需要使用我们SDK提供的人脸贴图相关的接口了**

```
- (void)recorderVideoDuration:(CGFloat)duration;
```

录制实时时长

```
- (void)recorderDidStopWithMaxDuration;
```

录制到最大时长时的回调，可应用于录制到最大时长时直接跳转界面

```
- (void)recorderDidStopRecording;
```

停止录制回调

```
- (void)recorderDidFinishRecording;
```

录制结束的回调。

```
- (void)recoderError:(NSError *)error;
```

录制异常。

实现人脸贴图的两种方式 实现人脸贴图功能都需要额外采购第三方人脸识别SDK。如果第三方人脸识别SDK仅提供特征点请使用方式一。如果第三方人脸SDK仅提供了动图渲染能力请使用方式二。如果第三方人脸SDK提供了特征点和渲染能力建议使用方式一，方式一的效率和性能会比较高。

方式一：使用第三方人脸识别提供的关键点，通过-(void)faceTrack:(NSArray<AliyunFacePoint *>*)facePoints;这个接口传入人脸SDK识别的特征点，目前支持传入左眼中心点、右眼中心的和嘴巴中心点。然后再通过SDK提供的前置贴图方法AliyunEffectPaster设置人脸贴图。

方式二：使用第三方人脸识别SDK提供的人脸贴图渲染能力，摄像头采集的数据通过-(void)recorderOutputVideoRawSampleBuffer:(CMSampleBufferRef)sampleBuffer这个代理给到人脸识别SDK，再通过人脸识别SDK贴完图将纹理通过-(NSInteger)recorderOutputVideoTextureName:(NSInteger)textureName textureSize:(CGSize)textureSie这个代理return给我们进行录制。这时不需要使用AliyunEffectPaster相关的接口，贴图素材也直接使用第三方提供的。

其他接口

滤镜、人脸贴图、水印

录制过程中可以添加效果,这些效果目前包含滤镜，水印，人脸动图三种类型。效果的基类是AliyunEffect类。三种类型具体对应如下类：

```
  `AliyunEffectFilter`--滤镜
`AliyunEffectPaster`--贴图
`AliyunEffectImage` --静态图片（水印）
```

添加效果的接口：

```
  - (void)addEffect:(AliyunEffect *)effect;
```

移除效果的接口：

```
  - (void)deleteEffect:(AliyunEffect *)effect;
```

添加水印代码示例：

```
//创建水印对象
AliyunEffectImage *waterMark = [[AliyunEffectImage alloc] initWithFile:‘设置水印图片文件路径’];
//设置水印大小
wartermark.frame = CGRectMake(10, 200, 50, 50)
//应用水印效果，可在录制时实时添加水印，不需要实时水印的如有编辑功能也可以在编辑界面添加水
```

```
印，两者取其一应用即可。
[recorder applyImage:wartermark];
```

添加滤镜代码示例：

```
//创建滤镜对象
AliyunEffectFilter *filter = [[AliyunEffectFilter alloc] initWithFile:‘设置滤镜目录地址’];
//应用滤镜效果，可实时作用于录制画面。
[recorder applyFilter:filter];
```

录制配乐与调速代码示例：

```
AliyunEffectMusic *effectMusic = [[AliyunEffectMusic alloc] initWithFile:`设置音乐路径`];
effectMusic.startTime = 5; // 配乐开始时间点，从5秒开始
effectMusic.duration = 15; // 配乐持续时长，持续15秒
[_recorder applyMusic:effectMusic];
[_recorder setRate:2]; // 两倍速
```

配乐建议使用AAC编码格式。如果是MP3格式的音乐，会增加处理时间

音乐设置需要在录制前，开始录制后禁止调用配乐接口

录制速率大小在0.5~2之间，超过此区间无效

可选接口

切换摄像头

-(AliyunIRecorderCameraPosition)switchCameraPosition;

改变视频分辨率，改变视频分辨率的同时需改变preview的大小，需保持一致的比例。

- (void)reStartPreviewWithVideoSize:(CGSize)videoSize;

照片模式下，循环切换闪光灯

-(AliyunIRecorderFlashMode)switchFlashMode;

照片模式下，切换为指定闪光灯模式

-(BOOL)switchFlashWithMode:(AliyunIRecorderFlashMode)flashMode;

视频模式下，循环切换手电筒模式

(AliyunIRecorderTorchMode)switchTorchMode;

视频模式下，切换为指定手电筒模式

-(BOOL)switchTorchWithMode:(AliyunIRecorderTorchMode)torchMode;

设置美颜开关

-(void)setBeautifyStatus:(BOOL)beautifyStatus;

设置美颜度

-(void)setBeautifyValue:(int)beautifyValue;

追踪识别的人脸点，这个接口应用于使用第三方人脸识别SDK提前特征点，然后用我们SDK提供的动图渲染能力进行渲染

-(void)faceTrack:(NSArray<AliyunFacePoint *> *)facePoints;

设置摄像头采集数据格式

```
 /**
提供三种格式：
kCVPixelFormatType_420YpCbCr8BiPlanarFullRange，
kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange，
kCVPixelFormatType_32BGRA，

默认kCVPixelFormatType_420YpCbCr8BiPlanarFullRange格式
*/
@property (nonatomic, assign) AliyunIRecorderVideoOutputPixelFormatType
outputType;
```

获取framework版本号

+ (NSString *)version;

12.设置录制速率

- (void)setRate:(CGFloat)rate;

13.静音

@property (nonatomic, assign) BOOL mute;

14.内置人脸识别

```
/**
使用自带人脸识别，开启该功能，系统会在检测到有人脸动图加入时自动进行追踪显示
*/
@property (nonatomic, assign) BOOL useFaceDetect;
```

15.设置输出视频码率

```
/** 码率 */
@property (nonatomic, assign) int bitrate;
/**
```

16.人脸数量回调

```
人脸数量的回调，在useFaceDetect开启的状态下生效
*/
@property (nonatomic, copy) void (^faceNumbersCallback)(int num);
```

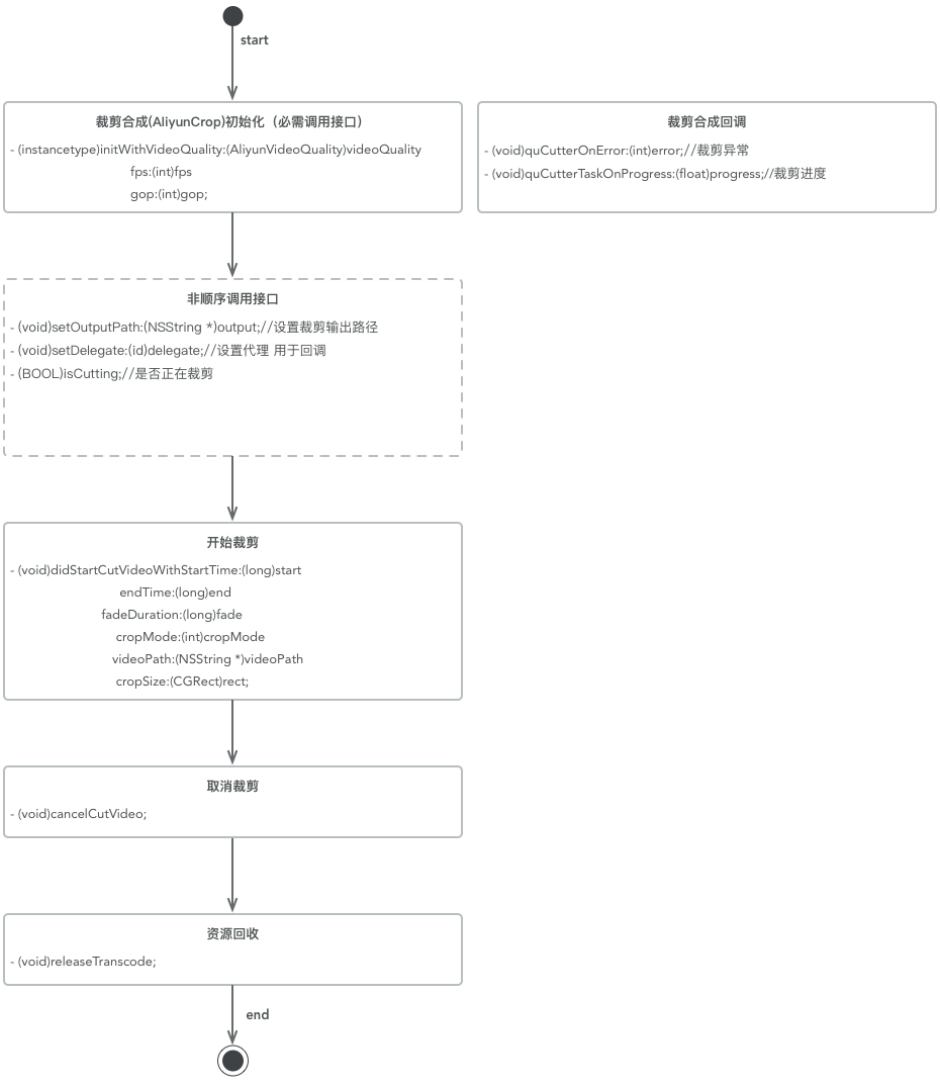
2.2 裁剪模块

2.2.1 接口说明

阿里视频云短视频SDK提供视频裁剪核心接口，详细定义如下：

头文件	功能说明
AliyunCrop.h	视频和音乐裁剪核心类，通过这个类可以实现视频按时长裁剪、按画面裁剪，裁剪模式可选画面填充和裁切，并支持视频输出参数设置。

2.2.2 裁剪模块功能接口调用顺序图：



初始化参数

使用裁剪功能，需要初始化AliyunCrop对象，初始化方法如下：

```
- (instancetype)initWithDelegate:(id<AliyunCropDelegate>)delegate;
```

各实例变量说明：

```
float startTime 截取时间起点 (单位：秒)
float endTime 截取时间终点 (单位：秒)
float fadeDuration 视频片段过渡动画时间 (单位：秒)
AliyunCropCutMode cropMode 0 填充黑边 1 裁剪画面
NSString *inputPath 视频或音乐资源路径
NSString *outputPath 裁剪完成后的文件存放路径
CGSize outputSize 裁剪后的视频尺寸
int fps 帧率
int gop 关键帧间隔
AliyunVideoQuality videoQuality 视频质量
```

```
CGRect rect 保留的视频尺寸 (cropMode 为0时 无效)
UIColor *fillBackgroundColor 视频填充模式下，填充的背景颜色
int bitrate 码率
BOOL useHW 是否启用gpu裁剪
```

关于裁剪范围rect中x,y,width,height值的计算：设定视频左上角为原点O(0,0),横向为x轴，纵向为y轴,假设视频裁剪区域为Z。rect中x值为Z区域的左上角顶点相对于原点O的x轴像素值，y值为Z区域的左上角顶点相对于原点O的y轴像素值。w为裁剪区域Z的宽度像素值，h为裁剪区域Z的高度像素值。

关于裁剪音乐，无需设置rect，outputSize，gop等视频相关参数。

裁剪视频代码示例：

```
 AliyunCrop * crop = [[AliyunCrop alloc] initWithDelegate:self];
crop.inputPath = '需要裁剪的视频文件地址';
crop.outputPath = '裁剪后输出的视频文件地址'
crop.cropMode = AliyunCropModeScaleAspectCut;
crop.outputSize = CGSizeMake(540, 540);//如输出的视频为540*540
crop.rect = CGRectMake(0, 0, 320, 320);//这里可以认为你在当前视频上画了一个裁剪框（要保持视频画面不被裁剪需要设置宽为输入视频宽，高为输输入视频的高），裁剪框大小需和输出视频的分辨率的大小保持固定的比例关系。否定可能变形。
crop.startTime = 0.0;
crop.endTime = 5.0;
crop.fillBackgroundColor = [UIColor greenColor];//填充模式下填充的背景颜色
//开始裁剪
[crop startCrop];
```

裁剪音乐代码示例：

```
 liyunCrop * crop = [[AliyunCrop alloc] initWithDelegate:self];
crop.inputPath = '需要裁剪的音乐文件地址';
crop.outputPath = '裁剪后输出的音乐文件地址'
crop.startTime = 0.0;
crop.endTime = 5.0;
//开始裁剪
[crop startCrop];
```

通过AliyunCropDelegate来监听裁剪完成、裁剪过程和裁剪错误。

```
    /**
裁剪失败时错误回调

@param error 错误码
*/
- (void)cropOnError:(int)error;

/**
裁剪进度回调

@param progress 当前进度
```

```
*/
- (void)cropTaskOnProgress:(float)progress;

/**
退出回调
主动取消或裁剪完成时回调
*/
- (void)cropTaskOnComplete;
```

开始裁剪

```
- (int)startCrop;

返回值为0时表示配置正确
```

取消裁剪

```
- (void)cancel;
```

版本号

```
+ (NSString *)version;
```

2.2.3 图片裁剪模块功能接口

- **初始化参数**使用裁剪功能，需要初始化AliyunImageCrop对象，调用alloc进行初始化。

各实例变量说明：

```
/**
需要被裁剪的图片
*/
@property (nonatomic, strong) UIImage *originImage;

/**
裁剪后的图片大小
*/
@property (nonatomic, assign) CGSize outputSize;

/**
照片裁剪模式， 0：填充模式 1:裁剪模式
*/
@property (nonatomic, assign) AliyunImageCropMode cropMode;

/**
裁剪区域，仅适用于裁剪模式，例如一张图片（200x200像素 ，cropRect可设置为 (0,0,100,100)（单位为像素））
*/
```

```
@property (nonatomic, assign) CGRect cropRect;

/**
照片填充模式下，填充的背景颜色
*/
@property (nonatomic, strong) UIColor *fillBackgroundColor;
```

裁剪出来的图片大小以像素为单位进行计算。

关于裁剪范围rect中x,y,width,height值的计算：设定视频左上角为原点O(0,0),横向为x轴，纵向为y轴,假设图片裁剪区域为Z。rect中x值为Z区域的左上角顶点相对于原点O的x轴像素值，y值为Z区域的左上角顶点相对于原点O的y轴像素值。w为裁剪区域Z的宽度像素值，h为裁剪区域Z的高度像素值。

开始裁剪

- (UIImage *)generateImage;

2.3 编辑模块

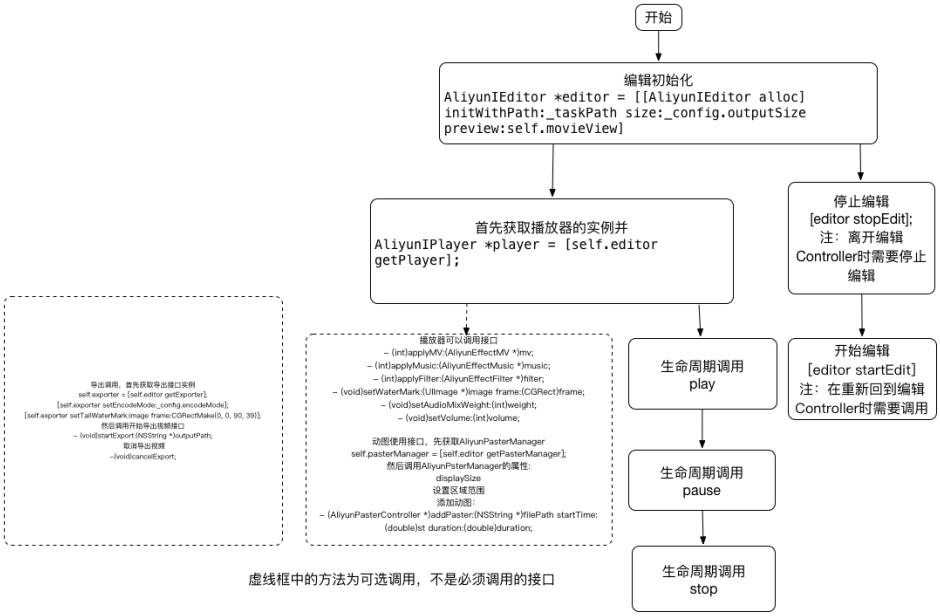
2.3.1 接口说明

阿里视频云短视频SDK提供视频编辑核心接口，详细定义如下：

头文件	功能说明
AliyunEditor.h	视频编辑核心类，编辑界面的播放器、各类特效（动图、MV、滤镜、音乐、水印）、视频的导出都通过此类来管理。
AliyunIPlayer.h	播放器控制类，实现播放控制相关的功能。
AliyunIPlayerCallback.h	播放器相关触发的回调类，包含开始、结束、播放进度、seek和异常状态的回调。
AliyunImporter.h	视频导入类，用来构建编辑界面需要json文件，目前是进入编辑界面前的必要条件。
AliyunEffect.h	各类特效（动图、MV、滤镜、音乐、水印）的基类、各类特效的初始化都是通过资源文件地址来创建的。
AliyunEffectFilter.h	添加滤镜的类，通过加载滤镜文件即可实现实时滤镜
AliyunEffectImage.h	添加水印的类，通过加载水印文件即可实现实时水印，支持水印位置设置
AliyunEffectPaster.h	实现贴图的类，此类和编辑模块的贴图类是公用的，当前在录制时这个类仅用于人脸贴图
AliyunEffectMusic.h	实现音乐效果类，通过此类添加音乐，静音、音量调节等功能通过编辑的edit来控制。
AliyunEffectPasterBase.h	动图和字幕的基类，主要用于实现动图的位置、大小、持续时间等方法。
AliyunEffectSubtitle.h	纯文字的类，这里面主要是配合AliyunPasterController来实现，如果直接文字渲

	染也可以将文字转换为图片，这是里面的属性就只需要使用到基类里面的位置和持续时间的属性了。
AliyunIExporter.h	视频合成导出的类，主要用于编辑界面视频导出，片尾水印也通过此类提供的方法来合成。
AliyunIExporterCallback.h	视频合成导出的回调了，用来监听导出视频时的各种状态。
AliyunIPasterRender.h	动图和字幕的直接渲染类，可以不使用AliyunPasterController来直接将动图、字幕合成到视频里面，主要用于UI和交互完全自定义的情况下实现动图和字幕功能。
AliyunPasterBaseView.h	动图的视图基类，里面提供的唯一方法是将视图转化为图片，主要用于直接渲染动图、字幕、涂鸦等功能。
AliyunPasterController.h	动图控制器，主要提供对动图的操控方法，demo中提供的动图移动、旋转、镜像、编辑等功能都是通过此类来实现的，主要用于在动图、字幕交互方面和我们demo一直的情况下使用。
AliyunPasterManager.h	动图管理类，主要提供对动图、字幕的添加、删除等方法，主要用于在动图、字幕交互方面和我们demo一直的情况下使用。
AliyunPasterUIEventProtocol.h	动图、字幕操作的回调方法，动图经过编辑后可以通过此类提供的方法监听，主要用于在动图、字幕交互方面和我们demo一直的情况下使用。
AliyunVideoParam.h	视频参数类，主要用户视频录制、导入、导出对视频相关参数的控制。

2.3.2编辑模块功能接口调用顺序图:



- **生成配置文件**编辑模块支持对一段或多段视频的编辑与导出。首先，需要生成包括视频片段路径和输出分辨率等信息的配置文件。这个工作由AliyunImporter提供。

初始化方法如下：

```
- (instancetype)initWithPath:(NSString *)taskPath outputSize:(CGSize)outputSize;
```

注：taskPath是文件夹路径，由调用者提供给SDK使用。SDK将配置文件和视频片段等存放在taskPath文件夹中，SDK模块间通过taskPath传递视频相关信息。

我们可以通过如下方法添加一段或多段视频片段或者图片片段：

```
/**
添加视频路径

@param videoPath 视频路径
@param animDuration 转场动画时长,单位秒

duartion为当前视频片段和上段视频片段间的转场时长，不能小于视频本身时长
添加的第一段视频没有转场，duartion需要设为0
*/
- (void)addVideoWithPath:(NSString *)videoPath animDuration:(CGFloat)animDuration;



/**
添加视频路径并指定编辑视频开始点和结束点

@param videoPath 视频路径
@parma startTime 视频开始时间
@parma duration 视频持续时间
@param animDuration 转场动画时长,单位秒

duartion为当前视频片段和上段视频片段间的转场时长，不能小于视频本身时长
添加的第一段视频没有转场，duartion需要设为0
*/
- (void)addVideoWithPath:(NSString *)videoPath startTime:(CGFloat)startTime duration:(CGFloat)duration
animDuration:(CGFloat)animDuration;



/**
增加图片
@param image 照片
@param duration 播放时间
@param animDuration 动画过渡时间
@return 图片路径
*/
- (NSString *)addImage:(UIImage *)image duration:(CGFloat)duration animDuration:(CGFloat)animDuration;
```

最后，调用如下方法生成配置文件：

```
- (void)generateProjectConfigure;
```

这样，SDK会在taskPath文件夹中生成一个配置文件。编辑模块初始化时只需要传递taskPath路径，SDK内部

会自动读取配置文件内容。导入代码示例：

```
//创建导入对象
AliyunImporter *importer = [[AliyunImporter alloc] initWithPath:‘用户设置的存放配置文件和视频片段的目录文件夹地址’ outputSize:‘用户设置的输出视频的分辨率’];
//添加需要导入的视频，可以导入多个文件。转场效果仅支持淡入淡出效果，设置为0表示没有转场效果，每一个视频直接都可以设置转场时间。设置了转场效果后两个视频会叠加，叠加后视频总时长会减少。
[importer addVideoWithPath:‘需要导入的视频文件地址’ animDuration：0];
//创建并设置视频参数，主要设置视频以裁剪模式还是填充模式进入编辑界面
AliyunVideoParam * videoParam = [[AliyunVideoParam alloc] init];
videoParam.scaleMode = AliyunCropCutModeScaleAspectFill;
[importer setVideoParam:videoParam];
//生成配置文件，生成配置文件后在edit的初始化使用
[importer generateProjectConfigure];
//导入后自己完成界面跳转
```

初始化参数

使用编辑功能，需要初始化AliyunIEditor对象，初始化方法如下:

```
- (instancetype)initWithPath:(NSString *)taskPath size:(CGSize)outputSize preview:(UIView *)preview;
```

参数说明： taskPath 文件夹路径 preview 编辑预览视图

视频预览

视频预览需要获取播放器，调用AliyunIEditor的如下接口获取一个播放器实例:

```
/**
获取播放器接口实例

@return AliyunIPlayer
*/
- (id<AliyunIPlayer>)getPlayer;
```

获取到一个AliyunIPlayer类的实例后，可以做如下操作

开始播放

```
- (void)play;
```

seek接口

```
-(void)seek:(float)time;
```

暂停播放

```
- (void)pause;
```

继续播放

```
- (void)resume;
```

获取是否正在播放

```
- (BOOL)isPlaying;
```

重新开始播放

```
- (void)replay;
```

停止播放

```
- (void)stop;
```

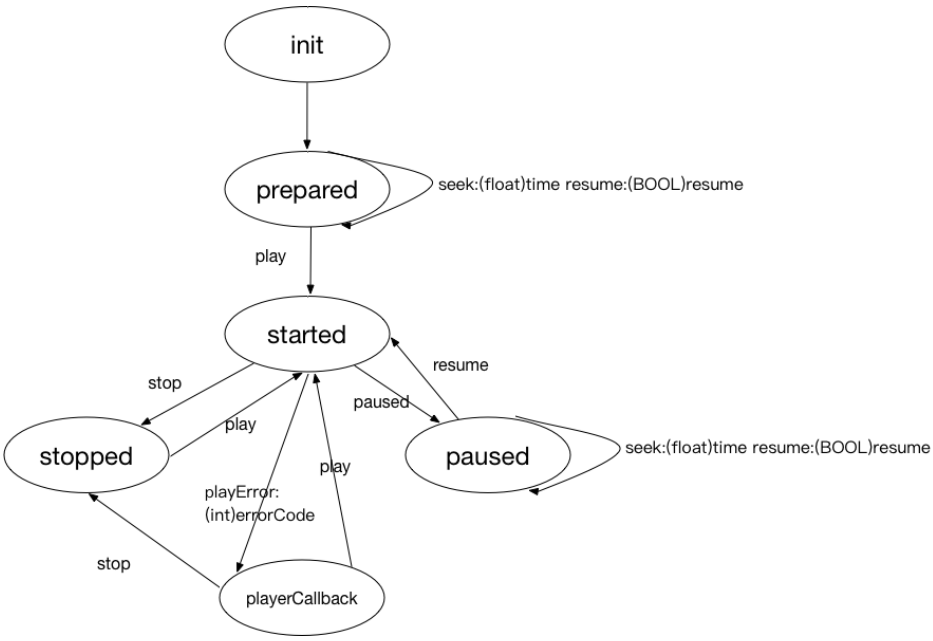
获取总时长

```
- (double)getDuration;
```

获取当前播放时间

```
- (double)getCurrentTime;
```

播放器状态及生命周期图：



播放器的使用示例：

```
   //创建播放器预览视图
UIView * preview = [[UIView alloc] init];
//设置preview的大小，需要和viewSize保持一致比例，比如是9:16全屏播放可设置为
preview.frame = CGRectMake(0,0,UIScreen.mainScreen.bounds.width,UIScreen.mainScreen.bounds.height);
[self.view.addSubview preview];
//创建编辑器,viewWillAppear里面实现
AliyunIEditor *edit = [[AliyunIEditor alloc] initWithPath:'录制/导入时设置的taskPath' size:  '输出视频的分辨率'
preview:preview];
//设置编辑的代理,然后实现AliyunIPlayerCallback, AliyunIExporterCallback代理方法
edit.delegate = self
//获取播放器
AliyunIPlayer *player = [edit getPlayer];
//接下来根据自己的需求调用播放器的控制方法
```

特效使用

使用特效之前确保播放器已经就绪,处于onPrepared的状态。可以使用的特效主要有滤镜,贴纸,字幕,音乐,MV,水印。其中贴纸,字幕 有交互需求,sdk提供了两种使用方式。第一种方式是AliyunPasterManager来管理各类交互动作，在交互方式和我们demo类似的情况下使用。第二种方式是使用AliyunPasterRender来直接渲染字幕和贴纸，主要在UI和交互完全自定义的情况下使用。另外，除贴纸,字幕外的其他特效均使用AliyunIEditor接口添加。 滤镜,音乐,MV三种特效均使用特效类AliyunEffect,注：该类为Base基类，具体特效有对应得子类

滤镜,音乐,MV特效使用:

```
/**
使用mv
@param mv mv配置文件路径
@return 错误码
*/
```

```
- (int)applyMV:(AliyunEffectMV *)mv;
/**
使用音乐
@param music music配置文件路径
@return 错误码
*/
- (int)applyMusic:(AliyunEffectMusic *)music;
/**
使用滤镜
@param filter filter配置文件路径
@return 错误码
*/
- (int)applyFilter:(AliyunEffectFilter *)filter;
```

设置音量、混音权重：

```
/**
设置是否静音

@param mute 静音
*/
- (void)setMute:(BOOL)mute;

/**
设置音量

@param volume 音量：0-200
*/
- (void)setVolume:(int)volume;

/**
设置混音权重

@param weight 混音权重0-100
*/
- (void)setAudioMixWeight:(int)weight;
```

注：一般在使用了音乐或MV后调用混音权重接口,如果没有音乐资源则调用该接口无效。

水印特效

```
/**
添加视频水印

@param imagePath 视频水印路径
@param frame 水印frame
*/
- (void)setWaterMark:(NSString *)imagePath frame:(CGRect)frame;
```

注：水印frame的宽高比需要与水印图片的真实宽高比相同。例如：一张100*200的图片在设置frame的时候需要width/height = 0.5；同时请用户确保imagePath真实存在水印图片示例:

```
[self.editor setWaterMark:[UIImage imageNamed:@"watermark"] frame:CGRectMake(20, 20, 40, 40)];
```

渲染背景颜色

```
/**
视频渲染最底层背景颜色 在填充模式下具有效果

@param color 颜色
*/
- (void)setRenderBackgroundColor:(UIColor *)color;
```

贴纸、字幕

贴纸和字幕拥有编辑可选项,如果添加贴纸后需要用户对贴纸进行大小,位置和显示时间的编辑,则可以使用AliyunPasterManager来添加贴纸特效,字幕亦然。

- 调用AliyunIEditor的如下接口，获取AliyunPasterManager实例

```
- (AliyunPasterManager *)getPasterManager;
```

AliyunPasterManager主要接口：

- 设置编辑区域

@property (nonatomic, assign) CGSize displaySize; 注:编辑区域即视频播放区域大小，可设置为播放器视图大小

- 添加动图

```
/**
添加动图

@param filePath 动图资源路径
@param st 动图开始时间
@param duration 动图持续时间
@return 返回动图控制器
*/
- (AliyunPasterController *)addPaster:(NSString *)filePath startTime:(double)st duration:(double)duration;
```

- 添加字幕

```
 /**
@param text 文字 如果为空 则会第一次会进入编辑 如果不为空 则直接显示 不会进入编辑状态
@param bounds 大小
@param st 字幕开始时间
@param duration 字幕持续时间
@return 返回动图控制器
*/
- (AliyunPasterController *)addSubtitle:(NSString *)text bounds:(CGRect)bounds startTime:(CGFloat)st
```

```
duration:(CGFloat)duration;
```

- 删除动图控制器，在删除动图时调用

```
/**
删除动图控制器，在删除动图时调用
@param pasterController 需要删除的动图控制器
@return YES：删除成功 NO:删除失败
*/
- (BOOL)deletePasterController:(AliyunPasterController *)pasterController;
```

- 获取所有的动图控制器

```
/**
获取所有的动图控制器

@return 动图控制器数组
*/
- (NSArray *)getAllPasterControllers;
```

- 通过id获取pasterController

```
/**
通过id获取pasterController

@param obj id
@return pasterController
*/
- (AliyunPasterController *)getPasterControllerByObj:(id)obj;
```

- 动图上某个位置是否存在动图

```
/**
动图上某个位置是否存在动图

@param point 点击的位置
@param time 当前视频播放的当前时间
@return 若当前时刻该位置有动图，则返回动图控制器，否则，返回nil
*/
- (AliyunPasterController *)touchPoint:(CGPoint)point atTime:(double)time;
```

- 删除所有动图控制器，即删除所有动图

```
- (void)removeAllPasterControllers;
```

- 删除所有的普通动图

```
- (void)removeAllNormalPasterControllers;
```

- 删除所有的字幕动图

```
- (void)removeAllCaptionPasterControllers;
```

- 删除所有的纯文字动图

```
- (void)removeAllSubtitlePasterControllers;
```

- 获取当前正在编辑的动图控制器

```
- (AliyunPasterController *)getCurrentEditPasterController;
```

获取AliyunPasterManager对象后,需要设置动图展示区域的大小,如果不设置,默认为以屏幕的宽作为边的正方形区域,设置后才能正确的计算出贴纸在展示区域内正确的大小和位置。使用贴纸后会返回AliyunPasterController对象,AliyunPasterController定义了贴纸的位置,大小等相关信息,可以通过get方法获取,它也提供了贴纸编辑的一些接口,比如移除贴纸,设置贴纸时间,通知贴纸开始编辑及结束编辑。

AliyunPasterController主要接口如下:

设置动图View，**必须要设置**

```
@property (nonatomic, strong) UIView *pasterView;
```

动图类型

```
@property (nonatomic, assign, readonly) AliyunPasterEffectType pasterType;
```

动图的旋转角度 单位：弧度

```
@property (nonatomic, assign) CGFloat pasterRotate;
```

动图的位置(x,y)

```
@property (nonatomic, assign) CGPoint pasterPosition;
```

动图的宽高

```
@property (nonatomic, assign) CGSize pasterSize;
```

动图的位置大小

```
@property (nonatomic, assign) CGRect pasterFrame;
```

动图镜像

```
@property (nonatomic, assign) BOOL mirror;
```

文字内容

```
@property (nonatomic, copy) NSString *subtitle;
```

文字位置 相对于动图本身的位置大小

```
@property (nonatomic, assign, readonly) CGRect subtitleFrame;
```

文字的后台配置字体

```
@property (nonatomic, copy, readonly) NSString *subtitleConfigFontName;
```

文字的后台配置字体的id

```
@property (nonatomic, assign, readonly) NSInteger subtitleConfigFontId;
```

文字是否描边

```
@property (nonatomic, assign, readonly) BOOL subtitleStroke;
```

文字颜色

```
@property (nonatomic, strong) UIColor *subtitleColor;
```

文字描边颜色

```
```

```
@property (nonatomic, strong) UIColor *subtitleStrokeColor;
```

文字字体

```
@property (nonatomic, copy) NSString *subtitleFontName;
```

关键帧图片

```
@property (nonatomic, strong, readonly) UIImage *kernelImage;
```

动图开始时间 单位：s

```
@property (nonatomic, assign) CGFloat pasterStartTime;
```

动图结束时间 单位：s

```
@property (nonatomic, assign) CGFloat pasterEndTime;
```

动图持续时间 单位：s

```
@property (nonatomic, assign) CGFloat pasterDuration;
```

编辑区域

```
@property (nonatomic, assign) CGSize displaySize;
```

- 关键帧图片

```
- (NSString *)getKernelImagePath;
```

- 唯一id

```
- (int)eid;
```

- 开始编辑 将要进入编辑状态

```
- (void)editWillStart;
```

- 开始编辑 进入编辑状态

```
- (void)editDidStart;
```

- 开始编辑 进入编辑

```
- (void)editProcess;
```

- 完成编辑

```
- (void)editCompleted;
```

- 字幕编辑完成

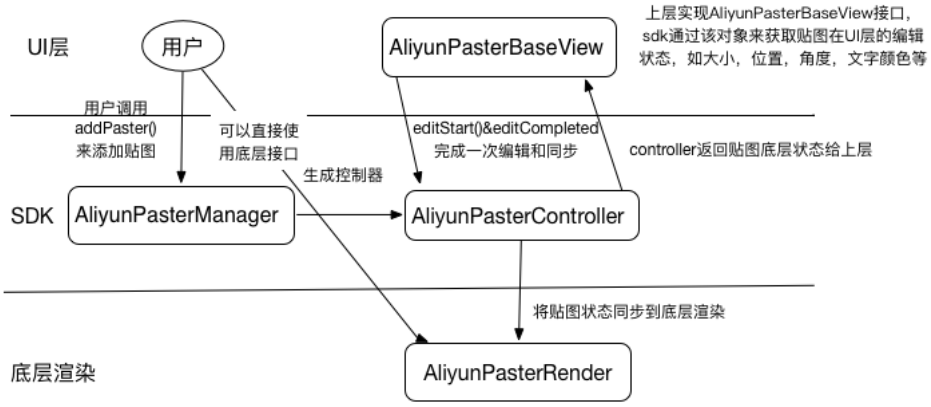
```
- (void)editCompletedWithImage:(UIImage *)image;
```

通过AliyunPasterControllerDelegate可以监听动图状态如下：

```
 /**
移除动图控制器

@param obj 动图控制器
*/
- (void)onRemove:(id)obj;

/**
动图控制器将要进入编辑

@param obj 动图控制器
*/
- (void)onEditWillBegin:(id)obj;

/**
动图控制器进入编辑

@param obj 动图控制器
*/
- (void)onEditDidBegin:(id)obj;

/**
动图控制器完成编辑

@param obj 动图控制器
*/
- (void)onEditEnd:(id)obj;

/**
动图控制器完成编辑
```

```
@param obj 动图控制器
@param image 需要渲染的图片
*/
- (void)onEditEnd:(id)obj image:(UIImage *)image;
```

AliyunPasterManager贴纸编辑添加MVC设计示意图:



从上面的示意图中我们看到AliyunPasterManager负责创建AliyunPasterController对象,控制器中定义了AliyunPasterBaseView接口来获取上层UI的状态,用户交互对UI的修改会通过AliyunPasterBaseView反馈到控制层,控制层再同步UI状态到渲染层,从而完成了贴纸的编辑渲染,这样UI层可以自由定制,只要实现AliyunPasterViewUIEventProtocol即可。

```
- (void)eventBoundsDidChanged:(CGRect)aBounds;
- (void)eventCenterDidChanged:(CGPoint)aCenter;
- (void)eventRotateDidChanged:(CGFloat)aRotate;
- (void)eventTextBoundsDidChanged:(CGRect)aBounds;
- (void)eventTextCenterDidChanged:(CGPoint)aCenter;
- (void)eventMirrorChanged:(BOOL)isMirror;
- (void)eventPasterViewClosed:(UIView *)pasterView;
- (void)eventEditDidEnd;
- (void)eventEditWillBegin;
```

用户直接调用底层AliyunPasterRender接口

用处除了可以使用前面的方式加贴图，前面的方式包含了我们一套默认UI，如果用户想完全采用自定义UI，也可以直接调用底层接口来实现。

调用AliyunIEditor的如下接口，获取AliyunPasterRender实例

```
- (AliyunPasterRender *)getPasterRender;
```

提供接口主要如下：

```
/**
底层接口，添加动图渲染

@param paster 动图对象
@return 是否成功
*/
- (BOOL)addGifPaster:(AliyunEffectPaster *)paster;
```

```
/**
底层接口，添加纯文字

@param subtitle 纯文字动图对象
@param textImage 文字截图
*/
- (void)addSubtitlePaster:(AliyunEffectSubtitle *)subtitle textImage:(UIImage *)textImage;

/**
底层接口，添加字幕动图

@param caption 字幕动图对象
@param textImage 文字截图
*/
- (void)addCaptionPaster:(AliyunEffectCaption *)caption textImage:(UIImage *)textImage;

/**
移除动图

@param basePaster 动图对象
*/
- (void)removePaster:(AliyunEffectPasterBase *)basePaster;

/**
隐藏动图

@param basePaster 动图对象
*/
- (void)hidePaster:(AliyunEffectPasterBase *)basePaster;


/**
显示普通动图

@param paster 动图对象
*/
- (void)showGifPaster:(AliyunEffectPaster *)paster;

/**
显示纯文字动图

@param subtitle 纯文字动图对象
*/
- (void)showSubtitlePaster:(AliyunEffectSubtitle *)subtitle;

/**
显示字幕动图

@param caption 字幕动图对象
*/
- (void)showCaptionPaster:(AliyunEffectCaption *)caption;
```

参数说明：AliyunEffectPaster、AliyunEffectCaption、AliyunEffectSubtitle均继承自

AliyunEffectPasterBase。AliyunEffectPasterBase的主要属性有：

```
    @property (nonatomic, assign) CGFloat startTime;//开始时间
@property (nonatomic, assign) CGFloat endTime;//结束时间
@property (nonatomic, assign) CGFloat minDuration;//最短时间
@property (nonatomic, assign) CGFloat rotate;//旋转角度
@property (nonatomic, assign) CGSize displaySize;//editzoneview
@property (nonatomic, assign) CGRect frame;//位置大小
@property (nonatomic, assign) CGPoint position;//位置
@property (nonatomic, assign) CGSize size;//大小
@property (nonatomic, assign) CGFloat width;
@property (nonatomic, assign) CGFloat height;
@property (nonatomic, assign) BOOL mirror;
```

说明：

startTime: 贴图开始的时间

endTime: 贴图结束的时间

minDuration: 贴图最小持续时间 一般从动图配置文件里面读取，如果设置的持续时间小于最小持续时间，会得不到想要的效果

rotate:动图的角度

displaySize:为编辑区域的大小，即播放视频的容器view的大小

frame:动图在编辑区域上的位置和大小

position: 动图在编辑区域上的中心点位置

size: 动图在编辑区域上的大小

width: 动图的宽

height: 动图的高

mirror: 是否镜像

用户在构建这个对象时，为了确定最后渲染时动图的位置大小，需要设置displaySize和动图自身的frame。

MV使用示例：

```
//根据MV所在的目录创建MV对象
AliyunEffectMV *mvEffect = [[AliyunEffectMV alloc] initWithFile:'MV文件目录地址'];
//应用MV效果
[edit applyMV:mvEffect];
//建议应用MV后从头开始播放，MV是不支持Seek的
[[edit getPlayer] replay];
```

音乐使用示例：

```
//根据MP3文件创建音乐效果对象
```

```
AliyunEffectMusic *music = [[AliyunEffectMusic alloc]initWithFile:'音乐文件的mp3地址'];
//应用音乐效果
[edit applyMusic:music];
//对音乐和原音比例调整
[edit setAudioMixWeight:50];
//设置音量
[edit setVolume:100];
```

滤镜使用示例：

```
//创建滤镜对象
AliyunEffectFilter *filter = [[AliyunEffectFilter alloc] initWithFile: ‘设置滤镜目录地址’ ];
//应用滤镜效果，可实时作用于录制画面。
[edit applyFilter:filter];
```

动图使用示例：

```
//使用动图资源文件目录创建动图对象
AliyunEffectPaster *motionSticker = [[AliyunEffectPaster alloc] initWithFile:'动图所在的文件目录地址'];
//设置动图的展示区域，展示区域设置为播放器的大小
motionSticker.displaySize = playerView.bounds.size;
//动图的位置，动图位置为动图中心点在播放器中的位置
motionSticker.position = CGPointMake(160,200);
//设置动图的大小，大小为在播放器中展示的大小，动图大小需要和动图提供的图片大小保存一致的比例，建议设置动图的默认大小为动图配置文件（json）中提供宽和高的一半
motionSticker.size = CGSizeMake(141,53);
//设置动图的开始时间，可以在视频的任意位置添加
motionSticker.startTime = 0.0;
//设置动图的结束时间，设置为全局动图时可将结束时间设置为视频时长
motionSticker.endTime = [[edit getPlayer] getDuration];
//添加动图，实时渲染到视频中去
[[edit getPasterRender] addGifPaster:motionSticker];
```

字幕（纯文字）使用示例

```
//纯文字对象可以不用任何参数直接创建
AliyunEffectSubtitle *subtitle = [[AliyunEffectSubtitle alloc] init];
//设置字幕的展示区域，展示区域设置为播放器的大小
subtitle.displaySize = playerView.bounds.size;
//字幕的位置，字幕位置为字幕中心点在播放器中的位置
subtitle.position = CGPointMake(160, 280);
//字幕的旋转角度，需要时使用，默认为水平（不设置时）
subtitle.rotate = M_PI_4;
//设置字幕的开始时间，可以在视频的任意位置添加，全局时设置为0
subtitle.startTime = 0.0;
//设置字幕的结束时间，设置为全局字幕时可将结束时间设置为视频时长
subtitle.endTime = [[edit getPlayer] getDuration];
//以下部分将文字转化为图片，然后将图片渲染到视频上
AliyunPasterBaseView *pastBaseView = [[AliyunPasterBaseView alloc] init];
pastBaseView.frame = CGRectMake(0, 0, UIScreen.mainScreen.bounds.width,50);//文字图片的大小
UILabel *textLable = [[UILabel alloc] initWithframe: pastBaseView.bounds];
textLable.text = "我是一个有趣的文字"
```

```
textLable.textColor = [UIColor redColor];
textLable.textAlignment = NSTextAlignmentCenter ;
textLable.backgroundColor = [UIColor blackColor];
textLable.font = [UIFont systemFontOfSize: 28];
[pastBaseView addSubview:textLable];//在动图基类上添加我需要的文字
UIImage *textImage = [pastBaseView captureImage];//使用动图基类生成一张图片
//设置字幕大小，字幕大小可设置为生成图片的大小
subtitle.size = pastBaseView.bounds.size;
//添加字幕，实时渲染到视频中去
[[edit getPasterRender] addSubtitlePaster:subtitle, textImage: textImage];
```

字幕动图（气泡）使用示例：

```
//使用字幕动图所在的文件夹目录创建字幕动图对象
AliyunEffectCaption *subtitleSticker = [[AliyunEffectCaption alloc] initWithFile: '动图字幕文件夹目录地址'];
//设置字幕动图的展示区域，展示区域设置为播放器的大小
subtitleSticker.displaySize = playerView.bounds.size;
//字幕动图的位置，字幕动图位置为字幕动图中心点在播放器中的位置
subtitleSticker.position = CGPointMake(160, 280);
//设置字幕动图的开始时间，可以在视频的任意位置添加，全局时设置为0
subtitle.startTime = 0.0;
//设置字幕动图的结束时间，设置为全局字幕动图时可将结束时间设置为视频时长
subtitle.endTime = [[edit getPlayer] getDuration];
//创建动图基类，主要用于将文字生成图片
AliyunPasterBaseView *pastBaseView = [[AliyunPasterBaseView alloc] init];
//设置生成文字图片的位置大小，文字图片在气泡里面展示，隐藏不能超出气泡的大小，可以将文字图片设置为和气泡一样大小，可以将json文件提供的气泡宽、高除2来设置，如下
pastBaseView.frame = CGRectMake(0, 0,375/2.0, 250/2.0);
UILabel *textLable = [[UILabel alloc] initWithframe: pastBaseView.bounds];
textLable.text = "我是一个有趣的文字"
textLable.textColor = [UIColor redColor];
textLable.textAlignment = NSTextAlignmentCenter;
textLable.font = [UIFont systemFontOfSize: 18];
[pastBaseView addSubview:textLable];//在动图基类上添加我需要的文字
UIImage *textImage = [pastBaseView captureImage];//使用动图基类生成一张图片
//设置字幕动图的大小，可以设置为json文件中宽高除以2作为字幕动图的大小
subtitleSticker.size = CGSizeMake(375/2.0, 250/2.0);
//设置字幕动图文字图片在气泡中的文字，根据自己的需求设置，但不要超过气泡的大小
subtitleSticker.textFrame = pastBaseView.frame;
//设置气泡里面的文字在气泡中的出现时间，比如先出现气泡后出现文字
subtitleSticker.textRelativeToBeginTime = 0.5;
//设置气泡里面的文字在气泡中结束时间，比如文字先结束，然后结束气泡。这里的气泡持续时间为2S（可以在字幕动图的json文件中获取），结束时间设置为1.5S
subtitleSticker.textRelativeToEndTime = 1.5;
//添加字幕动图，实时渲染到视频中去
[[edit getPasterRender] addCaptionPaster:subtitleSticker, textImage: textImage];
```

涂鸦

SDK提供了涂鸦功能，用户可以根据自身业务选择此功能。涂鸦主要涉及有两个类：AliyunIPaint和AliyunICanvasView。AliyunIPaint可以理解为画笔，AliyunICanvasView可以理解为画布。

画笔AliyunIPaint提供的方法有：

```
/**
线条颜色
*/
@property (nonatomic, strong) UIColor *lineColor;


/**
线条宽度
*/
@property (nonatomic, assign) CGFloat lineWidth;


/**
线条阴影颜色
*/
@property (nonatomic, strong) UIColor *shadowColor;


/**
线条阴影宽度
*/
@property (nonatomic, assign) CGFloat shadowWidth;


/**
init

@param lineWidth 线条宽度
@param lineColor 线条颜色
@return self
*/
- (instancetype)initWithLineWidth:(CGFloat)lineWidth
lineColor:(UIColor *)lineColor;
```

画笔目前支持线条颜色、宽度、线条阴影颜色和阴影宽度的设置。

画布AliyunICanvasView主要提供：

```
/**
回调
*/
@property (nonatomic, weak) id<AliyunICanvasViewDelegate> delegate;


/**
是否允许越界画图，默认不允许
*/
@property (nonatomic, assign) BOOL enableCrossBorder;



/**
画笔
*/
```

```
@property (nonatomic, strong) AliyunIPaint *paint;



/**
init

@param frame 画板frame
@param paint 画笔
@return self
*/
- (instancetype)initWithFrame:(CGRect)frame
paint:(AliyunIPaint *)paint;



/**
更改画笔配置

@param paint 画笔
*/
- (void)changePaint:(AliyunIPaint *)paint;



/**
清空所有线条(不可恢复)
*/
- (void)remove;


/**
撤销上一步
*/
- (void)undo;


/**
恢复上一步
*/
- (void)redo;



/**
完成

@return 涂鸦图片
*/
- (UIImage *)complete;
```

画布提供了基本的功能，如撤销、恢复、修改画笔配置等功能。

画笔和画布的使用示例代码如下：

```
    AliyunIPaint *paint = [[AliyunIPaint alloc] initWithLineWidth:SizeWidth(5.0) lineColor:[UIColor whiteColor]];
self.paintView = [[AliyunICanvasView alloc] initWithFrame:rect paint:paint];
self.paintView.delegate = self;
self.paintView.backgroundColor = [UIColor clearColor];
[self addSubview:self.paintView];
```

上面的示例代码中，paintView加入到你的目标view上，即可进行涂鸦。

涂鸦完成后，通过调用complete接口可以生成图片，生成的图片再通过AliyunEditor的applyPaint:接口即可渲染到视频上。

导出

通过AliyunIEditor的接口获取导出实例AliyunIExporter

```
- (id<AliyunIExporter>)getExporter;
```

AliyunIExporter主要接口如下:

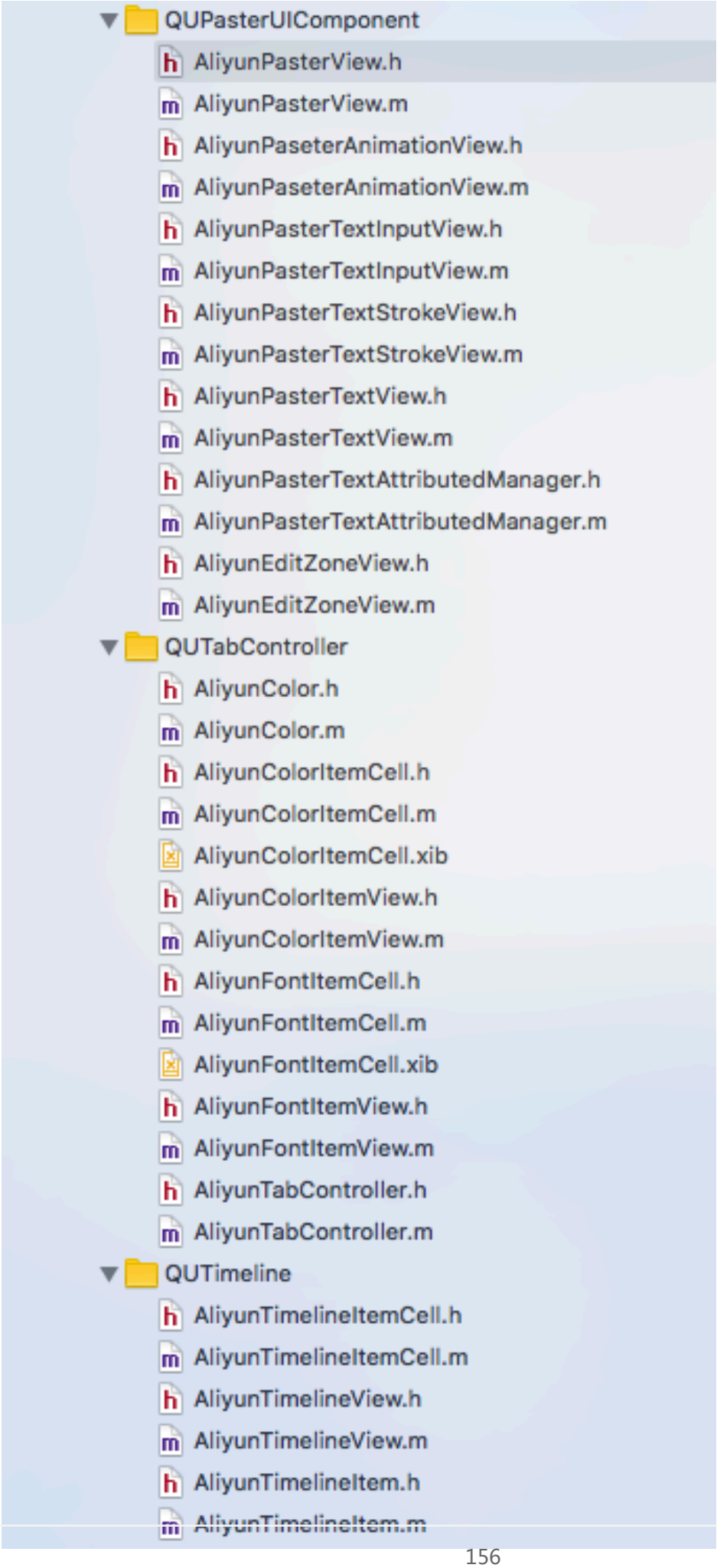
```
  /**
添加片尾水印

@param image 水印图片
*/
- (void)setTailWaterMark:(UIImage *)image frame:(CGRect)frame;

/**
设置编码模式

@param encodeMode 编码模式 0:软编 1:硬编
*/
- (void)setEncodeMode:(int)encodeMode;

/**
设置视频输出参数

@param videoParam 视频输出参数
*/
- (void)setVideoParam:(AliyunVideoParam *)videoParam;

/**
开始导出视频

@param outputPath 导出路径
*/
- (void)startExport:(NSString *)outputPath;

/**
取消导出视频
*/
-(void)cancelExport;
```

导出视频使用示例：

```
//获取导出对象
AliyunIExporter * exporter = [edit getExporter]；
//设置导出视频的参数，这里的视频参数可以和导入的视频参数一致，导入的视频参数已经设置，这里也可以不用设置
[exporter setVideoParam:videoParam];
//片尾效果可以根据自己的需求来选择是否调用此方法
[exporter setTailWaterMark:'水印图片UIImage类型',frame:CGRectMake(60, 150, 200,200)];
//开始导出，可以监听导出的各种状态：开始导出、结束导出、取消导出、导出结束、导出进度和导出异常
[exporter startExport:'输出后视频存放的文件地址'];
```

2.4 开源View

SDK所有UI部分全部开源，用户可以最大限度的进行自定义UI。


QUPasterUIComponent
AliyunPasterView.h
AliyunPasterView.m
AliyunPaseterAnimationView.h
AliyunPaseterAnimationView.m
AliyunPasterTextInputView.h
AliyunPasterTextInputView.m
AliyunPasterTextStrokeView.h
AliyunPasterTextStrokeView.m
AliyunPasterTextView.h
AliyunPasterTextView.m
AliyunPasterTextAttributedManager.h
AliyunPasterTextAttributedManager.m
AliyunEditZoneView.h
AliyunEditZoneView.m
QUTabController
AliyunColor.h
AliyunColor.m
AliyunColorItemCell.h
AliyunColorItemCell.m
AliyunColorItemCell.xib
AliyunColorItemView.h
AliyunColorItemView.m
AliyunFontItemCell.h
AliyunFontItemCell.m
AliyunFontItemCell.xib
AliyunFontItemView.h
AliyunFontItemView.m
AliyunTabController.h
AliyunTabController.m
QUTimeline
AliyunTimelineItemCell.h
AliyunTimelineItemCell.m
AliyunTimelineView.h
AliyunTimelineView.m
AliyunTimelineItem.h
AliyunTimelineItem.m

AliyunPasterView：动图展示UIAliyunPasterTextInputView：动图键盘文字输入UIAliyunTabController：键盘弹起组件AliyunTimeline：编辑页上方导航条UI

三、常见问题

iPhone4s开启人脸识别，进入录制页面会卡顿两秒，画面才动

答:4S性能差，人脸识别耗性能

iPhone4S开启人脸识别，录制动图视频，动图慢半拍

答:4S性能差，人脸识别耗性能

视频支持哪些格式？

答:目前只支持MP4、MOV。不支持3gp格式

taskPath是什么？

答:taskPath是文件夹路径，由调用者提供给SDK使用。SDK将配置文件和视频片段等存放在taskPath文件夹中，SDK模块间通过taskPath传递视频相关信息。

短视频常见问题