百度智能云

VideoCreatingSDK 文档

目录

功能发布记录

短视频SDK

发布时间	功能概述
2022-09	1.SDK支持arm64位系统 2.优化SDK架构
2020-12	1.增加接入文档预览和版本 2.支持1:1及4:3比例拍摄 3.支持预置相机拍摄水印
2020-09	1.美妆贴合精准度优化,人像分割/头发分割效果优化 2.新增人脸单帧图片处理（贴纸、美妆、滤镜） 3.引擎升级动力学效果，包括重力感和碰撞检测，如软绳、金属链
2020-08	1.支持自定义本地贴纸 2.支持双语字幕 3.新增贴纸的互动提示
2020-06	1.新增多种转场特效 2.新增多种滤镜特效 3.新增时间特效：慢速、快速，按倍速设置，并可针对局部设置变速
2020-05	1.音频编辑支持变声特效 2.图片合成视频支持设置图片显示时长

直播SDK

发布时间	功能概述
2021-2	支持录屏直播，包括： 1.隐私模式 2.叠加摄像头预览（安卓） 3.录屏横屏推流
2020-12	支持实时背景音乐，包括： 1.背景音乐循环播放 2.背景和人声音量调节 3.背景音乐混音和替换
2020-11	支持低延时直播
2020-09	1.支持连麦本地混流 2.优化用户端连麦效果
2020-07	1.增加拍摄倒计时 2.demo增加AR贴纸互动提示
2020-06	1.资源动态下载 2.包体优化
2020-04	1.支持动态码率 2.推流支持添加水印

短视频SDK

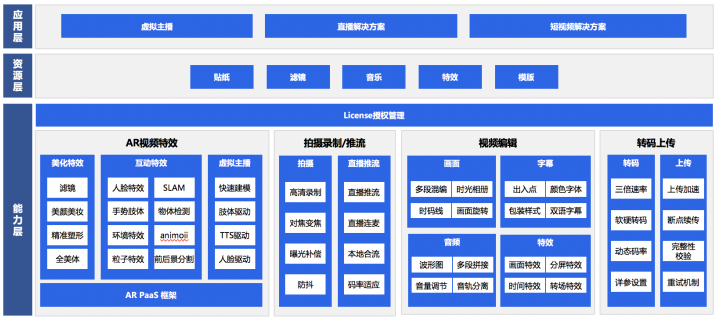
产品描述

产品简介

短视频SDK是智能视频SDK的短视频场景化产品，作为一站式音视频产品，提供集成了拍摄、AR特效、剪辑、播放、音乐、双语字幕等功能的客户端SDK，帮助用户聚焦短视频场景，快速轻松实现基于移动端的短视频应用。

作为短视频生产利器，支持滤镜/美颜/美妆/五官级塑型/人脸/手势/肢体/环境等AR互动特效等，可用于短视频的拍摄环节，也可以用于视频图片处理的后编辑环节。

产品功能全景图：



名词解释

- 流媒体上下文(Streaming Context)

包含时间线、预览窗口、采集、资源包管理等相关信息集合的对象，Context被销毁之后，SDk视频制作框架也随之不复存在。

- 采集(Capture)

捕获摄像头设备画面。

- 录制(Record)

输出采集画面到指定格式文件中。

- 动画贴纸(Animated Sticker)

带有动画效果的贴纸，叠加在视频上产生一些特殊效果。

- 滤镜（Filter）

带有不同颜色的滤镜，叠加在视频上改变画面颜色。

- AR特效（AR Effect）

根据人脸、手势和肢体关键点定位，产生各种互动效果的贴纸。

- 预览窗口(Live Window)

实时显示时间线或者采集图像的窗口。

- 视频解析度(Video Resolution)

视频的基本信息，包括图像宽高和像素横纵比等。

- 音频解析度(Audio Resolution)

音频的基本信息，包括采样率和声道数等。

功能说明

视频拍摄/编辑

模块	功能点	说明
界面	UI界面自定义	不限制界面布局，UI界面自由定制
资源	贴纸资源	提供丰富的贴纸资源，并免费提供工具可自定义贴纸
	音乐资源	提供版权音乐库，支持音乐库对接
基础拍摄	拍摄控制	支持拍摄时的前后摄像头切换、闪光灯的关闭
	焦距调节	支持手势双指调节焦距（放大或缩小）
	对焦模式	支持手动对焦和自动对焦。
	防抖	IOS支持拍摄时防抖
	清晰度	支持标清、高清、超清拍摄，支持自定义码率、帧率、gop
	时长设置	自定义拍摄的最短和最长时长
	分段录制	拍摄过程中可以暂停分段并且可以回删
	延时拍摄	设置/开启延时拍摄，倒计时结束后自动开始录制
	变速拍摄	支持拍摄变速，包括：极慢、慢、正常、快、极快
	背景音乐	拍摄前可以选择本地的 MP3 作为背景音
	照片/视频	支持拍照片及拍视频
拍摄美化特效	滤镜	调节画面：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等
	基础美颜	支持美白、磨皮、大眼、瘦脸
	美妆	支持腮红、高光、阴影、眉毛、睫毛、口红、眼线、眼影、美瞳
	五官精准塑形	支持眼睛、鼻子、下巴、脸型、额头等设置，详见“AR特效”sheet，描述
视频/图片编辑	多段视频合成	可将多个视频、图片串接合成为一个视频(数量不限)
	视频/图片混编	可以使用图片、视频混合进行制作,实现MV、相册等功能
	视频/图片素材添加	在多段制作中,插入一个新的视频或图片
	视频/图片素材删除	可以删除多段制作中的某一个视频或图片
	视频/图片素材排序	调整多段制作中的素材顺序
	视频&图片旋转	可以旋转视频、图片的方向
	图片时长设定	设定图片展现的时长
	图片运动效果	可以设定图片的运动效果（开始结束画面）
	图片显示方式	设定图片在画面中展现全部或者只显示区域内容
	视频入出点设置	支持设置视频的入点、出点，实现视频掐头去尾功能
	视频分割	可在暂停位置，将视频分割成两段
	视频抽帧	允许将视频中的任意一帧画面抽取为图片保存，可作为视频封面
	缩略图时码线	SDK提供编辑时的视频缩略图获取
	效果实时预览	拍摄、编辑中添加滤镜、贴纸、主题等各种特效，支持实时预览
音频编辑	背景音乐	支持添加一路音频
	音轨分离及音量调节	在视频中实现分离音轨，可独立设置原声、背景音的音量、静音
	音乐裁剪	可拖动设置音乐的入点
字幕编辑	多段字幕	一次制作可添加多个字幕，并支持多行文字
	字幕入出点设定	设定每一个字幕在视频上出现和消失时间
	字幕画面位置	设定字幕在画面上的显示位置
	字幕样式	可设置字幕样式，包括：字体、字号、描边（粗细、透明度、颜色）、阴影（距离、透明度、颜色）等
编辑特效	转场特效	实现片段间过渡，带有淡入、缩放、渐变等几种基础转场
	画面特效	黑白、放大、灵魂出窍等多种特效
	分屏特效	支持三屏、六屏、九屏特效
AR特效后编辑	滤镜	支持编辑环节设置画面滤镜
	基础美颜	支持编辑环节设置基础美颜，包括：美白、磨皮、红润、大眼、瘦脸
	五官塑形	支持编辑环节调整五官精准塑形，包括：眼睛、鼻子、下巴、脸型、额头等
	人脸贴纸	支持编辑环节设置人脸贴纸，包括：2D/3D/皮肤贴纸
	支持类型	以上特效在图片和视频的后编辑中均支持
主题模版	卡点视频	支持按卡点视频模版，选择图片文件，一键生成视频
	MV视频	支持按MV视频模版，选择图片和视频文件，一键生成视频
合成上传	视频合成	可设置打包压缩的的码率、分辨率等
	视频上传	提供上传能力至
拍摄互动特效	人脸/手势/肢体/环境互动特效	详见“AR特效”功能列表，全部可用于短视频拍摄场景

AR特效

功能项		功能说明
造型美	基础滤镜	支持内置和自定义色卡滤镜，如：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等
	贴纸	屏幕贴纸，支持图片、序列帧动画等随屏素材，支持27种混合模式（如相加、平均、差值、变暗、反色、底片、线性减淡、正片叠底、滤色、柔光等）
	屏幕特效	支持分屏（上下/左右分屏、3/4/6/9屏等）、镜像、闪屏、灵魂出窍、抖动、水波纹、眩晕、斜进出等屏幕特效，也可通过shader自定义屏幕特效。
轻美妆	美白	美颜能力,可调节美白强度,满足个性需求
	磨皮	美颜能力,可调节磨皮强度,满足个性需求
	红润	美颜能力,可调节红润强度,满足个性需求
微整形	瘦脸	美颜能力，实时调节瘦脸参数
	大眼	美颜能力，实时调节大眼参数
高级美妆塑形	美妆特效	全局支持口红，腮红，修容，眼影（含睫毛），眼线，眉毛；单项支持美瞳、祛法令纹、祛黑眼圈
	高级美颜	五官精准塑形（眼睛、鼻子、下巴、脸型、额头、颧骨、发际线等）
人脸特效	人脸贴纸	基于人脸190+特征点精准定位和跟踪，添加2D/3D动态面具贴纸
	皮肤贴纸	基于人脸识别，实现皮肤贴纸，如：脸谱、豹子脸
	人脸变形	基于人脸识别，实现面部变形功能，如：嘟嘟脸、方脸
	表情识别	表情识别，挑眉、眨眼、张嘴、点头、嘟嘴（5个）触发特效，如：喷火帽、巫师帽
	多人脸	通过AI算法，检测画面中多个人脸区域
	换脸	实现双人换脸特效
	人脸实时驱动	基于人脸识别，支持实时驱动3D面具表情的玩法，如：柴犬面具
手势特效	手势识别触发	手势识别触发特效，支持比V、点赞、OK、单手比心、食指比1、握拳、手掌的检测（7个手势）
	手部贴纸	基于手势识别，添加2D/3D贴纸道具
	指尖检测	指尖点的精准检测和跟踪
	手势实时驱动	基于手部21关节点检测，实现手部驱动手偶的玩法
肢体特效	瘦身/增高	基于全局实现瘦身/增高/哈哈镜效果
	动作识别	识别肢体动作，并触发特效，包含2D/3D特效
前后景分割	人像分割	通过AI算法，识别视频中的人物区域（上半身），实现替换背景，背景虚化等功能
	头部分割-大头特效	通过AI算法，可精准识别头部区域，可实现大头特效玩法
	天空分割	通过AI算法，识别视频中的天空区域，实现天空背景更换，替换及素材叠加。
	头发分割	通过AI算法，识别视频中的头发区域，可实现染发、换发型等效果
环境特效	SLAM放置	通过即时定位与跟踪算法，在实景中放置3D模型动画
	空间粒子特效	模型粒子特效，能模拟现实中的雨、雪、落叶、飞花、流星等自然景观，以及魔法球、光环、火焰、冬去春来、花瓣雨、控雨等特效
	触屏手势粒子	触屏绘制烟花或气泡状的图像、写字，出粒子效果
	天空顶	星空、粉色天空等穹顶效果，360空间氛围渲染
	分屏	设置分屏滤镜，实现画面分割
	物体检测	通过AI算法，检测画面中的物体（当前支持杯子、其他物体需定制）
	透明视频/绿幕视频	支持通过手机姿态定位，在空间中放置透明/绿幕扣背景视频素材

核心优势

- 创新引领：智能的AR特效视频拍摄工具，全面的视频生产玩法，引领业界风向标。支持人脸特效、美化及滤镜、交互&环境特效总计30+特效视频能力。
- 双语字幕：依托百度技术优势，双语语音识别，自动生成双语字幕。
- 简单快捷：标准易用的配置操作引导，拍摄器业务上线，快人一步。
- 端到端全链路解决方案：提供短视频的采、编、播、管、存、发，快速搭建智能小视频业务。
- 灵活架构：插拔式模块化设计，支持三方采集和特效处理。
- 丰富的资源生态：营造贴纸生产者和购买者联系的生态平台，提供丰富的生态资源。
- 全网最低价：性价比高，在体验最优质的技术服务的同时，享受超低价格。

购买指南

产品定价

短视频SDK套餐包内容如下表所示。套餐详细价格、贴纸资源、音乐资源请点击短视频SDK价格详情查看。

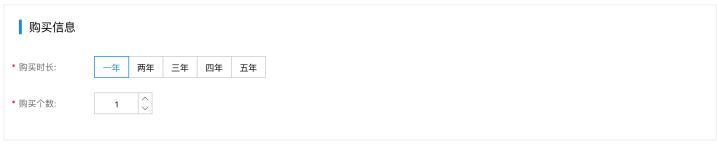
模块	功能	功能说明	精简版	基础版	标准版	专业版
界面	UI界面自定义	不限制界面布局，UI界面自由定制	✓	✓	✓	✓
资源	贴纸资源	可自行筛选贴纸库里的贴纸内容，并免费提供工具供设计师制作贴纸	×	×	20款	40款
基础拍摄	拍摄控制	支持拍摄时的前后摄像头切换、闪光灯的关闭	✓	✓	✓	✓
	焦距调节	支持手势双指调节焦距（放大或缩小）	✓	✓	✓	✓
	对焦模式	支持手动对焦和自动对焦	✓	✓	✓	✓
	防抖	iOS支持拍摄时防抖	✓	✓	✓	✓
	清晰度	支持标清、高清、超清拍摄，支持自定义码率、帧率、gop	✓	✓	✓	✓
	时长设置	自定义拍摄的最短和最长时长	✓	✓	✓	✓
	分段录制	拍摄过程中可以暂停分段并且可以回删	✓	✓	✓	✓
	延时拍摄	设置/开启延时拍摄，倒计时结束后自动开始录制	✓	✓	✓	✓
	背景音乐	拍摄前可以选择本地的 MP3 作为背景音。	✓	✓	✓	✓
	变速录制	支持极慢、慢、正常、快、极快等变速拍摄设置	✓	✓	✓	✓
	拍照	支持拍摄照片	✓	✓	✓	✓
拍摄美化	调色滤镜	支持内置和自定义色卡滤镜，如：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等	×	✓	✓	✓
	基础美颜	可调节美白、磨皮、大眼、瘦脸强度,满足个性需求	×	✓	✓	✓
	风格化滤镜	将实时视频进行动漫等风格化滤镜处理，如漫画风、蜡笔风、彩色和黑白铅笔画风	×	×	✓	✓
	精细化美妆	支持口红、腮红、修容、眼影（含睫毛）、眼线、眉毛、祛法令纹、祛黑眼圈等美妆子项和组合妆效	×	×	✓	✓
	脸部精准塑形	对人像进行精细化美颜，可支持大眼、瘦脸、窄脸、下颌角、颧骨、脸长、发际线、中庭、下庭、眼距、眼角、嘴大、鼻长、鼻宽等面部比例及五官的精细化调整	×	×	✓	✓
AR拍摄特效	人脸动效贴纸	基于人脸关键点定位追踪，同时支持最多4人脸检测；支持2D、3D、皮肤级、变形，素材支持27种混合模式；支持挑眉、眨眼、张嘴、点头、嘟嘴（5个）触发特效	×	×	✓	✓
	多人换脸	实现多人的人脸互换特效	×	×	✓	✓
	动漫驱动偶像	2.5D人脸驱动，融合虚拟动漫形象（如：小恶魔、娃娃脸），灵活适配脸型和面部表情动作	×	×	✓	✓
	3D人脸驱动贴纸（Dumoji）	基于人脸识别，支持人脸表情实时驱动3D面具达成表情，如：张嘴、眨眼、咧嘴等	×	×	✓	✓
	手势	支持手势识别触发和实时跟踪，支持比V、点赞、OK、单手比心、食指比1、握拳、手掌的检测（7个手势）	×	×	×	✓
	手势实时驱动	基于手部21关节点检测，实现手部驱动操控的玩法，如：操控玩偶	×	×	×	✓
	肢体	基于人体骨骼点/轮廓点精准定位和识别，实现动作检测、触发跟踪特效，亦可实现人体哈哈镜效果	×	×	×	✓
	人像分割	通过AI算法，识别视频中的人物区域，实现替换背景（支持动态）或背景虚化	×	×	×	✓
	头发分割	通过AI算法，识别视频中的头发区域，可实现染发效果（支持动态、渐变等染发效果）	×	×	×	✓
	天空分割	通过AI算法，识别视频中的天空区域，替换指定效果及素材叠加（支持动态素材）	×	×	×	✓
	天空顶	星空、粉色天空等穹顶效果，360空间氛围渲染，天空顶素材支持27种混合模式	×	×	✓	✓
	SLAM放置	通过即时定位与跟踪算法，在实景中放置3D模型动画	×	×	×	✓
	物体检测（杯子）	通过AI算法，检测画面中的物体（当前支持杯子）	×	×	×	✓
	绿幕视频	支持通过手机姿态定位，在空间中放置透明/绿幕扣背景视频素材	×	×	✓	✓
视图混编	视频/图片混编	可以使用图片、视频混合进行制作,实现MV、相册等功能	✓	✓	✓	✓
	多段拼接合成	支持在多段制作中插入、删除新的视频或图片并进行排序	✓	✓	✓	✓
	视频&图片旋转	可以旋转视频、图片的方向	✓	✓	✓	✓
	时长设定	设定视频、图片展现的时长	✓	✓	✓	✓
	封面设置	允许将视频中的任意一帧画面抽取为图片保存，可作为视频封面	✓	✓	✓	✓
	效果实时预览	添加滤镜、贴纸、转场等各种特效后支持实时预览	✓	✓	✓	✓
	背景音乐设置	支持添加背景音乐，可调节音量大小，支持背景音乐剪裁	✓	✓	✓	✓
声音特效	变声特效	支持萝莉、正太、大叔等变声特效	✓	✓	✓	✓
字幕编辑	多段字幕	一次制作可添加多个字幕，并支持多行文字，可设置每个字幕出现消失时间、画面显示位置	✓	✓	✓	✓
	字幕样式	支持设置字幕样式，包括：字体、字号、描边（粗细、透明度、颜色）、阴影（距离、透明度、颜色）等	✓	✓	✓	✓
	双语字幕	支持智能添加中英双语字幕（服务端单独计费）	✓	✓	✓	✓
编辑特效	转场特效	实现片段间过渡，带有淡入、缩放、渐变等几种基础转场	✓	✓	✓	✓
	画面特效	黑白、放大、灵魂出窍等多种特效	✓	✓	✓	✓
	分屏特效	支持三屏、六屏、九屏特效	✓	✓	✓	✓
	变速特效	支持编辑时设置极慢、慢、正常、快、极快等变速特效	✓	✓	✓	✓
主题模板	图片模板	支持按炫酷的卡点视频模板，选择图片文件，一键生成视频	×	✓	✓	✓
	视图混编模板	支持按MV视频模板，选择图片和视频文件，一键生成视频	×	✓	✓	✓
美化后处理	滤镜	可调节视频/图片的滤镜	×	✓	✓	✓
	美颜塑形	可对视频/图片进行美白磨皮处理，支持五官级的精准塑形	×	×	✓	✓
	贴纸	可对视频/图片添加人脸贴纸，支持2D、3D、皮肤级	×	×	✓	✓
合成上传	码率设置	可设置打包压缩的的码率、分辨率等	✓	✓	✓	✓
	上传云	默认上传至百度智能云BOS，用户可自定义上传路径	✓	✓	✓	✓
点播播放	点播	提供iOS、Android、web播放器，支持多清晰度切换、弹幕、缩略图等功能	✓	✓	✓	✓

如何购买

在线购买

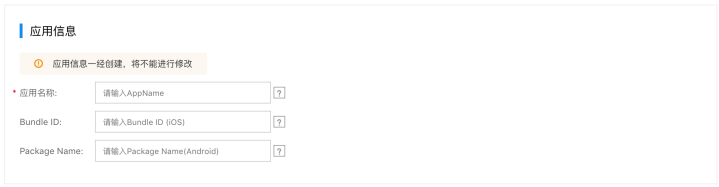
1）license购买

您可在线自助购买，购买入口为“license购买”，如下图所示，选择购买项，支付成功即可。各购买项的详细描述详见“价格说明”。
其他订单查询和退款等事项同百度云标准流程，可在官网-财务中心查看。



2）license开通

购买成功后，即可申请license进行对接集成。开通入口为“license申请”，如下图所示，填写appname、bundleID、packagename，选择能力项、贴纸，点击确定。



我们收到申请后，会在2个工作日内进行审核，审核通过后，您可在console下载license文件和SDK工程包进行对接测试。

线下购买

如果线上购买项不能满足您的需求，请提交工单，会有客户经理根据您的需求为您提供全方位服务。

工单信息需包括：

- 公司名称
- 使用场景
- 需求描述：
- 联系电话：
- userID：在百度智能云官网注册账号，登录后，在用户中心->基本信息里即可查看到；
- appname
- packagename
- bundleID

欠费说明

到期提醒

- 试用版到期前5天提醒；
- 正式版到期前1个月、前7天、前1天提醒；
- 提醒方式：站内信、短信、邮件等；

停服说明

- 当license有效期到期后，我们会在到期当日停止服务，届时所有API和能力将无法使用；
- 在停服后，系统会以短信或邮件的方式通知您。

使用指南

License申请

申请试用license

您可以免费申请短视频SDK license进行对接测试，免费周期为2个月。申请步骤如下：

- 登录百度智能云账号，并完成实名认证（不完成实名认证，无法开通）。
- 进入短视频SDK官网产品页，点击【立即使用】，并开通短视频SDK产品。

申请正式license

试用完成后，若您想由试用转成正式的继续使用，请联系您的客户经理或者提交工单申请。 在商务合同/付款等事宜确定后，您可在console里选择授权类型【正式】，弹框里选择商务确定的购买周期，点击确认，我们的运营人员审核通过后即可生效。

短视频SDK申请

- 点击【license申请】，进入新建页面，输入应用名称、BundleID（IOS）或PackageName（Android），【套餐选择】可以选择短视频套餐，还可单独【AR特效选择】勾选所需的AR能力，勾选所需的贴纸，授权信息勾选"试用"，点击【立即申请】。套餐所包含的功能项、贴纸数、价格等详见【价格说明】。

- 百度智能云运营人员收到申请后会在2个工作日内审核完成，审核通过后，您可以在console里下载license文件和SDK工程包（包含SDK，demo源码，接入说明文档）进行集成测试。

AR特效SDK申请

- 点击【license申请】，进入新建页面，输入应用名称、BundleID（IOS）或PackageName（Android），不用选择套餐，【AR特效选择】勾选所需要的AR能力，勾选所需的贴纸，授权信息勾选"试用"，点击【立即申请】。AR特效对应的功能项、价格等详见【价格说明】。

- 百度智能云运营人员收到申请后会在2个工作日内审核完成，审核通过后，您可以在console里下载license文件和SDK工程包（包含SDK，demo源码，接入说明文档）进行集成测试。

开发指南

iOS开发说明

iOS版本开发接入文档

一、概述

百度云短视频产品(SDK)专注移动端视音频场景研发，提供端到端的一站式视音频技术解决方案，不限于采集、录制、合成、上传、存储、分发，极大降低客户接入音视频产品的技术门槛。 **1.1 注意事项 1.1.1 头文件引用说明** 本SDK使用C++开发，对于引用<BDCloudAVContext/BDCloudAVContext.h>头文件的类文件，需要将.m文件修改成.mm文件,以适配C++。 **1.1.2 运行环境** iOS9系统以上，Xcode->General->Depolyment Info->Depolyment Target

二、快速接入

请从百度云开发者中心下载最新版本拍摄器。

在使用拍摄器SDK，需要申请产品对应的授权文件，如无授权，产品无法正常使用。 申请成功后，会得到一个licenseID和对应授权文件下载地址，用户下载成功后，需要手动添加到项目工程中 注意：授权文件后缀名为.license。

三、使用说明

3.1 录制设置

录制的相关接口是在BDCloudAVStreamContext类里，包括采集预览(startPreview)，录制(startRecording:)，添加视频美颜(applyBeautyBaseVideoFx)等。注意：百度云拍摄器SDK所有的类都是以“BDCloud”开头。

3.1.1 BDCloudAVStreamContext类 BDCloudAVStreamContext是拍摄器SDK的流媒体上下文类，是接入拍摄器SDK产品的入口。开始使用前，需要先初始化BDCloudAVStreamContext类，注意BDCloudAVStreamContext是单例类。

BDCloudAVStreamContext初始化代码如下

`_avStreamContext = [BDCloudAVStreamContext sharedInstance];` BDCloudAVStreamContext销毁代码如下

```
[_avStreamContext destroyInstance];
_avStreamContext = nil;
```

注意：BDCloudAVStreamContext初始化后，需要调用verifySDKLicense: completionHandler:验证客户的合法性，其中appid即用户申请的licenseID，若未申请授权，请参考快速接入

3.1.2 BDCloudAVStreamSettings类配置

BDCloudAVStreamSettings是拍摄器SDK配置项，支持客户调整摄像头位置、摄像头类型、视频维度（分辨率）、视频帧速率、视频帧码率及手电筒、对焦、曝光补光、预览放大缩小、预览模式。

考虑到配置项较多且偏专业，拍摄器SDK提供默认配置项，代码如下：

```
BDCloudAVStreamSettings *settings = [BDCloudAVStreamSettings defaultSettings];
```

3.1.3 预览前设置

完成流媒体上下文初始化及配置项后，下面就要创建预览前配置，创建拍摄器及预览画面，代码如下：

```
 // 使用配置项创建拍摄器
_avStreamContext = [_avStreamContext initWithCaptureConfig:settings];
// 获取拍摄器预览界面
UIView *preview = _avStreamContext.view;
// 设置预览界面展示区域
preview.frame = self.view.bounds;
// 设置拍摄器回调代理
_avStreamContext.delegate = self;
// 将预览界面添加到用户控制器上
[self.view insertSubview:preview atIndex:0];
```

3.1.4 启动预览

启动预览对应startPreview接口，此接口发检查用户照相机和麦克风系统权限。

3.1.5 录制与停止录制

调用录制接口前startRecording: recordCallBack:，需要先创建录制文件的路径，建议用户使用.MOV和.mp4文件。录制接口在写入文件的同时，会不断回调给使用方，用户需要关心回调状态BDCloudAVStreamFileOutputState。

代码示例如下：

```
    [[BDCloudAVStreamContext sharedInstance] startRecording:recordVideoPath recordCallBack:^(BDCloudAVStreamFileOutputState state, double recordSecond, NSError *error) {

    switch (state) {

        case BDCloudAVStreamFileOutputStateStarting:
            break;
        case BDCloudAVStreamFileOutputStateStarted:
            break;
        case BDCloudAVStreamFileOutputStateCancel:
            break;
        case BDCloudAVStreamFileOutputStateError:
            //错误信息
            break;
        case BDCloudAVStreamFileOutputStateEnding:
            //录制结束
            break;
        case BDCloudAVStreamFileOutputStateEnded:
            break;
        default:
            break;
    }

}];
```

停止录制，代码示例如下：

```
[[BDCloudAVStreamContext sharedInstance] stopRecording];
```

3.1.6 录制设置

设置闪光灯是否开启，代码示例如下:

```
    AVCaptureFlashMode      _flashMode;
_flashMode = _flashMode == AVCaptureFlashModeOff ? AVCaptureFlashModeOn : AVCaptureFlashModeOff;
    [[BDCloudAVStreamContext sharedInstance] toggleFlash:_flashMode];
    if (_flashMode == AVCaptureFlashModeOff) {
        [[BDCloudAVStreamContext sharedInstance] toggleTorch:NO];
    }else {
        [[BDCloudAVStreamContext sharedInstance] toggleTorch:YES];
    }
```

设置自动对焦

```
    [[BDCloudAVStreamContext sharedInstance] setCameraContinuousAutofocus:YES];
    [[BDCloudAVStreamContext sharedInstance] setCameraFocusPointOfInterest:[BDCloudAVStreamContext sharedInstance].view.center];
```

设置曝光点

```
    [[BDCloudAVStreamContext sharedInstance] setCameraExposurePointOfInterest:[BDCloudAVStreamContext sharedInstance].view.center];
```

设置缩放

```
    [[BDCloudAVStreamContext sharedInstance] setCameraZoomFactor:1.3];
```

3.2 美颜设置

添加美颜(applyBeautyBaseVideoFx)特效后，在预览窗口就可以看到美颜效果。录制视频时，用户需要根据手机性能任意选择带美颜录制或者不带美颜录制。 美颜特效分为基础美颜(美白、磨皮)和高级美颜(大眼、瘦脸)。

开启美颜，代码示例如下：

```
//开启美颜
[_avStreamContext applyBeautyBaseVideoFx];
//调整美白
[[BDCloudAVStreamContext sharedInstance] adjustBeautyWhiteLevel:newValue];
//调整磨皮
[[BDCloudAVStreamContext sharedInstance] adjustBeautyBlurLevel:newValue];
//调整大眼
[[BDCloudAVStreamContext sharedInstance] adjustBeautyEnlargingLevel:newValue];
//调整瘦脸
[[BDCloudAVStreamContext sharedInstance] adjustBeautyThinningLevel:newValue];
```

注意：需要在初始化拍摄器后开启美颜。 使用高级美颜时，需要申请对应权限，免费版本不支持此功能

3.3 滤镜设置

开启滤镜(applyEffect:)特效后，在预览窗口就可以看到滤镜效果。

代码示例如下：

`[[BDCloudAVStreamContext sharedInstance] applyEffect:filterID];` 注意：当前filterID仅支持内置滤镜使用，用户在申请授权成功后，会得到滤镜资源包，具体使用可以参考短视频SDK DemoD拍摄模块

3.4 贴纸设置

开启滤镜(applyLocalStickerVideoFx: stickerModelPath: stickerType: stickerIdentify:)特效后，在预览窗口就可以看到贴纸效果。

代码示例如下：

```
    // 内置贴纸
    NSString *name = [NSString stringWithFormat:@"AR.bundle/%@.zip", stickerID];

    NSString *path = [[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:name];

    BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path];

    //解压

    if (isExist) {

        NSString *destPath = [self localPath];

        destPath = [destPath stringByAppendingPathComponent:stickerID];

        BOOL bSucess = [self unzipStickerPackage:path destPath:destPath];

        if (bSucess) {
            // 开启贴纸
            [[BDCloudAVStreamContext sharedInstance] applyLocalStickerVideoFx:destPath stickerModelPath:@"" stickerType:@"10" stickerIdentify:@"10273"];

        }

    } else {
            // 关闭贴纸
        [[BDCloudAVStreamContext sharedInstance] disabledStickerVideoFx];

    }
```

快速接入

鉴权认证

在使用拍摄器SDK前，需要申请产品对应的授权文件，如无授权，产品无法正常使用。

申请成功后，会得到一个licenseID和对应授权文件下载地址，用户下载后，需要手动添加到项目工程中。

licenseID，在调用SDK授权接口时传入。

> 注意：授权文件后缀名为.license。

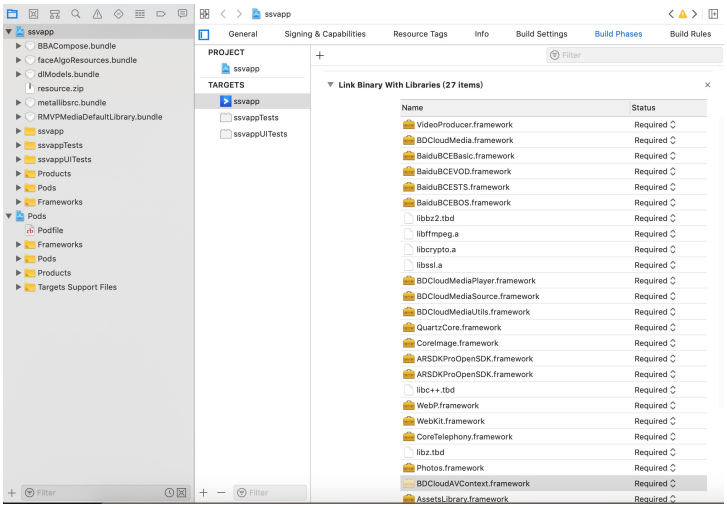
依赖包导入

目前短视频SDK提供如下依赖包，需要加入到您的开发项目中。

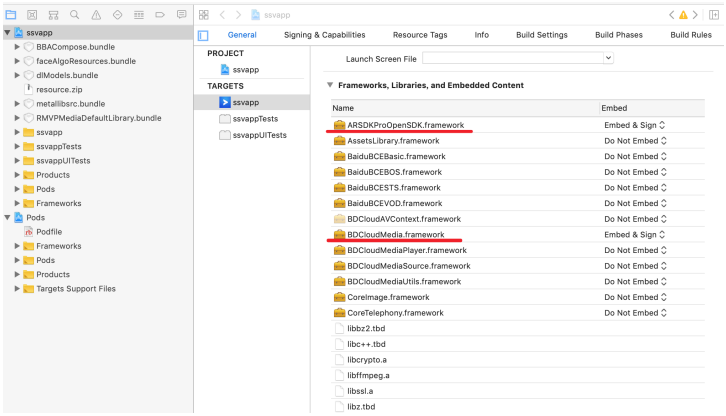
- BDCloudAVContext.framework 百度云拍摄SDK 静态包
- BDCloudMedia.framework 百度云媒体SDK 动态库
- VideoProducer.framework 百度云编辑SDK 静态包
- ARSDKProOpenSDK.framework 百度ARSDK 动态包
- BaiduBCEBasic\VOD\STS\BOS.framewrok 百度BOS\VOD\上传SDK 静态库
- libcrypto.a libssl.a openssl静态库

其中：

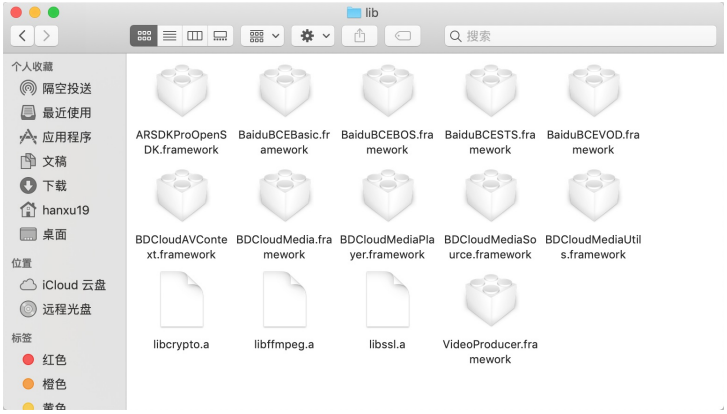
- BDCloudAVContext.framework、VideoProducer.framework、BaiduBCEBasic\VOD\STS\BOS.framewrok、libcrypto.a、libssl.a 需要添加到 Xcode > Build Phases > Link Binary With Libraries中。



- ARSDKProOpenSDK.framework BDCloudMedia.framework需要添加到Xcode > General > Frameworks,Libraries,and Embedded Content中



解压SDK包后，进入ssvapp > lib中，即可找到上述文件。



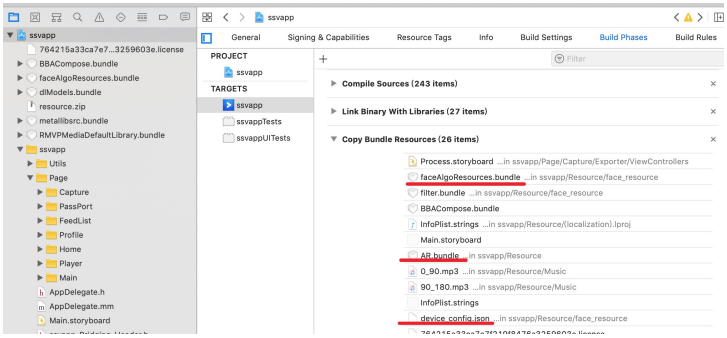
算法相关

- faceAlgoResources.bundle
- device_config.json

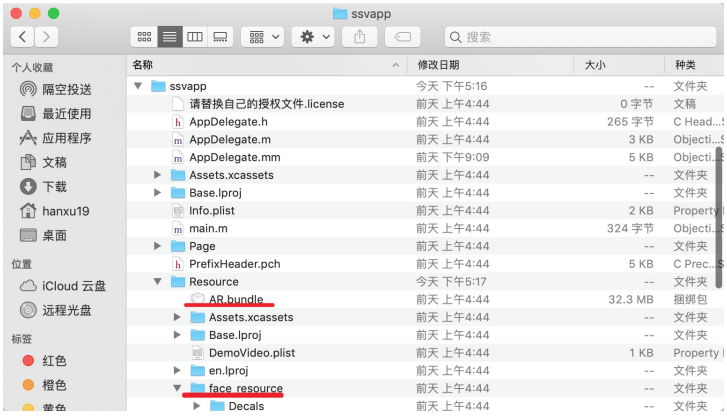
贴纸和滤镜相关

- AR.bundle

以上资源需要加载到Xcode > Build Phases > Copy Bundle Resources中，否则美颜、滤镜、贴纸无法使用。



解压SDK包后，进入ssvapp > ssvapp >Resources中，即可找到上述文件。



另外AR.bundle跟您的SDK版本类型相关，若您手中的SDK包里没有请联系百度云或对应商务。

合成

- 合成模块负责将编辑后的视频导出成本地视频

导出预览视频

- 导出预览视频，将媒体轨道中心的所管理的视频、音频、字幕导出本地，默认是.mp4文件。导出时，需要暂停预览
- 代码示例如下：

```
 //暂停预览
[self.previewer pausePreview];

    NSDate *currentDate = [NSDate date];

    double timeStamp = [currentDate timeIntervalSince1970];

    NSString *videoName = [[NSString stringWithFormat:@"%f",timeStamp] stringByAppendingString:@".mp4"];


NSString *documentsDirPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) firstObject];

    NSString *videoDirPath = [documentsDirPath stringByAppendingPathComponent:@"video_export"];

    [[NSFileManager defaultManager] createDirectoryAtPath:videoDirPath withIntermediateDirectories:YES attributes:nil error:nil];

    NSURL *documentsDirUrl = [NSURL fileURLWithPath:videoDirPath isDirectory:YES];

    NSURL *output = [NSURL URLWithString:name relativeToURL:documentsDirUrl];

    RMVPVideoEditConfig *config = [RMVPMediaConfigManager defaultEditConfig];

    self.mediaExporter = [[RMVPMediaExporter alloc] initWithMediaTracksCenter:self.videoTrack outputURL:output config:config];

    [self.mediaExporter startExportWithProgress:^(CGFloat progress) {

        NSLog(@"media - exporter - progress %f", progress);

    } completion:^(BOOL success, NSError *error) {

        if (success) {

            NSLog(@"media - exporter - success");

            dispatch_async(dispatch_get_global_queue(0, 0), ^{

                [[PHPhotoLibrary sharedPhotoLibrary] performChanges:^{

                    [PHAssetChangeRequest creationRequestForAssetFromVideoAtFileURL:output];

                } completionHandler:^(BOOL success, NSError * _Nullable error) {

                    if (success) {

                        [self.previewer resumePreview];

                    }

                }];

            });

        } else {
        }

    }];
```

转场

- 转场的相关接口在VideoProducer.framewrok中RMVPMediaTrack.h头文件。

添加基础转场

- 添加基础转场，目前支持淡入、闪黑、闪白、模糊、横滑、纵滑。添加转场，需要传入媒体轨道上插入点(即，片段相应位置，这里可参考转场UI逻辑)
- 代码示例如下：

```
 //创建转场效果实例，这里以读取本地配置文件为例
NSString *name = [NSString stringWithFormat:@"bundle/%@.zip",转场唯一签名];

  NSString *path = [[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:name];

  BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path];

  //解压

  if (isExist) {

     NSString *destPath = [self localPath];

     destPath = [destPath stringByAppendingPathComponent:item.sign];

     BOOL bSucess = [self unzipStickerPackage:path destPath:destPath];

     if (bSucess) {

        // 读取json文件

        NSString *jsonPath =  [destPath stringByAppendingPathComponent:@"transition_config.json"];

        BOOL isJsonExist = [[NSFileManager defaultManager] fileExistsAtPath:jsonPath];

        if (isJsonExist) {

           NSData *data=[NSData dataWithContentsOfFile:jsonPath];

           NSError *error;

           NSDictionary *jsonDict =[NSJSONSerialization JSONObjectWithData:data

                                                                   options:NSJSONReadingAllowFragments

                                                                     error:&error];

           if(jsonDict != nil && jsonDict.count > 0) {

              NSString *library_name      = @"library_name";

              NSString *vertex_function   = @"vertex_function";

              NSString *fragment_function = @"fragment_function";

              NSString *durationKey       = @"duration";

              NSString *library = [jsonDict objectForKey:library_name];

              NSString *libraryPath = [destPath stringByAppendingPathComponent:library];

              if (![[NSFileManager defaultManager] fileExistsAtPath:libraryPath]) {

                 return nil;

              }

              NSString *vertexShaderName = [jsonDict objectForKey:vertex_function];

              NSString *fragmentShaderName = [jsonDict objectForKey:fragment_function];

              CGFloat defaultDuration = 0.5 * 1000;

              CGFloat duration = [jsonDict hk_floatForKey:durationKey];

              if (duration <0.01) {

                 duration = defaultDuration;

              }
   //创建转场效果实例
              RMVPMediaVideoTransitionItem *mediaItem = [[RMVPMediaVideoTransitionItem alloc] initWithType:RMVPMediaVideoTransitionItemTypeDual
transitionType:RMVPMediaVideoTransitionItemTypeTransitionTypeDefault libraryPath:libraryPath vertexFunctionName:vertexShaderName fragmentFunctionName:fragmentShaderName];

              mediaItem.duration = CMTimeMakeWithSeconds(duration / 1000, NSEC_PER_SEC);

              mediaItem.transitionId = item.transition_id;

           }

        }

     }

  }


 //删除原有转场效果
 [self.videoTrack removeTransitionItemAtIndex:index];


 //绑定最新转场效果
 [self.videoTrack bindTransitionItem:mediaItem atIndex:index];
```

字幕

创建字幕轨道

- 创建字幕轨道，使用字幕相关功能，需要创建字幕轨并添加到媒体轨道中心，并创建一个字幕片段，配合字幕UI逻辑使用，详见智能小视频源码,涉及组件BDHKVlogSubtitlesView(字幕位置)，BDMVSubtitleInputAccessoryView(字幕输入框)，BDMVInputEventBottomBar(字幕确认框)
- 代码示例如下：

```
  //创建字幕轨道
 RMVPMediaVideoBlendItem *blendItem1 = [[RMVPMediaVideoBlendItem alloc] initWithBlendMode:RMVPMediaVideoBlendModeNormal];

      subtitleTrack = [[RMVPMediaVisibleTrack alloc] initWithBlendItem:blendItem1];

      subtitleTrack.trackId = kBDMVMediaVisibleSubtitleTrackId;

      [self.videoTrack addVideoTrack:subtitleTrack];
 //添加字幕片段
      RMVPMediaTextSegment *subtitleSegment = [[RMVPMediaTextSegment alloc] init];

      [subtitleTrack addSegment: subtitleSegment];


 //配置字幕与预览实时显示
  CGFloat videoWidth = self.previewer.previewConfig.width;

    CGFloat videoHeight = self.previewer.previewConfig.height;

   [self.videoTrack setupSubtitleStyleWithPreviewSize:CGSizeMake(videoWidth, videoHeight)];
```

添加多段字幕

- 添加多段字幕，请参考智能小视频编辑预览BBAShortVideoPreviewViewController.mm文件中的字幕逻辑，字幕默认显示2S
- 代码示例如下：

```
 //获取字幕轨道中字幕片段
RMVPMediaTextSegment *textSegment = [self.videoTrack subtitleSegment];

//设置字幕内容，获取媒体轨道中心当前时间
  CMTime currentTime = self.previewer.cursor.cursorTime;


//新增字幕
if (!currentItem) {

    currentItem = [[RMVPMediaTextItem alloc] init];

    currentItem.mainText = self.tmpMainTitle;

    currentItem.subText = self.tmpSubTitle;

    RMVPMediaTextItem *nextTextItem = nil;

    NSArray *textAllItems = [textSegment textAllItem];

    for (RMVPMediaTextItem *tmpItem in textAllItems) {

        CMTimeRange timeRange = tmpItem.timeRange;
        if (CMTimeCompare(timeRange.start, currentTime) > 0) {
            nextTextItem = tmpItem;
            break;
        }
    }

    CGFloat newSubtitleDuration = [BDMVEditSubtitleSettings sharedInstance].addSubtitleDuration;

    if (newSubtitleDuration <= 0) newSubtitleDuration = 2;

    if (nextTextItem) {

        CMTime eventualDuration;

        CMTime nextItemStartTime = nextTextItem.timeRange.start;

        CMTime gapTime = CMTimeSubtract(nextItemStartTime, currentTime);

        if (CMTimeCompare(gapTime, CMTimeMake(newSubtitleDuration, 1)) >= 0) {
            eventualDuration = CMTimeMake(newSubtitleDuration, 1);
        } else {
            eventualDuration = gapTime;
        }

        currentItem.timeRange = CMTimeRangeMake(currentTime, eventualDuration);

        NSInteger index = [textSegment textIndexAtRefTime:nextItemStartTime];
 //插入字幕到指定位置
        [textSegment insertTextAtIndex:index item:currentItem];

        [self.subtitleEditManager.seekBarAera insertSubtitleAtIndex:index];

    } else {

        CMTime eventualDuration;

        CMTime mediaDuration = self.tracksCenter.userTrack.trackDuration;

        CMTime remainingDuration = CMTimeSubtract(mediaDuration, currentTime);

        if (CMTimeCompare(remainingDuration, CMTimeMake(newSubtitleDuration, 1)) >= 0) {

            eventualDuration = CMTimeMake(newSubtitleDuration, 1);

        } else {

            eventualDuration = remainingDuration;

        }

        currentItem.timeRange = CMTimeRangeMake(currentTime, eventualDuration);
 //追加字幕
        [textSegment addTextWithItem:currentItem];

        [self.subtitleEditManager.seekBarAera addSubtitle];

    }

    NSInteger index = [textSegment textIndexAtRefTime:self.previewer.cursor.cursorTime];
//更新字幕显示位置，详见[3.3.4 设置字幕画面位置]
    [self updateSubtitleDisplayAndRealSubtitlePositonWithTransition:SubtitleInitTranslation itemIndex:index];

 } else { // 编辑字幕

    currentItem.mainText = self.tmpMainTitle;

    currentItem.subText = self.tmpSubTitle;

    NSInteger index = [textSegment textIndexAtRefTime:currentTime];

    [self.subtitleEditManager.seekBarAera editSubtitleAtIndex:index];
 }
```

设置字幕画面位置

- 设置字幕画面位置，可以调整字幕轨道上某个字幕的显示位置
- 代码示例如下：

```
 //设置字幕显示位置
currentFrame = self.displaySubtitleTextView.frame;//displayView的位置已经更新

CGSize scale = [self currentVideoScale];

BDMVPreviewTransform transform = self.subtitlePreviewTransform;

CGFloat left = (CGRectGetMinX(currentFrame) - CGRectGetMinX(availabelRect) + Subtitle_InputView_OffsetX) / transform.scale;

left += [self currentVideoInvisibleHorizontalPadding];

CGFloat bottom = (CGRectGetMaxY(availabelRect) - CGRectGetMaxY(currentFrame) + Subtitle_InputView_OffsetX) / transform.scale;

CGFloat right = (CGRectGetMaxX(availabelRect) - CGRectGetMaxX(currentFrame) + Subtitle_InputView_OffsetX) / transform.scale;

right += [self currentVideoInvisibleHorizontalPadding];

RMVPMediaTextSegment *currentSegment = [self.tracksCenter subtitleSegment];

NSDictionary *attributes = @{@"marginBottom":@(bottom * scale.height),

                             @"marginLeft":@(left * scale.width),

                             @"marginRight":@(right * scale.width)};

[currentSegment updateStyleAtIndex:index styleAttributes:attributes];
```

编辑

- 编辑的相关接口在VideoProducer.framewrok中。

视频预览

- 完成拍摄或选取本地视频，将实时预览窗口与创建的媒体轨道中心(MediaTrackCenter)连接，对已完成拍摄的视频或本地视频根据需要进行编辑，然后生成视频输出.mov格式的文件
- 代码示例如下：

```
 //初始化预览视图

 self.previewer = [[RMVPMediaPreviewer alloc] init];

 self.previewer.displayView.frame = CGRectMake(65, 9, SCREEN_WIDTH - 65*2, SCREEN_HEIGHT - 9 - 221);

 self.previewer.displayView.backgroundColor = [UIColor blackColor];

 self.previewer.delegate = self;

 self.previewer.needRepeatPlay = YES;

 [self.view addSubview:self.previewer.displayView];


//需要视频原始尺寸

 RMVPMediaPreviewConfig *previewConfig = [RMVPMediaPreviewConfig defaultConfig];

 previewConfig.width = 720;

 previewConfig.height = 1280;

 previewConfig.gravity = RMVPMediaPreviewGravityAspectFit;

 self.previewer.previewConfig = previewConfig;
```

插入视频或图片

- 编辑阶段中的视频，称之为片段(segment)，由媒体轨道中心(MediaTrackCenter)管理，负责插入、删除，插入视频完成后，再与配置好的预览参数进行连接，就可以实时预览页面。
- 代码示例如下：

```
 //创建媒体轨道中心
self.trackCenter = [[RMVPMediaTracksCenter alloc] init];

  RMVPMediaVideoBlendItem *blendItem = [[RMVPMediaVideoBlendItem alloc] initWithBlendMode:RMVPMediaVideoBlendModeNormal];

  self.videoTrack = [[RMVPMediaVisibleTrack alloc] initWithBlendItem:blendItem];


//创建视频片段(segment)
RMVPMediaVideoSegment *segment = [[RMVPMediaVideoSegment alloc] initWithVideoAsset:tempAVAsset decoderType:RMVPMediaSegmentDecoderTypeAVPlayer];

  [self.videoTrack addSegment:segment];


//插入视频到媒体轨道中心
[self.trackCenter addVideoTrack: self.videoTrack];


//连接编辑预览与媒体轨道中心
[self.previewer loadTracksCenter:self.trackCenter];


//开始预览
[self.previewer resumePreview];
```

素材删除

- 片段/素材删除，片段成功添加到轨道中心后，通过传入片段索引，可以移除轨道中心上对应素材
- 代码示例如下：

```
 //删除轨道中心索引对应视频
[self.videoTrack removeSegmentAtIndex:index];


//重新链接预览与轨道中心
[self.previewer loadTracksCenter:self.trackCenter];


//重新计算轨道视频长度
[self.previewer seekToTime:kCMTimeZero completionBlock:^(BOOL success) {

      [self.previewer resumePreview];

  }];
```

视频裁剪

- 视频裁剪，通过设置视频出点和入点，基于时间出入点裁剪视频，此接口同样适应于音乐裁剪
- 代码示例如下：

```
 // 拖拽片段头, 向后拖拽 2 秒钟
CMTime deltaTime = CMTimeMakeWithSeconds(2, NSEC_PER_SEC);

[self.videoTrack clipSegmentAtIndex:draggingSegmentIndex forTime:deltaTime

position:RMVPMediaTrackClipSegmentPositionPre];


// 拖拽片段尾部, 向前拖拽 3 秒钟
CMTime deltaTime = CMTimeMakeWithSeconds(-3, NSEC_PER_SEC);

[self.videoTrack clipSegmentAtIndex:draggingSegmentIndex forTime:deltaTime

position:RMVPMediaTrackClipSegmentPositionPost];
```

视频时码线

- 视频时码线，素材\片段添加到媒体轨道后，会抽取片段缩略图形式平铺显示，这里使用系统级接口完成，主要提供两个UI组件，一个用于设置视频裁剪出点、入点视频进度条组件。另一个是抽取缩略图组件。
- 代码示例如下：

```
 //初始化视频进度条组件
//segments当前媒体轨道中心的片段数据集合
_progressBar = [[BDMVPhotoVideoProgressBar alloc] initWithFrame:CGRectMake(0, 0, SCREEN_WIDTH, 58) photoVideoSegments:self.segments totalDuration:duration minDuration:3 maxSellectDuration:60];
// 设置代理，用户接收用户拖动行为
  _progressBar.delegate = self;
[self.view addSubview:_progressBar];
//BDMVPhotoVideoProgressBar中会调用抽取缩略图组件
//计算抽取缩略图个数
//默认缩略图分辨率{38,50}，最大缩略图分辨率{100,100}。视频最大长度不超过60s
```

视频截图

- 视频截图，通过传递参数截取指定媒资的在某时间上的图片

```
UIImage *image = [BDMVVideoClipViewController buildImageWithAsset:newSegment.asset maximunSize:CGSizeMake(200.f,200.f) time:CMTimeMake(1, 10)];

+ (UIImage *)buildImageWithAsset:(AVAsset *)avasset maximunSize:(CGSize)size time:(CMTime)time {
    if (avasset) {
        AVAssetImageGenerator *imageGenerator = [AVAssetImageGenerator assetImageGeneratorWithAsset:avasset];
        imageGenerator.appliesPreferredTrackTransform = YES;
        if (!CGSizeEqualToSize(size, CGSizeZero)) {
            imageGenerator.maximumSize = size;
        }
        if (CMTIME_IS_INVALID(time) || CMTimeCompare(time, kCMTimeZero) == 0) {
            time = CMTimeMake(1, 10);
        }
        NSError *error = nil;
        CGImageRef cgimage = [imageGenerator copyCGImageAtTime:time actualTime:nil error:&error];

        UIImage *image = [UIImage imageWithCGImage:cgimage];
        CGImageRelease(cgimage);
        return image;
    }
    return nil;
}
```

音乐剪辑

- 支持将选择的音乐设定入点，进入音乐剪辑页面

```
// 选择的音乐
BDMVComposeMusicItem *item = self.audioConfig.musicItem;

[item setupNetMusicLocalPath];

BDMVMusicClipViewController *vc = [[BDMVMusicClipViewController alloc] initWithItem:item];

vc.delegate = self;

[self presentViewController:vc animated:YES completion:^{}];
```

创建剪辑面板

- 剪辑面板是UI页面，可添加到指定父页面中，用于用户选择入点

```
self.waveView = [[BDMVWaveformAbstractView alloc] initWithFrame:CGRectMake(16, 84, self.view.frame.size.width - 32, 52)];

self.waveView.delegate = self;

NSURL *musicUrl = [NSURL fileURLWithPath:self.musicItem.localPathString];

AVAsset *musicAsset = [AVAsset assetWithURL:musicUrl];

self.waveView.asset = musicAsset;

// MusicClipView任意父页面
[MusicClipView addSubview:self.waveView];
```

设置剪辑入点

- 在选择的音乐基础上(通过初始化BDMVMusicClipViewController类传入)，设置入点

```
self.musicItem.clipStartPos = time;

self.musicItem.clipOffset = self.startPos;
```

录制

录制的相关接口是在BDCloudAVStreamContext类里，包括采集预览(startPreview)，录制(startRecording:)，添加视频美颜(applyBeautyBaseVideoFx)等。

注意：百度云拍摄器SDK所有的类都是以"BDCloud"开头。

BDCloudAVStreamContext类

BDCloudAVStreamContext是拍摄器SDK的流媒体上下文类，是接入拍摄器SDK产品的入口。开始使用前，需要先初始化BDCloudAVStreamContext类，注意BDCloudAVStreamContext是单例类。

BDCloudAVStreamContext初始化代码如下

```
_avStreamContext = [BDCloudAVStreamContext sharedInstance];
```

BDCloudAVStreamContext销毁代码如下

```
 [_avStreamContext destroyInstance];
_avStreamContext = nil;
```

注意：BDCloudAVStreamContext初始化后，需要调用verifySDKLicense: completionHandler:验证客户的合法性，其中appid即用户申请的licenseID，若未申请授权，请参考快速接入

BDCloudAVStreamSettings类配置

BDCloudAVStreamSettings是拍摄器SDK配置项，支持客户调整摄像头位置、摄像头类型、视频维度（分辨率）、视频帧速率、视频帧码率及手电筒、对焦、曝光补光、预览放大缩小、预览模式。

考虑到配置项较多且偏专业，拍摄器SDK提供默认配置项，代码如下：

```
BDCloudAVStreamSettings *settings = [BDCloudAVStreamSettings defaultSettings];
```

预览前设置

完成流媒体上下文初始化及配置项后，下面就要创建预览前配置，创建拍摄器及预览画面，代码如下：

```
 // 使用配置项创建拍摄器
  _avStreamContext = [_avStreamContext initWithCaptureConfig:settings];
// 获取拍摄器预览界面
  UIView *preview = _avStreamContext.view;
// 设置预览界面展示区域
  preview.frame = self.view.bounds;
// 设置拍摄器回调代理
  _avStreamContext.delegate = self;
// 将预览界面添加到用户控制器上
  [self.view insertSubview:preview atIndex:0];
```

启动预览

启动预览对应startPreview接口，此接口发检查用户照相机和麦克风系统权限。

录制与停止录制

录制

- 调用录制接口前startRecording: recordCallBack:，需要先创建录制文件的路径，建议用户使用.MOV和.mp4文件
- 录制接口在写入文件的同时，会不断回调给使用方，用户需要关心回调状态BDCloudAVStreamFileOutputState。
- 代码示例如下：

```
[[BDCloudAVStreamContext sharedInstance] startRecording:recordVideoPath recordCallBack:^(BDCloudAVStreamFileOutputState state, double recordSecond, NSError *error) {

    switch (state) {

        case BDCloudAVStreamFileOutputStateStarting:
break;
        case BDCloudAVStreamFileOutputStateStarted:
            break;
        case BDCloudAVStreamFileOutputStateCancel:
            break;
        case BDCloudAVStreamFileOutputStateError:
            //错误信息
            break;
        case BDCloudAVStreamFileOutputStateEnding:
            //录制结束
            break;
        case BDCloudAVStreamFileOutputStateEnded:
            break;
        default:
            break;
    }

}];
```

停止录制

- 代码示例如下：

```
[[BDCloudAVStreamContext sharedInstance] stopRecording];
```

录制设置

- 设置闪光灯是否开启，代码示例如下:

```
AVCaptureFlashMode      _flashMode;
    _flashMode = _flashMode == AVCaptureFlashModeOff ? AVCaptureFlashModeOn : AVCaptureFlashModeOff;
    [[BDCloudAVStreamContext sharedInstance] toggleFlash:_flashMode];
    if (_flashMode == AVCaptureFlashModeOff) {
        [[BDCloudAVStreamContext sharedInstance] toggleTorch:NO];
    }else {
        [[BDCloudAVStreamContext sharedInstance] toggleTorch:YES];
    }
```

- 设置自动对焦

```
[[BDCloudAVStreamContext sharedInstance] setCameraContinuousAutofocus:YES];
    [[BDCloudAVStreamContext sharedInstance] setCameraFocusPointOfInterest:[BDCloudAVStreamContext sharedInstance].view.center];
```

- 设置曝光点

```
[[BDCloudAVStreamContext sharedInstance] setCameraExposurePointOfInterest:[BDCloudAVStreamContext sharedInstance].view.center];
```

- 设置缩放

```
[[BDCloudAVStreamContext sharedInstance] setCameraZoomFactor:1.3];
```

开启美颜

- 添加美颜(applyBeautyBaseVideoFx)特效后，在预览窗口就可以看到美颜效果。录制视频时，用户需要根据手机性能任意选择带美颜录制或者不带美颜录制。 美颜特效分为基础美颜(美白、磨皮)和高级美颜(大眼、瘦脸)。
- 开启美颜，代码示例如下：

```
 //开启美颜
[_avStreamContext applyBeautyBaseVideoFx];
//调整美白
[[BDCloudAVStreamContext sharedInstance] adjustBeautyWhiteLevel:newValue];
//调整磨皮
[[BDCloudAVStreamContext sharedInstance] adjustBeautyBlurLevel:newValue];
//调整大眼
[[BDCloudAVStreamContext sharedInstance] adjustBeautyEnlargingLevel:newValue];
//调整瘦脸
[[BDCloudAVStreamContext sharedInstance] adjustBeautyThinningLevel:newValue];
```

- 注意，开启美颜需要在初始化拍摄器后。
- 注意：使用高级美颜时，需要申请对应权限，免费版本不支持此功能

开启滤镜

- 开启滤镜(applyEffect:)特效后，在预览窗口就可以看到滤镜效果。
- 代码示例如下：

```
[[BDCloudAVStreamContext sharedInstance] applyEffect:filterID];
```

- 注意：当前filterID仅支持内置滤镜使用，用户在申请授权成功后，会得到滤镜资源包，具体使用可以参考短视频SDK DemoD拍摄模块

开启贴纸

- 开启滤镜(applyLocalStickerVideoFx: stickerModelPath: stickerType: stickerIdentify:)特效后，在预览窗口就可以看到贴纸效果。注：【v3.0.0】版本后，贴纸不在支持内置使用，详见下方下载贴纸
- 代码示例如下：

```
- (void)setLocalSticker:(NSString *)stickerID andSubtype:(NSString *)subType withModel:(NSString *)model gestureSupport:(BOOL)isSupport {
    dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
    dispatch_async(queue, ^{
    if (stickerID == nil || stickerID.length == 0) {
        [[BDCloudAVStreamContext sharedInstance] disabledStickerVideoFx];
    } else {
        [[BDCloudAVStreamContext sharedInstance] disabledStickerVideoFx];
        NSString *path       = [BDSSVFileUtils getDocumentPath]; //大文件放在沙盒下的Library/Caches
        NSString *finishPath = [NSString stringWithFormat:@"%@/Stickers", path]; //保存解压后文件的文件夹的路径
        NSString *resultpath = [NSString stringWithFormat:@"%@/%@.zip", finishPath, stickerID]; //下载的zip包存放路径
        //模型地址
        NSString *modelpath =  [NSString stringWithFormat:@"%@/%@.zip", finishPath, model]
        BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:resultpath];
        BOOL isModelExist = [[NSFileManager defaultManager] fileExistsAtPath:modelpath];
        //解压
        if (isExist) {
            NSString *localPath = [self localPath];

            //资源地址
            NSString *resultDestPath = [localPath stringByAppendingPathComponent:stickerID];
            BOOL bSucess = [self unzipStickerPackage:resultpath destPath:resultDestPath];
            NSString *modelDestPath = [localPath stringByAppendingPathComponent:[NSString stringWithFormat:@"%@-model", stickerID]];
            BOOL bModelSucess = NO;
            if (isModelExist) {
                bModelSucess = [self unzipStickerPackage:modelpath destPath:modelDestPath];
            }
            if (!bModelSucess) {
                modelDestPath = @"";
            }
            if (bSucess) {
                if (isSupport) {
                    [[BDCloudAVStreamContext sharedInstance] applyLocalStickerVideoFx:resultDestPath stickerModelPath:modelDestPath stickerType:subType stickerIdentify:@"10273" upportGesture:YES];
                } else {
                    [[BDCloudAVStreamContext sharedInstance] applyLocalStickerVideoFx:resultDestPath stickerModelPath:modelDestPath stickerType:subType stickerIdentify:@"10273" upportGesture:NO];
                }
            }
        } else {
            [[BDCloudAVStreamContext sharedInstance] disabledStickerVideoFx];
        }
    }
    });
}
```

- 注意：当前仅支持贴纸内置使用，用户在申请授权成功后，会得到贴纸资源包，具体使用可以参考短视频SDK DemoD拍摄模块

下载贴纸

【v3.0】版本后，/**贴纸采用后下载方式使用**/，开通license的同时可以在Console购买或选择贴纸信息

下载贴纸涉及两个接口，贴纸列表信息和贴纸下载使用。

- 贴纸列表信息，即用于产品展示，建议开发者提前获取贴纸列表，可缓存，减少用户等待。

下方获取贴纸列表数据实例代码：

```
- (NSMutableArray<BDMVComposeStickerItem *> *)stickerItems {
    if (!_stickerItems) {
        dispatch_semaphore_t semaphore = dispatch_semaphore_create(0);
        _stickerItems =  [[NSMutableArray<BDMVComposeStickerItem *> alloc] init];

        /**在线获取贴纸信息**/
        //建议提前获取贴纸列表&&缓存本地
        [[BDCloudAVStreamContext sharedInstance] downloadStickerList:^(NSArray * _Nonnull stickerList,
                                                                       NSError * _Nonnull error)
        {
            if (stickerList) {
                for (NSDictionary *dict in stickerList) {
                    BDMVComposeStickerItem *item = [[BDMVComposeStickerItem alloc] initWitStickerDict:dict type:YES];
                    [self->_stickerItems addObject:item];
                }
            }
            dispatch_semaphore_signal(semaphore);
        }];
        dispatch_semaphore_wait(semaphore, DISPATCH_TIME_FOREVER);
        return _stickerItems;
    }
    return _stickerItems;
}
```

贴纸下载使用，即用户选择某个贴纸后，下载贴纸并且加载使用。注：部分贴纸还会涉及模型下载，见下方示例代码

```
- (void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath *)indexPath {

    BDMVComposeStickerItem *item = self.stickers[indexPath.row];

    if (item.state == BDMVComposeStickerItemState_Downloading || indexPath == self.selectedIndexPath) {
        return;
    }
    // 移除当前贴纸
    if (self.selectedIndexPath && self.selectedIndexPath.row < self.stickers.count) {
        BDMVComposeStickerItem *item = self.stickers[self.selectedIndexPath.row];
        item.state = BDMVComposeStickerItemState_Inexistence;
        [collectionView reloadItemsAtIndexPaths:@[self.selectedIndexPath]];
    }
    //选择新贴纸
    if (indexPath.row < self.stickers.count) {
        if (![self isExistWithStickerItem:item]) {
            //本地无缓存当前贴纸
            item.state = BDMVComposeStickerItemState_Downloading;
            [collectionView reloadItemsAtIndexPaths:@[indexPath]];
            // 首先创建缓存文件夹
            // 再下载贴纸列表
            if (![self isExistWithStickerDir]) {
                NSLog(@"缓存贴纸文件夹创建失败~~");
            }
            [[BDCloudAVStreamContext sharedInstance] downloadStickerVideoFx:item.identifier
```

```
                                        downloadCallBack:^(NSDictionary * _Nonnull stickerDic,
                                                           NSError * _Nonnull error) {
            if (stickerDic) {
                BDMVComposeStickerItem *remoteItem = [[BDMVComposeStickerItem alloc] initWitStickerDict:stickerDic type:YES];
                [self downloadStickerZip:remoteItem.decryptFile md5:remoteItem.fileMd5 completionHandler:^(NSString *md5){
                    // fix 用户快速切换贴纸
                    if ([md5 containsString:item.fileMd5]) {
                        dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
                        dispatch_async(queue, ^{
                            // 部分贴纸还依赖模型文件
                            if (remoteItem.modelRemotePath) {
                                [self asynDownloadDecalsWithURL:[NSURL URLWithString:remoteItem.modelRemotePath] destination:remoteItem.model_sk completion:^(NSURLResponse *response, NSURL *filePath,
NSError *error) {
                                    if (!error) {
                                        [self setLocalSticker:item.fileMd5 andSubtype:item.sub_type withModel:remoteItem.model_sk gestureSupport:item.gestureSupport];
                                    }else {
                                        NSLog(@"AR模型下载失败");
                                    }
                                }];
                            }else {
                                [self setLocalSticker:item.fileMd5 andSubtype:item.sub_type withModel:remoteItem.model_sk gestureSupport:item.gestureSupport];
                            }

                            dispatch_async(dispatch_get_main_queue(), ^{

                                if (self.selectedIndexPath) {
                                    BDMVComposeStickerItem *selectedItem = self.stickers[self.selectedIndexPath.row];
                                    selectedItem.state = BDMVComposeStickerItemState_Inexistence;
                                    [collectionView reloadItemsAtIndexPaths:@[self.selectedIndexPath]];
                                }
                                item.state = BDMVComposeStickerItemState_OK;
                                self.selectedIndexPath = indexPath;
                                [collectionView reloadItemsAtIndexPaths:@[indexPath]];

                            });
                        });
                    } else {
                        remoteItem.state = BDMVComposeStickerItemState_Inexistence;
                        [collectionView reloadItemsAtIndexPaths:@[indexPath]];
                    }
                }];
            }
        }];
    } else {
        //本地有缓存
        NSLog(@"贴纸已缓存");
        [self setLocalSticker:item.fileMd5 andSubtype:item.sub_type withModel:item.model_sk gestureSupport:item.gestureSupport];
        dispatch_async(dispatch_get_main_queue(), ^{
            if (self.selectedIndexPath) {
                BDMVComposeStickerItem *selectedItem = self.stickers[self.selectedIndexPath.row];
                selectedItem.state = BDMVComposeStickerItemState_Inexistence;
```

```
                [collectionView reloadItemsAtIndexPaths:@[self.selectedIndexPath]];
            }
            item.state = BDMVComposeStickerItemState_OK;
            self.selectedIndexPath = indexPath;
            [collectionView reloadItemsAtIndexPaths:@[indexPath]];
        });
    }
  }
}
```

Android开发说明

Android版本开发接入文档

一、概述

百度云短视频创作专注移动端视音频场景研发，提供端到端的一站式视音频技术解决方案，不限于采集、录制、合成、上传、存储、分发，极大降低客户接入音视频产品的技术门槛。

1.1 注意事项

运行环境

- Android 4.1系统，API Level 16以上

二、快速接入

请先通过工单申请获取邀测资格，通过审批后会有专人联系您提供SDK源码包。

解压SDK压缩包后将aar包放入项目libs目录：

- 在项目build.gradle添加库依赖：

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


dependencies {

   api(name: 'multimedia-processor-release', ext: 'aar')
   api(name: 'aiphoto-v1.1', ext: 'aar')


   api 'com.baidu.minivideo:transcoder-sdk:1.0.2.156'
   api 'com.baidu.minivideo:core-armv5:1.0.2.156'
   api 'com.baidu.minivideo:core-armv7a:1.0.2.156'
   api 'com.google.code.gson:gson:2.8.5'
   api 'com.googlecode.mp4parser:isoparser:1.0.1'
   api 'com.googlecode.plist:dd-plist:1.16'
}
```

- 在使用拍摄器SDK，需要申请产品对应的授权文件，如无授权，产品无法正常使用。
- 申请成功后，会得到一个licenseID和对应授权文件下载地址，用户下载成功后，需要手动添加到项目工程中

注意：授权文件后缀名为.license。

三、接口使用说明

3.1 录制设置

- 录制的相关接口是在com.baidu.ugc.record.RecordManager类里，包括录制、美颜、贴纸和滤镜

3.1.1 初始化接口

设置预览画面

```
 void setGLSurfaceView(GLSurfaceView view)
```

初始化录像机

```
 void init(RecordManager.ICamera cameraManager, int cameraFrameRate, int videoBitrate, boolean isCameraFront, java.lang.String outputDir, java.lang.String arDataPath)
```

数据加载

```
 void loadData(RecordManager.OnDataLoadCallback listener)
```

注意：在实例化new RecordManager(this, appid)需要传人appid即用户申请的licenseID，若未申请授权，请参考快速接入

3.1.2 生命周期方法

一般的生命周期为：初始化 -> resume -> pause -> destroy

停止预览，一般在在Activity的onStop时调用

```
void    pause()
```

开始预览，一般在Activity的onStart时调用

```
void    resume()
```

对象销毁，销毁后需要重新创建对象

```
void    destroy()
```

3.1.3 相机设置

获取到摄像头的状态

```
boolean isFrontCamera()
```

相机的当前摄像头是否正在处于预览中

```
boolean isPreviewing()
```

设置前后摄像头

```
protected void  setCameraFace(boolean front)
```

相机尺寸，默认720P,如果摄像头不支持，那么会取相近比例的尺寸

```
void    setCameraSize(int w, int h)
```

相机切换前后摄像头

```
void    switchCamera()
```

3.1.4 录制与停止录制

获取当前录制视频的完整路径

```
String    getVideoAbsoluteFile()
```

是否为正在录制

```
boolean isRecording()
```

设定视频输出目录

```
boolean setVideoPath(java.lang.String videoFileDir)
```

开始录制

```
boolean startRecording()
```

停止录制

```
boolean stopRecording()
```

3.2 美颜设置

美颜设置接口，代码示例如下：

调整美颜程度

```
void setBeauty(float v) //参数0.0f不美颜
```

调整美白程度

```
void setBeautyWhite(float value)
```

调整磨皮程度

```
void setBeautyBlure(float value)
```

调整大眼程度

```
void    setEnlargeEye(float value)
```

调整瘦脸程度

```
void    setCheekThin(float value)
```

- 注意：使用高级美颜时，需要申请对应权限，免费版本不支持此功能

3.3 滤镜设置

设置滤镜

```
void setFilter(Filter filter)
```

滤镜参数

```
void setFilterLevel(float v)
```

3.4 贴纸设置

设置贴纸道具

```
void setStickerEffect(Sticker sticker, java.lang.String tabName)
```

下载贴纸

- 【v3.0】版本后，贴纸采用后下载方式使用，开通license的同时可以在Console购买或选择贴纸信息；
- 使用贴纸涉及两个接口，获取贴纸列表信息、贴纸文件下载使用。

一、获取贴纸列表信息，即用于产品贴纸展示，建议开发者提前获取贴纸列表，可缓存，减少用户等待。下方获取贴纸列表数据实例代码：

1：调用加载贴纸列表数据

```
 相关类：com.baidu.smartminivideo.capture.sticker.helper.LoadSticker


public void loadStickerDatas() {
    String appId = Config.LICENSE_APPID;
    String sdkVersion = Config.sdkVersion;
    DownStickerHelper loadStickerHelper = new DownStickerHelper(SmartminivideoApplication.getContext(),
            appId, sdkVersion);
    loadStickerHelper.loadStickerList(this);
}
```

2：在第一步调用之后，会回调两个方法，将贴纸数据字符串传输回来

```
 成功回调：
@Override
public void onListSuccess(String stickerList) {}


失败回调：
@Override
public void onFail(String msg) {}
```

二、贴纸文件下载使用。即用户选择某个贴纸后，下载贴纸并且加载使用。
注：部分贴纸还会涉及模型下载，见下方示例代码:

1：调用加载单个贴纸数据

```
 相关类：com.baidu.smartminivideo.capture.sticker.helper.LoadSticker

public void loadOneSticker(String faceId) {
    String appId = Config.LICENSE_APPID;
    String sdkVersion = Config.sdkVersion;
    DownStickerHelper loadStickerHelper = new DownStickerHelper(SmartminivideoApplication.getContext(),
            appId, sdkVersion);
    loadStickerHelper.loadOneStickerList(Long.parseLong(faceId), this);
}
```

2：在第一步调用之后，会回调两个方法，将单个贴纸数据字符串传输回来

```
 成功回调：
@Override
public void onOneSuccess(String sticker) {}

失败回调：
@Override
public void onFail(String msg) {}
```

3：在第3步回调之后，若成功获取单个贴纸数据，则将继续下载贴纸文件，见下方实例代码:

```
 具体可查看demo工程中相关类：
com.baidu.smartminivideo.capture.sticker.StickerAdapter
com.baidu.smartminivideo.capture.beautify.StickerDownloadManager


StickerDownloadManager.download(item, new StickerDownloadManager.OnStickerDownloadCallback<FuFaceItem>() {
        @Override
        public void onStarted(FuFaceItem tag) {
            holder.startLoadingAnim();
        }

        @Override
        public void onProgress(FuFaceItem tag, long finished, long total, int progress) {

        }

        @Override
        public void onCompleted(FuFaceItem faceItem) {
            if (mLastClickIndex != position) {
                notifyDataSetChanged();
                return;
            }
            mSelectedIndex = position;
            notifyDataSetChanged();
            if (mListener != null) {
                mListener.onSelectSticker(item, faceItem.getFilePath());
            }
        }

        @Override
        public void onFailed(FuFaceItem faceItem, int what, int ext, String msg) {
            Log.e("StickerAdapter", "onFailed:" + faceItem.name
                    + ", what:" + what + ", ext:" + ext + ", msg:" + msg);
            notifyDataSetChanged();
        }
    });
}
```

概述

百度云短视频产品(SDK)专注移动端视音频场景研发，提供端到端的一站式视音频技术解决方案，不限于采集、录制、合成、上传、存储、分发，极大降低客户接入音视频产品的技术门槛。

注意事项

运行环境

- Android 4.1系统，API Level 16以上

快速接入

请提交工单进行邀测申请，申请通过后，我们会有专人与您对接并将拍摄器SDK发送给您。

解压后将aar包放入项目libs目录：

- 在项目build.gradle添加库依赖：

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


dependencies {

    api(name: 'multimedia-processor-release', ext: 'aar')
    api(name: 'aiphoto-v1.1', ext: 'aar')


    api 'com.baidu.minivideo:transcoder-sdk:1.0.2.156'
    api 'com.baidu.minivideo:core-armv5:1.0.2.156'
    api 'com.baidu.minivideo:core-armv7a:1.0.2.156'
    api 'com.google.code.gson:gson:2.8.5'
    api 'com.googlecode.mp4parser:isoparser:1.0.1'
    api 'com.googlecode.plist:dd-plist:1.16'
}
```

- 在使用拍摄器SDK，需要申请产品对应的授权文件，如无授权，产品无法正常使用。
- 申请成功后，会得到一个licenseID和对应授权文件下载地址，用户下载成功后，需要手动添加到项目工程中

注意：授权文件后缀名为.license。

合成

- 合成模块负责将编辑后的视频数据合成为.mp4视频文件

导出预览视频

- 导出预览视频，将媒体轨道中心的所管理的视频、音频、字幕导出本地，默认是.mp4文件。导出时，需要暂停预览
- 代码示例如下：

```
 VideoMuxer mVideomuxer = new VideoMuxer();
mVideomuxer.setListener(new com.baidu.ugc.api.VideoMuxer.OnMuxerListener() {
    @Override
    public void onMuxerMusicEnd() {
    }

    @Override
    public void onMuxerProgress(int i) {

    }

    @Override
    public void onMuxerEnd(String s) {
        // s即是合成后的视频.mp4文件产出
    }

    @Override
    public void onMuxerAbord() {

    }

    @Override
    public void onMuxerFail(String s) {

    }

    @Override
    public void onMuxerStart() {

    }
});

mVideomuxer.startMuxer(muxerData);
```

转场

视频专场

转场在业务逻辑层实现，请参考Demo代码。

字幕

创建字幕控制器

- 创建字幕控制器，使用字幕相关功能，需要创建字幕轨并添加到媒体轨道中心，并创建一个字幕片段，配合字幕UI逻辑使用，详见智能小视频源码,涉及组件BDHKVlogSubtitlesView(字幕位置)，BDMVSubtitleInputAccessoryView(字幕输入框)，BDMVInputEventBottomBar(字幕确认框)
- 代码示例如下：

```
//创建字幕控制器
    mUgcSubtitleEditController = new UgcSubtitleEditController(this, mUgcPreViewFrameLayout, mVideoView);
mUgcSubtitleEditController.setUgcVideoPreviewActivity(this);
mUgcSubtitleEditController.setSubtitleEditControllerListener(
        new UgcSubtitleEditController.SubtitleEditControllerListener() {
            @Override
            public List<MultiMediaData> getDataList() {
                return mDataSourceList;
            }

            @Override
            public void updateSubtitleList(List<SubTitleUnit> subTitleUnits) {
                if (mVlogEditManager != null) {
                    mVlogEditManager.setSubtitle(subTitleUnits);
                }
                if (mAEffectProcessor != null) {
                    mAEffectProcessor.changeEffect(mVlogEditManager.getShaderConfigMap(),
                            mVlogEditManager.getUpdateMediaTracks());
                }
                if (mUgcSubtitleEditController != null) {
                    mUgcSubtitleEditController.changeSubtitleIcon(!ListUtils.isEmpty(subTitleUnits));
                }
            }

            @Override
            public String getThemeId() {
                return "";
            }

            @Override
            public void updateSubtitleConfig(SubTitleConfig subTitleConfig) {
                if (mVlogEditManager != null) {
                    mVlogEditManager.setSubtitleConfig(subTitleConfig);
                }

                VideoDraftBean videoDraftBeanDb = getVideoDraftBeanDb();
                if (videoDraftBeanDb != null) {
                    videoDraftBeanDb.setVideoSubtitleConfig(SubTitleConfig.beanToJson(subTitleConfig));
                    CurrentVideoBeanManager.updateVideoDraftBean(videoDraftBeanDb);
                }
            }

            @Override
            public void setIsNeedPauseWhenEdit(boolean isNeedPauseWhenEdit) {
                mIsNeedPauseWhenEdit = isNeedPauseWhenEdit;
            }

            @Override
            public void saveSubtitleDraft(List<SubTitleUnit> subTitleUnits) {
                VideoDraftBean videoDraftBeanDb = getVideoDraftBeanDb();
                if (videoDraftBeanDb != null) {
                    videoDraftBeanDb.setVideoSubtitleData(SubTitleUnit.arrayToJson(subTitleUnits));
                    CurrentVideoBeanManager.updateVideoDraftBean(videoDraftBeanDb);
                }
            }

            @Override
            public void updateIsPauseByUser(boolean isPauseByUser) {
                mIsPausePlayByUser = isPauseByUser;
                if (mIsPausePlayByUser) {
                    if (mVlogEditManager != null) {
                        mVlogEditManager.pause();
                    }
                    if (mPreviewMusicPlayer != null) {
                        mPreviewMusicPlayer.onPause();
                    }
                }
            }
        });
```

设置字幕画面位置

- 设置字幕画面位置，可以调整字幕轨道上某个字幕的显示位置
- 代码示例如下：

```
private void inflateDragSubtitleLayout() {
    if (mDragSubtitleViewStub == null) {
        return;
    }
    mDragSubtitleLayout = (DragSubtitleLayout) mDragSubtitleViewStub.inflate();
    mDragSubtitleLayout.setOnDragItemClickListener(this);
    mDragSubtitleViewStub = null;
    if (mSubTitleConfig != null) {
        mDragSubtitleLayout.setSubtitleConfig(mSubTitleConfig);
    }
    mDragSubtitleLayout.setOnDragItemMovedListener(new DragSubtitleLayout.OnDragItemMovedListener() {
        @Override
        public void onDragItemMoved() {
            mIsChangeSubtile = true;
        }
    });
    mDragSubtitleLayout.setOnCancelFocusManuallyListener(new DragSubtitleLayout.OnCancelFocusManuallyListener() {
        @Override
        public void onCancelFocusManually(SubTitleUnit subTitleUnit) {
            if (mSubtitleEditLayout != null && mSubtitleEditLayout.getVideoShaft() != null
                    && mIPlayerDataSource != null) {
                mSubtitleEditLayout.getVideoShaft().clearSelectState();
            }
        }
    });
```

录制

- 录制的相关接口是在com.baidu.ugc.record.RecordManager类里，包括录制、美颜、贴纸和滤镜

初始化接口

设置预览画面

```
void   setGLSurfaceView(GLSurfaceView view)
```

初始化录像机

```
void   init(RecordManager.ICamera cameraManager, int cameraFrameRate, int videoBitrate, boolean isCameraFront, java.lang.String outputDir, java.lang.String arDataPath)
```

数据加载

```
void   loadData(RecordManager.OnDataLoadCallback listener)
```

注意：在实例化new RecordManager(this, appid)需要传人自己的appid（即用户申请的licenseID），否则无法通过鉴权；

可通过修改配置文件com.baidu.smartminivideo.capture.shoot.config.Config.java中的LICENSE_APPID进行appid的修改； 若未申请授权，请参考快速接入

生命周期方法

初始化 -> resume -> pause -> destroy

停止预览：一般在在Activit的onStop时调用

```
void    pause()
```

开始预览：一般在Activity的onStart时调用

```
void    resume()
```

对象销毁，销毁后需要重新创建对象

```
void    destroy()
```

相机设置

获取到摄像头的状态

```
boolean isFrontCamera()
```

相机的当前摄像头是否正在处于预览中

```
boolean isPreviewing()
```

设置前后摄像头

```
protected void  setCameraFace(boolean front)
```

相机尺寸，默认720P,如果摄像头不支持，那么会取相近比例的尺寸

```
void    setCameraSize(int w, int h)
```

相机切换前后摄像头

```
void    switchCamera()
```

录制与停止录制

录制

获取当前录制视频的完整路径

```
String    getVideoAbsoluteFile()
```

是否为正在录制

```
boolean isRecording()
```

设定视频输出目录

```
boolean setVideoPath(java.lang.String videoFileDir)
```

开始录制

```
boolean startRecording()
```

停止录制

```
boolean stopRecording()
```

美颜设置

- 美颜设置接口，代码示例如下：

```
 //调整美颜程度
void setBeauty(float v) //参数0.0f不美颜
//调整美白程度
void setBeautyWhite(float value)

//调整磨皮程度
void setBeautyBlure(float value)

//调整大眼程度
void    setEnlargeEye(float value)

//调整瘦脸程度
void    setCheekThin(float value)
```

- 注意：使用高级美颜时，需要申请对应权限，免费版本不支持此功能

滤镜设置

设置滤镜

```
void setFilter(Filter filter)
```

滤镜参数

```
void setFilterLevel(float v)
```

贴纸设置

设置贴纸道具

```
void setStickerEffect(Sticker sticker, java.lang.String tabName)
```

下载贴纸

- 【v3.0】版本后，**贴纸采用后下载方式使用，开通license的同时可以在Console购买或选择贴纸信息；**
- Console地址跳转：跳转Console
- 使用贴纸涉及两个接口，**获取贴纸列表信息、贴纸文件下载使用**。

一、获取贴纸列表信息，即用于产品贴纸展示，建议开发者提前获取贴纸列表，可缓存，减少用户等待。下方获取贴纸列表数据实例代码：

1：调用加载贴纸列表数据

```
 相关类：com.baidu.smartminivideo.capture.sticker.helper.LoadSticker


public void loadStickerDatas() {
    String appId = Config.LICENSE_APPID;
    String sdkVersion = Config.sdkVersion;
    DownStickerHelper loadStickerHelper = new DownStickerHelper(SmartminivideoApplication.getContext(),
            appId, sdkVersion);
    loadStickerHelper.loadStickerList(this);
}
```

2：在第一步调用之后，会回调两个方法，将贴纸数据字符串传输回来

```
 成功回调：
@Override
public void onListSuccess(String stickerList) {}


失败回调：
@Override
public void onFail(String msg) {}
```

二、贴纸文件下载使用。即用户选择某个贴纸后，下载贴纸并且加载使用。注：部分贴纸还会涉及模型下载，见下方实例代码:

1：调用加载单个贴纸数据

```
 相关类：com.baidu.smartminivideo.capture.sticker.helper.LoadSticker

public void loadOneSticker(String faceId) {
    String appId = Config.LICENSE_APPID;
    String sdkVersion = Config.sdkVersion;
    DownStickerHelper loadStickerHelper = new DownStickerHelper(SmartminivideoApplication.getContext(),
            appId, sdkVersion);
    loadStickerHelper.loadOneStickerList(Long.parseLong(faceId), this);
}
```

2：在第一步调用之后，会回调两个方法，将单个贴纸数据字符串传输回来

```
 成功回调：
@Override
public void onOneSuccess(String sticker) {}

失败回调：
@Override
public void onFail(String msg) {}
```

3：在第3步回调之后，若成功获取单个贴纸数据，则将继续下载贴纸文件，见下方实例代码:

```
 具体可查看demo工程中相关类：
com.baidu.smartminivideo.capture.sticker.StickerAdapter
com.baidu.smartminivideo.capture.beautify.StickerDownloadManager


StickerDownloadManager.download(item, new StickerDownloadManager.OnStickerDownloadCallback<FuFaceItem>() {
        @Override
        public void onStarted(FuFaceItem tag) {
            holder.startLoadingAnim();
        }

        @Override
        public void onProgress(FuFaceItem tag, long finished, long total, int progress) {

        }

        @Override
        public void onCompleted(FuFaceItem faceItem) {
            if (mLastClickIndex != position) {
                notifyDataSetChanged();
                return;
            }
            mSelectedIndex = position;
            notifyDataSetChanged();
            if (mListener != null) {
                mListener.onSelectSticker(item, faceItem.getFilePath());
            }
        }

        @Override
        public void onFailed(FuFaceItem faceItem, int what, int ext, String msg) {
            Log.e("StickerAdapter", "onFailed:" + faceItem.name
                    + ", what:" + what + ", ext:" + ext + ", msg:" + msg);
            notifyDataSetChanged();
        }
    });
}
```

视频指导

视频指南

注册账号及实名认证请您参考注册账号和实名认证。短视频SDK目前对个人认证还是企业认证不做要求，推荐您进行企业认证。

SDK体验

能力限制

支持的系统与设备

项目	说明
手机设备	- 华为、小米、OPPO、VIVO等主流机型 - iphone6及以上
操作系统	- ios 9.0以上系统 - Android 4.4以上系统

特效能力

美颜建议值			
效果	默认值	最大值	建议值
磨皮	40%	100%	0.7
美白	50%	100%	0.14
大眼	35%	100%	0.72
瘦脸	60%	100%	0.7

SDK下载

短视频SDK

- 开发者：北京百度网讯科技有限公司
- 版本号：见SDK下载列表
- 主要功能：提供视频拍摄、视频录制及剪辑、AR特效等功能；
- 个人信息处理规则： 见 百度短视频SDK隐私政策
- 合规使用说明：见 百度短视频SDK开发者个人信息保护合规指引

短视频SDK中集成了AR特效的能力，您可以选择下载您申请的license中的AR版本对应的SDK版本

Android短视频SDK下载列表

SDK版本	集成AR版本	更新说明	更新时间	SDK下载
版本号：V6.1.0 包名：com.baidu.cloudvideo.edit MD5：e522792b42066845355f828b78fb12b9	V5.1	1.SDK支持arm64位系统 2.优化SDK架构	2022-9-9	下载

Demo体验

1 Demo下载

我们提供完整的demo APP，包含AR特效能力和短视频拍摄&编辑全能力；
您可扫描以下二维码下载安装体验。

- **Android**

- **IOS**

下载demo需要密码，密码为：bdcloud

2 特效demo体验

（1）互动特效case-人脸特效

- 多人脸贴纸特效

（2）互动特效case-手势互动（7款手势）

- 手势控雨特效

（3）互动特效case-肢体互动

- 肢体互动特效-雷电

（4）互动特效case-身体美化

- 头发染色特效

（5）互动特效case-环境特效

- 手势切换粒子+天空顶特效

（6）互动特效case-SLAM放置

- 在贴纸中支持，放置人物，动物，产品等3D模型

常见问题

常见问题总览

开发类问题

- 开发环境要求
- 提示license过期
- 美颜、美妆、贴纸等无效果
- 需要录音权限
- 本地添加贴纸

内容制作类

- 怎样制作特效贴纸？
- 有音乐资源吗？

常见错误码

以下为集成SDK中可能遇到的常见错误码

内容制作类问题

怎样制作特效贴纸？

可提供适用于SDK的特效贴纸制作工具（支持人脸、手势、人像分割等）。工具面向设计师，采用可视化的界面、操作成本比较低，设计师即可独立完成内容的开发制作，极大的节省了技术开发资源。
如无设计团队，可采购我们的贴纸素材库，贴纸库里已有上百款内容。

有音乐资源吗？

接入了太合版权库，有20万首版权音乐可供选择

常见错误码

常见错误码

1、错误码BDCloudAVAuthFailure = 20000，可能是因为licenseID与.license不对应 ,请去官网比对
2、错误码BDCloudAVAuthLicenseMiss = 20004，项目中没有找到.license文件
3、错误码BDCloudAVAuthLicenseExpired = 20006，可能是licenseId配置成了包名，请区分licenseId与包名不同
4、错误码BDCloudAVAuthLicenseStickerNoPermission = 20007，没有获取对应贴纸的权限，请重新编辑服务请求，选取对应的贴纸
5、错误码BDCloudAVAuthControlFailure = 20001，
BDCloudAVAuthSessionTokenFailure = 20002，
BDCloudAVAuthResponseNone = 20003，
BDCloudAVAuthLicenseSignFailure = 20005，
BDCloudAVAuthParamError = 20008，
可能与请求参数配置有关，可以联系我们

开发类问题

开发环境要求

1、Android Studio 3.2或以上版本，Gradle 4.6或以上版本。编译环境请选择支持java8， Android 4.4系统以上，API Level 19以上；
2、iOS9及以上系统，Xcode 9.0+

提示license过期

1、app的build.gradle中的ApplicationId需要和申请license时的包名一致
2、和包名匹配的license文件以及和包名匹配的 licenseID
注意：三者不能填错（须自查）；开发前请阅读demo中ReadMe文件

美颜、美妆、贴纸等无效果

请查看license是否设置正确，具体请参考工程搭建文档。
如果设置正确，请查看设置美颜、美妆、贴纸的时机，是否在onSetup方法回调后，具体请参考上方美颜、美妆、贴纸设置文档。

添加本地贴纸

若需要使用自定义本地贴纸，需要把资源放到工程项目Sources/LocalStickers文件夹中，同一个贴纸资源命名一致（包括贴纸图片，贴纸资源，贴纸模型资源）。

需要录音权限

请在授予相机权限，麦克风权限，存储空间权限后再录制。

相关协议

短视频SDK开发者个人信息保护合规指引

亲爱的开发者：

感谢您在所开发的移动应用中集成并使用百度旗下软件开发工具包（SDK）。

百度非常重视用户个人信息保护，包括集成百度旗下短视频软件开发工具包（SDK）（以下简称“百度SDK”）的移动应用的最终用户（以下简称“最终用户”）个人信息保护，特制定《短视频SDK个人信息保护合规开发者指引》，以供您在所开发的移动应用（以下简称“移动应用”）中集成并使用百度SDK时参照执行。

一、 个人信息保护合规基本要求

在所开发的移动应用中集成并使用百度SDK，您需要首先遵守以下个人信息保护合规基本要求：

1. 您应遵守收集、使用最终用户个人信息有关的所有可适用法律法规及规范性文件要求，包括但不限于《个人信息保护法》、《网络安全法》、《App违法违规收集使用个人信息行为认定方法》、《工业和信息化部关于开展APP侵害用户权益专项整治工作的通知》（工信部信管函［2019］337号）、《工业和信息化部关于开展纵深推进APP侵害用户权益专项整治行动的通知》（工信部信管函〔2020〕164号）、《工业和信息化部关于进一步提升移动互联网应用服务能力的通知》（工信部信管函〔2023〕26号）等，保护用户个人信息安全。

2. 您的APP需要制定一份独立的隐私政策。隐私政策的内容建议通俗易懂，对您的APP收集、使用个人信息的目的、方式、范围，清晰、完整、准确地向个人信息主体进行明示告知，并充分给予最终用户独立的选择权，确保在获得个人信息主体授权同意后方可进行个人信息的处理活动。《隐私政策》应由用户自主选择是否同意，不应以默认勾选同意的方式或是以欺骗诱导的方式取得用户授权。但请您注意，仅是改善服务质量、提升用户体验、定向推送信息、研发新产品还不足以能成为要求用户同意收集其个人信息的理由。

3. 您应将在移动应用中集成并使用百度SDK服务的情况，以及百度SDK对最终用户必要个人信息的收集、使用和保护规则（具体请见短视频SDK隐私政策），在移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（具体请参考本指引第二部分 “如何告知最终用户”及第三部分“告知文案示例”），并获得最终用户对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。如果最终用户是未满14周岁的未成年人，请您务必确保获得最终用户的父母或其他监护人对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。

4. 您应向最终用户提供易于操作的访问、更正、删除其个人信息，撤销或更改其授权同意、注销其个人账号等用户权利实现机制。

5. 您应确保在移动应用首次运行时，应在最终用户阅读并同意移动应用隐私政策之后，方可初始化百度SDK进行最终用户信息采集。

6. 百度SDK会根据产品升级优化、提升安全性能、法律及监管要求等原因，不断升级迭代SDK版本，不同版本的SDK获取的字段信息可能会有差异。为了保证合法合规开展合作，并切实履行保护用户个人信息的责任与义务，请您务必确保您已将APP中的百度SDK升级至官方最新版本，以避免因使用旧版本SDK而出现违法违规的问题，导致您的APP遭受监管处罚的风险。百度SDK更新后会及时通过官网通知、站内信、公告、邮件、短信等有效方式对您进行告知，请您及时关注并尽快更新。

除了上述个人信息保护合规基本要求外，您还应遵守您所开发的移动应用所集成并使用的短视频SDK隐私政策。

SDK基本业务功能与拓展业务功能：

短视频SDK基本业务功能为提供视频拍摄、AR特效、剪辑、播放、音乐、双语字幕、水印、数据统计、质量监控。短视频SDK扩展业务功能为滤镜、美颜、美妆、五官级塑型、人脸、手势、肢体、环境等AR互动特效等。

以下是短视频SDK获取您的APP的最终用户的个人信息以及权限，由于不同SDK版本采集的字段与是否可选可能存在一定差异，具体采集情况以实际接入的SDK版本为准：

SDK收集个人信息情况：

功能及服务	个人信息类型	个人信息字段（区分必选信息和可选信息）
区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌	必选
区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号	必选
区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本	必选
区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息	必选
区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况	必选
区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息	必选
区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率	必选

安卓操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集视频数据，进行短视频录制	视频录制时
录音	开启麦克风	用于采集音频数据，进行短视频录制	视频录制时
相册	存储图片或视频	录制或截屏时，将图片或视频存储在相册中	短视频编辑时

IOS操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集音频数据，进行短视频录制	视频录制时
录音	开启麦克风	用于采集音频数据，进行短视频录制	视频录制时
相册	存储图片或视频	录制或截屏时，将图片或视频存储在相册中	短视频编辑时

SDK初始化设置：

请务必确保您已将短视频SDK升级到最新版。上述版本调用初始化函数不会采集用户个人信息，也不会向百度联盟后台上报数据。请确保用户同意《隐私政策》后再进行start/autoTrace采集配置，方可采集用户个人信息并上报数据。

调用初始化函数文档： https://cloud.baidu.com/doc/VideoCreatingSDK/s/Zkidxpnim#311-%E5%88%9D%E5%A7%8B%E5%8C%96%E6%8E%A5%E5%8F%A3

二、 如何告知最终用户

为帮助您明确地告知最终用户百度SDK个人信息收集、使用和保护相关事宜，我们为您提供了以下告知方式，供您参考执行：

1. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并显示相应**百度SDK隐私政策链接**以告知最终用户，百度SDK收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

2. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并参考相应百度SDK隐私政策内容，**以条款或表格等形式列明**收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

3. 当最终用户在移动应用中首次打开/使用相应功能/服务时，以**弹窗、页面提示**方式显示相应**百度SDK隐私政策链接**，以告知最终用户相应功能/服务由百度SDK提供，百度SDK为提供相应功能/服务而收集、使用的最终用户个人信息类型、目的及用途，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

三、 告知文案示例

为帮助您明确地告知最终用户百度SDK个人信息保护规则相关事宜，我们为您提供了以下告知方文案示例，供您参考执行：

文案示例A

为向您提供短视频编辑功能/服务，我们集成了北京百度网讯科技有限公司的短视频SDK。在为您提供短视频编辑功能/服务时，北京百度网讯科技有限公司短视频SDK需要收集、使用您必要的个人信息，包括设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息，用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用短视频拍摄、美颜特效及剪辑等功能。关于北京百度网讯科技有限公司短视频SDK收集、使用的个人信息类型、目的及用途，以及短视频SDK将如何保护所收集、使用的个人信息，请您仔细阅读《短视频SDK隐私政策》了解。

文案示例B

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	个人信息类型	使用目的	官网链接/隐私政策链接
短视频SDK	北京百度网讯科技有限公司	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用短视频拍摄、美颜特效及剪辑等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/4ki89s6x2

文案示例C

为向您提供短视频编辑功能/服务，我们集成了北京百度网讯科技有限公司的短视频SDK。在为您提供短视频编辑功能/服务时，北京百度网讯科技有限公司的短视频SDK收集、使用您的设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息，用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用短视频拍摄、美颜特效及剪辑等功能，具体请您仔细阅读《短视频SDK隐私政策》了解。

文案示例D

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	业务功能	个人信息类型	调用权限类型	具体目的/用途	隐私政策链接
短视频SDK	北京百度网讯科技有限公司	短视频剪辑	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	相机、录音、相册	用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用短视频拍摄、美颜特效及剪辑等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/Fmcx55nhn

四、 使用SDK服务的合规注意事项

1. 您接入百度SDK前，应当仔细阅读SDK的协议约定、本合规规范、用户协议、隐私政策等内容，并依据相关内容对您APP的《隐私政策》及您APP收集、存储、使用、共享等处理个人信息的情况进行合规自查。
2. 您知悉并认可百度SDK具备收集个人信息的功能，并认可该等信息的收集均为双方合作之必要目的所需。
3. 您承诺已制定并按照相关要求公示您APP的《隐私政策》，并已清晰、明确、显著地说明有关通过SDK收集个人信息的必要性、收集数据的范围、方式以及用途。同时，您应确保在APP首次运行时以弹窗等合规方式显著提示用户阅读您APP的《隐私政策》并取得用户的合法授权同意，经过合法授权后再初始化百度SDK进行个人信息的收集与处理。
4. 您已认真阅读并理解百度SDK平台协议、合作规范、隐私政策、接入文档等约定和要求，并承诺在您使用百度SDK服务期间，针对百度SDK收集、使用、处理、共享、转让相关个人信息，您已取得了用户持续有效的授权和同意，并保证您不会违反国家相关法律法规、相关国家标准以及双方约定的目的。
5. 如果您的APP面向不满十四周岁的儿童及未成年人用户提供服务，您承诺遵守儿童个人信息保护及未成年人保护相关的法律法规，您承诺已采取相关措施并保证已获得其监护人的授权同意。
6. 如因您违反百度SDK的平台协议、合作规范、隐私政策、接入文档等约定，导致您的用户或第三方对百度主张任何形式的索赔或权利要求，或导致百度因此产生任何法律纠纷的，您将负责解决并承担全部责任，如因此给百度及其关联主体造成损失的，您应赔偿因此给百度及其关联主体造成的全部损失。
7. 您保证对于您从百度SDK获取的数据，无论是在合作期间或是合作停止后，均承担保密义务，不擅自提供、泄露、透露给任何第三方，并应采取一切合法措施以使上述数据免于散发、传播、披露、复制、滥用及被无关人员接触，避免导致相关数据被超出双方合作目的使用。

短视频SDK隐私政策

欢迎使用短视频软件开发工具包（SDK）（简称"短视频SDK"）服务！

短视频SDK 是一款为移动应用开发者（以下简称"开发者"）提供移动端直播推流服务的软件开发工具包（SDK）。开发者在其移动应用内集成短视频SDK后，可通过短视频SDK平台向其移动应用（以下简称"开发者应用"）的最终用户（以下简称"最终用户"）提供本隐私政策所说明的功能及服务。开发者在其移动应用集成并使用短视频SDK服务时，委托短视频SDK处理开发者应用相关数据信息，其中可能包括开发者应用最终用户（以下简称"最终用户"）的个人信息。此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策"附录1：名词解释"来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用短视频SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用短视频SDK服务的情况，以及短视频SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于短视频SDK收集、使用最终用户相关个人信息的完整、合法、在使用短视频SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于短视频SDK收集、使用最终用户相关个人信息的完整、合法、在使用短视频SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

（4）关于上述承诺的具体落地执行可参考《短视频SDK开发者个人信息保护合规指引》。

2、我们希望集成并使用短视频SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的短视频SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，短视频SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用短视频SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订
8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用短视频SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌（必选）	采用加密传输的安全处理方式	iOS及Android
2	区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号（必选）	采用加密传输的安全处理方式	iOS及Android
3	区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本（必选）	采用加密传输的安全处理方式	iOS及Android
4	区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息（必选）	采用加密传输的安全处理方式	iOS及Android
5	区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况（必选）	采用加密传输的安全处理方式	iOS及Android
6	区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息（必选）	采用加密传输的安全处理方式	iOS及Android
7	区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率（必选）	采用加密传输的安全处理方式	iOS及Android

2. 设备权限调用

为了保证最终用户能正常使用短视频SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能，各项功能及功能对设备权限的调用情况如下：

Android系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
相册	录制、截屏功能将视频和图片存储到相册	同上

iOS系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
相册	录制、截屏功能将视频和图片存储到相册	同上

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用短视频SDK产品和服务时遇到的问题，改进及优化短视频SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用短视频SDK服务时遇到的问题，改进及优化短视频SDK的服务体验，提高短视频SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用短视频SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与短视频SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息：

仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；
2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；
3. 如果我们或我们的关联公司涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；
2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；
6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；
8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在短视频SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。
2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。
3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系，**并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。
4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过短视频SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加密，注意确保您个人信息的安全。**
5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**
6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案，**由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。
2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。
3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

短视频SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成短视频SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及**《百度儿童个人信息保护声明》**，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过工单联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用短视频SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；
2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；
3. 如果我们停止提供产品或者服务，或者保存期限已届满；
4. 如果最终用户撤回同意；
5. 法律、行政法规规定的其他情形。

当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

短视频SDK愿一直陪伴开发者和最终用户，若因特殊原因导致短视频SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过【八、如何联系我们】中的方式与我们取得联系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在短视频SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用短视频SDK产品和服务，若开发者继续集成并使用短视频SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用短视频SDK服务期间持续有效的授权同意。**若最终用户不同意该等变更，可以停止使用短视频SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用短视频SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；
2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；
3. 个人信息共享、转让或公开披露的主要对象发生变化；
4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；
5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；
6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

短视频SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对短视频SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过工单反馈开发者及最终用户对短视频SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台(http://help.baidu.com/personalinformation) 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。 敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行短视频SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等唯一设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问短视频SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的**精确位置信息**，也包括通过IP地址或其他网络信息等获取的**粗略地理位置信息**。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。

匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度手机助手及其他百度系产品https://www.baidu.com/more/)等网站、程序、服务、工具及客户端。

关联公司是指短视频SDK的经营者【北京百度网讯科技有限公司】及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对短视频SDK的信任和使用！

北京百度网讯科技有限公司

【2023】年【12】月【15】日更新

【2023】年【12】月【15】日生效

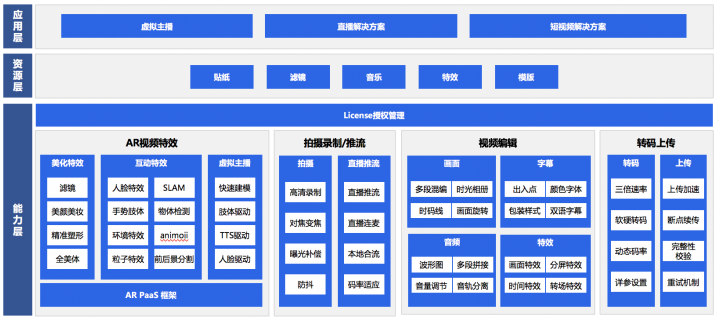
移动直播SDK

产品描述

产品简介

移动直播SDK是智能视频SDK的移动直播场景化产品，为用户提供移动端直播推流的快速集成方案。支持美颜滤镜和五官级精细调节，可通过人脸道具贴纸塑造更美的主播形象。 支持虚拟主播等AR特效玩法，为直播带来更多趣味互动体验。

产品功能全景图：



名词解释

- 推流

将本地视频源和音频源推送到百度直播云服务器

- 拉流

即直播播放，指已实现直播推流之后，用指定地址将百度云服务器中的视频源和音频源拉取播放的过程。其视频源是实时生成的，而实时直播在播视频的时候是没有进度条的。

- 推流域名

指用于推送直播流的域名，必选配置，该域名必须在使用直播服务前完成注册并备案。配置完推流域名后，直播服务会生成对应的推流地址。移动直播SDK可选择LSS音视频直播

- 播放域名

指用于播放直播流的域名，必选配置，该域名必须在使用直播服务前完成注册并备案。

- License

移动直播支持两个版本的SDK License，通过不同版本的License对应不同的移动直播SDK功能服务

- AR特效（AR Effect）

根据人脸、手势和肢体关键点定位，产生各种互动效果的贴纸。

- 视频解析度(Video Resolution)

视频的基本信息，包括图像宽高和像素横纵比等。

- 音频解析度(Audio Resolution)

音频的基本信息，包括采样率和声道数等。

功能说明

直播

模块	功能点	说明
界面	UI界面自定义	不限制界面布局，UI界面自由定制
直播推流	静音推流	可设置静音推流，仅推送画面流
	断流重连	断流后自动重连，默认无限次重连，用户可自定义重连策略
	RTMP推流	用于实现主播端的手机推流功能（秀场直播）
	SRT推流	用于实现低延时推流
	自定义metadata信息	可自定义流中的metadata信息，可用于直播答题、直播换装等
	横版/竖版	支持推流画面为横版竖版
	码率自适应	可设定最大最小码率，根据网络情况在设定期间内动态调整推流码率大小
	参数自定义	支持自定义推流参数，包括：码率、分辨率等
互动直播	视频连麦	支持视频连麦互动
	音频连麦	支持音频连麦互动
	旁路直播	支持服务端合流后，转直播推流
	画中画	支持视频连麦时的画中画交互，布局可自定义
	背景音乐混音	支持背景音乐播放，可调节音乐，支持循环播放
基础拍摄	拍摄控制	支持拍摄时的前后摄像头切换、闪光灯的关闭
	焦距调节	支持手势双指调节焦距（放大或缩小）
	对焦模式	支持手动对焦和自动对焦
	防抖	IOS支持拍摄时防抖
	清晰度	支持标清、高清、超清拍摄，支持自定义码率、帧率、gop
	镜像设置	支持本地镜像和远端镜像
拍摄美化	滤镜	调节画面：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等
	基础美颜	支持美白、磨皮、大眼、瘦脸
	美妆	支持腮红、高光、阴影、眉毛、睫毛、口红、眼线、眼影、美瞳
	五官精准塑形	支持眼睛、鼻子、下巴、脸型、额头等设置，详见“AR特效”描述
拍摄互动特效	人脸/手势/肢体/分割/环境互动特效	详见“AR特效”功能列表

AR特效

功能项		功能说明
造型美	基础滤镜	支持内置和自定义色卡滤镜，如：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等
	贴纸	屏幕贴纸，支持图片、序列帧动画等随屏素材，支持27种混合模式（如相加、平均、差值、变暗、反色、底片、线性减淡、正片叠底、滤色、柔光等）
	屏幕特效	支持分屏（上下/左右分屏、3/4/6/9屏等）、镜像、闪屏、灵魂出窍、抖动、水波纹、眩晕、斜进出等屏幕特效，也可通过shader自定义屏幕特效。
轻美妆	美白	美颜能力,可调节美白强度,满足个性需求
	磨皮	美颜能力,可调节磨皮强度,满足个性需求
	红润	美颜能力,可调节红润强度,满足个性需求
微整形	瘦脸	美颜能力，实时调节瘦脸参数
	大眼	美颜能力，实时调节大眼参数
高级美妆塑形	美妆特效	全局支持口红，腮红，修容，眼影（含睫毛），眼线，眉毛；单项支持美瞳、祛法令纹、祛黑眼圈
	高级美颜	五官精准塑形（眼睛、鼻子、下巴、脸型、额头、颧骨、发际线等）
人脸特效	人脸贴纸	基于人脸190+特征点精准定位和跟踪，添加2D/3D动态面具贴纸
	皮肤贴纸	基于人脸识别，实现皮肤贴纸，如：脸谱、豹子脸
	人脸变形	基于人脸识别，实现面部变形功能，如：嘟嘟脸、方脸
	表情识别	表情识别，挑眉、眨眼、张嘴、点头、嘟嘴（5个）触发特效，如：喷火帽、巫师帽
	多人脸	通过AI算法，检测画面中多个人脸区域
	换脸	实现双人换脸特效
	人脸实时驱动	基于人脸识别，支持实时驱动3D面具表情的玩法，如：柴犬面具
手势特效	手势识别触发	手势识别触发特效，支持比V、点赞、OK、单手比心、食指比1、握拳、手掌的检测（7个手势）
	手部贴纸	基于手势识别，添加2D/3D贴纸道具
	指尖检测	指尖点的精准检测和跟踪
	手势实时驱动	基于手部21关节点检测，实现手部驱动手偶的玩法
肢体特效	瘦身/增高	基于全局实现瘦身/增高/哈哈镜效果
	动作识别	识别肢体动作，并触发特效，包含2D/3D特效
前后景分割	人像分割	通过AI算法，识别视频中的人物区域（上半身），实现替换背景，背景虚化等功能
	头部分割-大头特效	通过AI算法，可精准识别头部区域，可实现大头特效玩法
	天空分割	通过AI算法，识别视频中的天空区域，实现天空背景更换，替换及素材叠加。
	头发分割	通过AI算法，识别视频中的头发区域，可实现染发、换发型等效果
环境特效	SLAM放置	通过即时定位与跟踪算法，在实景中放置3D模型动画
	空间粒子特效	模型粒子特效，能模拟现实中的雨、雪、落叶、飞花、流星等自然景观，以及魔法球、光环、火焰、冬去春来、花瓣雨、控雨等特效
	触屏手势粒子	触屏绘制烟花或气泡状的图像、写字，出粒子效果
	天空顶	星空、粉色天空等穹顶效果，360空间氛围渲染
	分屏	设置分屏滤镜，实现画面分割
	物体检测	通过AI算法，检测画面中的物体（当前支持杯子、其他物体需定制）
	透明视频/绿幕视频	支持通过手机姿态定位，在空间中放置透明/绿幕扣背景视频素材

核心优势

- 智能直播：智能的移动直播推流工具，丰富AI特效和更低延迟。
- 虚拟主播：依托百度技术优势，可生成定制虚拟形象，筑建更趣味直播场景。
- 简单快捷：标准易用的配置操作引导，快人一步。
- 更完善的直播解决方案：提供直播推流播放全链路服务，帮助快速搭建直播业务。
- 丰富的资源生态：营造贴纸生产者和购买者联系的生态平台，提供丰富的生态资源。
- 更高性价比：在体验优质的技术服务的同时，享受超低价格。

购买指南

产品定价

移动直播SDK套餐包内容如下表所示。套餐详细价格、贴纸资源、音乐资源请点击移动直播SDK价格详情查看。

模块	功能点	说明	直播基础版	直播专业版
界面	UI界面自定义	不限制界面布局，UI界面自由定制	✓	✓
直播推流	静音推流	可设置静音推流，仅推送画面流	✓	✓
	断流重连	断流后自动重连，默认无限次重连，用户可自定义重连策略		
	rtmp推流	用于实现主播端的手机推流功能（秀场直播）		
	srt推流	用于实现低延时推流		
	自定义metadata信息	可自定义流中的metadata信息，可用于直播答题、直播换装等		
	横版/竖版	支持推流画面为横版竖版		
	码率自适应	可设定最大最小码率，根据网络情况在设定期间内动态调整推流码率大小		
	参数自定义	支持自定义推流参数，包括：码率、分辨率等		
互动直播	视频连麦	支持视频连麦互动	✓	✓
	音频连麦	支持音频连麦互动		
	旁路直播	支持服务端合流后，转直播推流		
	画中画	支持视频连麦时的画中画交互，布局可自定义		
	背景音乐混音	支持背景音乐播放，可调节音乐，支持循环播放		
基础拍摄	拍摄控制	支持拍摄时的前后摄像头切换、闪光灯的关闭	✓	✓
	焦距调节	支持手势双指调节焦距（放大或缩小）		
	对焦模式	支持手动对焦和自动对焦		
	防抖	IOS支持拍摄时防抖		
	清晰度	支持标清、高清、超清拍摄，支持自定义码率、帧率、gop		
	镜像设置	支持本地镜像和远端镜像		
拍摄美化	滤镜	调节画面：回忆、少女、都市、红唇、霓虹、白茶、暗调、橘子汽水、美食、海礁、胡桃、日光、暮光之城、蔚蓝、夜景、白皙、微光、草莓、清凉等	✓	✓
	基础美颜	支持美白、磨皮、大眼、瘦脸	✓	✓
	美妆	支持腮红、高光、阴影、眉毛、睫毛、口红、眼线、眼影、美瞳	×	✓
	五官精准塑形	支持眼睛、鼻子、下巴、脸型、额头等设置，详见“AR特效”描述	×	✓
	特效人脸贴纸	支持人脸特效贴纸，包括2D、3D、动态、静态效果	×	✓
拍摄互动特效	手势/肢体/环境互动特效	详见的“AR特效”功能列表，全部可用于直播场景	×	×

如何购买

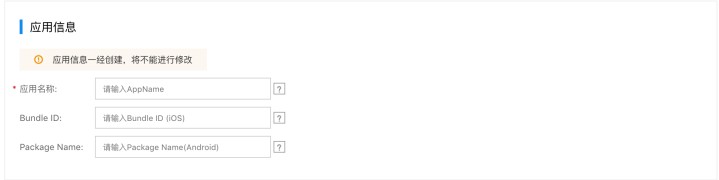
在线购买

1）license购买

您可在线自助购买，购买入口为“license购买”，如下图所示，选择购买项，支付成功即可。各购买项的详细描述详见“价格说明”。
其他订单查询和退款等事项同百度云标准流程，可在官网-财务中心查看。

2）license开通

购买成功后，即可申请license进行对接集成。开通入口为“license申请”，如下图所示，填写appname、bundleID、packagename，选择能力项、贴纸，点击确定。



我们收到申请后，会在2个工作日内进行审核，审核通过后，您可在console下载license文件和SDK工程包进行对接测试。

线下购买

如果线上购买项不能满足您的需求，请提交工单，会有客户经理根据您的需求为您提供全方位服务。

工单信息需包括：

- 公司名称
- 使用场景
- 需求描述：
- 联系电话：
- userID：在百度智能云官网注册账号，登录后，在用户中心->基本信息里即可查看到；
- appname
- packagename
- bundleID

欠费说明

到期提醒

- 试用版到期前5天提醒；
- 正式版到期前1个月、前7天、前1天提醒；
- 提醒方式：站内信、短信、邮件等；

停服说明

- 当license有效期到期后，我们会在到期当日停止服务，届时所有API和能力将无法使用；
- 在停服后，系统会以短信或邮件的方式通知您。

使用指南

接入教程

本文将帮助您理顺接入移动直播SDK所需要的步骤，基本操作流程如图所示：

注册及实名认证

使用百度智能云移动直播SDK前，用户需要拥有一个百度智能云账号并完成实名认证，具体操作如下：

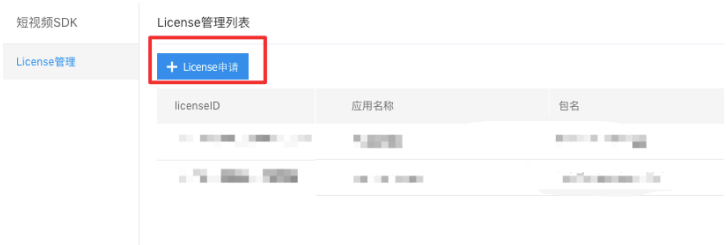
1.注册并登录百度智能云平台，请参考注册和登录。

2.完成实名认证，操作细节请参考实名认证。

3.登录成功后，在控制台导航栏中选择“产品服务>智能视频>智能视频SDK”，即可登录智能视频SDK的控制台。

license申请

- 进入智能视频SDK控制台，点击『license申请』，填写相关信息后提交。



- 百度智能云运营人员收到申请后会在2个工作日内审核完成，license审核通过后，"审核状态"会变为"审核通过"，您可以在console里下载license文件和SDK工程包。

SDK包里包含了完整的APP源码，以及提供移动直播能力的依赖包。您既可以使用全部的APP源码来从无到有的搭建起一个完整的直播推流APP，也可以仅将AR能力包导入已有的开发工程中，为您的APP增加特效能力。

集成直播能力

提供给您的SDK包里有已经调试好的移动直播demo完整源码，经过几步简单的配置即可调试运行，您可以借助该demo源码进行性能测试、功能调试和内部演示。

License申请

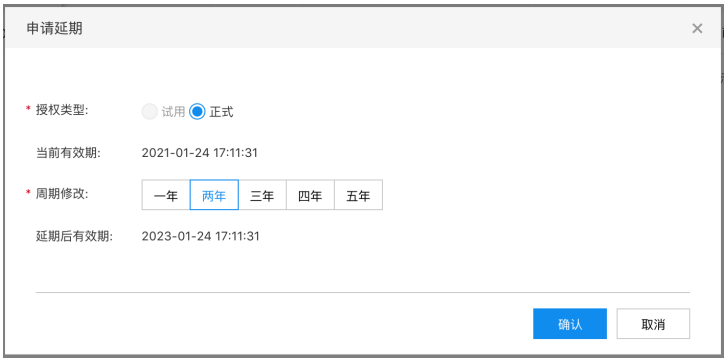
申请试用license

您可以免费申请移动直播SDK license进行对接测试，免费周期为2个月。申请步骤如下：

- 登录百度智能云账号，并完成实名认证（不完成实名认证，无法开通）。
- 进入短视频SDK官网产品页，点击【立即使用】，并开通移动直播SDK产品。

申请正式license

试用完成后，若您想由试用转成正式的继续使用，请联系您的客户经理或者提交工单申请。 在商务合同/付款等事宜确定后，您可在console里选择授权类型【正式】，弹框里选择商务确定的购买周期，点击确认，我们的运营人员审核通过后即可生效。



移动直播SDK申请

- 点击【license申请】，进入新建页面，输入应用名称、BundleID（IOS）或PackageName（Android），【套餐选择】可以选择移动直播套餐，还可单独【AR特效选择】勾选所需的AR能力，勾选所需的贴纸，授权信息勾选"试用"，点击【立即申请】。套餐所包含的功能项、贴纸数、价格等详见【价格说明】。

- 百度智能云运营人员收到申请后会在2个工作日内审核完成，审核通过后，您可以在console里下载license文件和SDK工程包（包含SDK，demo源码，接入说明文档）进行集成测试。

AR特效SDK申请

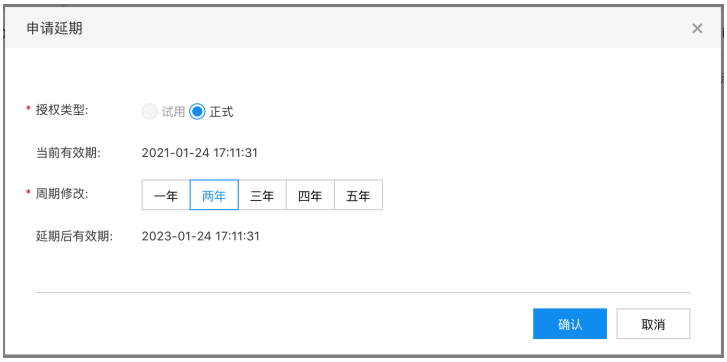
- 点击【license申请】，进入新建页面，输入应用名称、BundleID（IOS）或PackageName（Android），不用选择套餐，【AR特效选择】勾选所需要的AR能力，勾选所需的贴纸，授权信息勾选"试用"，点击【立即申请】。AR特效对应的功能项、价格等详见【价格说明】。

- 百度智能云运营人员收到申请后会在2个工作日内审核完成，审核通过后，您可以在console里下载license文件和SDK工程包（包含SDK，demo源码，接入说明文档）进行集成测试。

License续费

License续费

若您购买的license即将到期，为了不影响线上正常使用，请联系您的客户经理或者提交工单申请。 在商务合同/付款等事宜确定后，您可在console里点击【延期】，弹框里选择商务确定的购买周期，点击确认，我们的运营人员审核通过后即可生效。

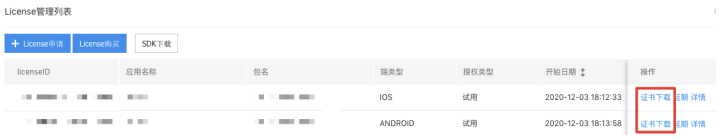


开发指南

IOS开发说明

IOS接入概述

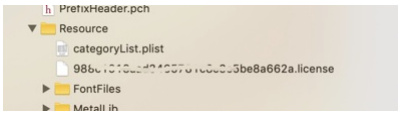
1.前置条件 拍摄器SDK使用的C++开发，对于引用SDK头文件的类文件，需要将.m文件改成 .mm文件，以兼容C++语言。
下载使用SDK，必须申请相对应的服务权限，获取license,SDK下载地址：https://console.bce.baidu.com/bvc/?_=1589879055994#/bvc/license/list



2.开发环境 拍摄器SDK适配iOS9及以上系统
Xcode 9.0+

3.申请license步骤 第一步：请到百度智能云平台申请。
第二步：填写应用信息、选择服务、素材选择、授权信息、立即申请，申请相对应的服务（注意：包名与项目BundleId必须一致）。
第三步：申请成功后，会得到一个licenseID和对应授权文件下载地址。下载成功后，.license文件需要手动添加到项目工程中。

4.配置license步骤 第一步：将下载好的.license文件添加到工程中，确保在Build Phases -> Copy Resource Bundle可以看到，可以参考Demo中的Resource文件夹中的结构目录，如图所示：

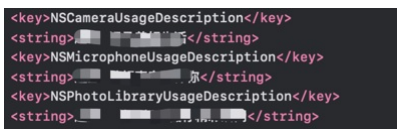


第二步：在AppDelegate.mm文件中didFinishLaunchingWithoptions方法里调用SDK的认证方法，licenseID在百度管理控制台云后台获取，licenseID必须与.license文件相对应，地址：
https://console.bce.baidu.com/bvc/?_=1589879055994#/bvc/license/list

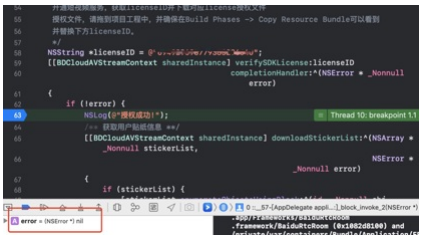
示例代码如图所示，具体代码可参考Demo中的AppDelegate.mm 文件。

注意：licenseID必须与.license文件一一对应，且项目BundleID与控制台包名一致。

SDK使用到相册，相机，声音权限，info.plist需要配置相对应的权限：



5.运行成功 依据上述步骤，申请对应服务license，下载SDK，导入框架资源到工程中，配置工程，在AppDelegate.mm中调用SDK的认证方法，运行工程后，认证方法返回的error为空即代表SDK接入运行成功。



拉流

音视频直播拉流相对与推流场景更加的常见。主要是使用支持实时在线流的播放器进行播放（支持HTTP-FLV;、HLS、SRT等协议的播放器）。

直播拉流文档 百度云同样提供播放器SDK提供直播拉流服务，详细请参考：https://cloud.baidu.com/doc/MCT/s/2jwvz54zo

直播推流

音视频直播的相关接口是在BDCloudAVLiveContext.h文件里，包括开始/结束推流、开始/结束动态码率、开/关静音推流、发送metadata信息等功能。
注意：直播SDK所有的类都是以"BDCloud"开头。

音视频推流直播需要注意以下两点：

1. 直播推流模块仅用来处理音视频数据（编码，打包，上传），不含音视频数据采集功能，我们需要依赖短视频模块进行数据采集后传递给直播模块。当然我们同样支持用户自己采集音视频数据按照我们要求的结构转入直播模块也能实现完整的直播功能。
2. 直播推流SDK当前支持两种网络协议（RTMP和SRT），不同协议的延迟不同。使用的场景需要根据自身业务情况自行评估。

1.1 推流参数设置 BDCloudAVLiveParameter用于配置推流地址、协议类型、断网内部重连次数、重连时间间隔、音视频编码参数等。
常见的音视频编码信息配置信息如下：

1. 视频分辨率：视频编码分辨率，视频编码场景中分辨率与清晰度正相关。
2. 视频帧率：视频编码帧率，视频编码场景中帧率与画面连贯正相关。
3. 视频码率：视频编码码率，视频编码场景中码率与清晰度正相关。
4. 视频gop长度：视频编码关键帧间隔，视频编码场景中关键帧间隔时间。
5. 视频编码显示类型：视频编码物理分辨率与显示分辨率不一致时的编码显示类型（等比、剪裁、拉伸）。
6. 音频采样率：音频编码采样率，音频编码场景中可以修改音频的采样率。
7. 音频码率：音频编码码率，音频编码场景中码率与清晰度成正相关。
   对于不熟悉音视频产品的用户，直播SDK提供默认的音视频编码配置，方便用户初始化。对于熟悉音视频直播推流的用户，可以自行修改参数以适配业务场景。

推流地址

主要用于RTMP协议下的推流地址。
类型：属性
默认值：nil
@property(nonatomic, copy) NSURL* url;
示例代码：

```
1. par.url = [NSURL URLWithString:[settings pushUrl]];
```

主要用于SRT协议下的推流地址。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString* c_url;
示例代码：

```
1. par.url = [settings pushUrl];
```

断网内部重连 次数设置
类型：属性
默认值：5
@property(nonatomic, assign) NSInteger reconnectCount;
示例代码：

```
1. par.reconnectCount = 5;
```

重连间隔
类型：属性
默认值：1
@property(nonatomic, assign) NSTimeInterval reconnectInterval;
示例代码：

```
1. par.reconnectInterval = 1;
```

设置协议类型 直播SDK支持两种协议类型（RTMP、SRT）,我们需要通过枚举区分设置。

类型：属性
默认值：BDCloudAVLiveTypeNormal.
@property(nonatomic, assign) BDCloudAVLiveType streamType;
示例代码：

```
1. par.streamType = NO
```

设置视频编码参数 视频编码参数支持分辨率、帧率、码率、关键帧间隔、编码类型（H264/H265）、显示类型、编码profile。
类型：属性
默认值：nil
@property(nonatomic, strong) BDCloudAVVideoOutputSettings* videoSettings;
示例代码：

```
par.videoSettings = [BDCloudAVVideoOutputSettings defaultSettings];
// 编码分辨率
par.videoSettings.dimension =  self.settings.dimension;
// 编码码率
par.videoSettings.bitRate = self.settings.bitRate * 1000;
// 编码帧率
par.videoSettings.frameRate = self.settings.frameRate;
// 关键帧间隔
par.videoSettings.maxKeyFrameInterval = 2;
// 编码类型
par.videoSettings.codecID = AVVideoCodecH264
// 显示类型
par.videoSettings.scalingMode = AVVideoScalingModeResizeAspect
// 编码profile
par.videoSettings.profile = AVVideoProfileLevelH264HighAutoLevel
```

设置音频编码参数 音频编码参数支持修改原有的音频参数，重新采样成目标采样率。同时支持修改编码码率。
类型：属性
默认值：nil
@property(nonatomic, strong) BDCloudAVAudioOutputSettings* audioSettings;
示例代码：

```
par.audioSettings = [BDCloudAVAudioOutputSettings defaultSettings];
// 设置码率
par.audioSettings.bitrate = 96000;
// 设置采样率
par.audioSettings.sampleRate = 44100;
```

1.2 metadata发送参数 用于配置直播流中发送metadata消息。可由百度云播放器解析并通过通知发送给每一个观众用户。常用于直播答题、直播换装等需要由主播添加进实时流内的消息功能。
设置鉴权参数AK 鉴权参数主要是百度云STS服务。由AK、SK、Token、expiredTime组成。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString *accessKey;
示例代码：

```
self.config.accessKey = signature[@"stsAccessKey"];
```

设置鉴权参数SK 鉴权参数主要是百度云STS服务。由AK、SK、Token、expiredTime组成。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString * secretKey;
示例代码：

```
self.config.secretKey = signature[@"stsSecretKey"];
```

设置鉴权参数Token 鉴权参数主要是百度云STS服务。由AK、SK、Token、expiredTime组成。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString * tokenKey;
示例代码：

```
1．self.config.tokenKey= signature[@"stsToken"];
```

设置鉴权参数expiredTime 鉴权参数主要是百度云STS服务。由AK、SK、Token、expiredTime组成。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString * expiredTimeInSeconds;
示例代码：

```
self.config.expiredTimeInSeconds= 1800;
```

设置流参数domin 流参数主要时拉流服务。由domin、appName、streamName组成。主要定位到flv拉流地址进行metadata信息下发。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString *domin;
示例代码：

```
1．_config.domin = domin;
```

设置流参数appName 流参数主要时拉流服务。由domin、appName、streamName组成。主要定位到flv拉流地址进行metadata信息下发。
类型：属性
默认值：nil
@property(nonatomic, copy) NSString *appName;
示例代码：

```
_config.appName = appName;
```

设置流参数streamName 流参数主要时拉流服务。由domin、appName、streamName组成。主要定位到flv拉流地址进行metadata信息下发。

类型：属性
默认值：nil
@property(nonatomic, copy) NSString *streamName;
示例代码：

```
1. _config.streamName = streamName;
```

1.3 状态回调代理 直播状态回调提供给用户连接成功或者连接失败，当失败时提供错误码提醒用户，用户可以根据自身业务情况判断是否进行重连。
直播连接成功回调 类型：方法

- (void)liveContextConnected;

直播断开回调 类型：方法
参数：错误码。

- (void)liveContextError:(RtmpSocketErrorCode)code;

特效直播

直播模块不含音视频采集功能，我们需要使用拍摄器模块进行音视频的采集。当然我们也可以参考直播Demo中的 'BDCloudExternalCapture.m'文件实现外部采集后讲数据传递给直播模块。

示例代码：

```
1. // 1. 开起相机
2. – (void)start {
3.     if (self.isRunning) {
4.         return;
5.     }
6.     // 拍摄配置
7.     BDCloudAVStreamSettings *settings = [BDCloudAVStreamSettings defaultSettings];
8.     // 视频帧率
9.     settings.videoFrameRate = self.setting.frameRate;
10.      // 视频分辨率
11.     settings.dimensions = self.setting.caputreDimension;
12.     // 视频横屏设置
13.     settings.enablelandscapeLeftRecord = self.setting.enablelandscapeLeftRecord;
14.     settings.stabilizationMode = AVCaptureVideoStabilizationModeStandard;
15.     // 直播状态配置（非常关键）
16.     settings.enableLive = YES;
17.     // 根据相机配置初始化相机
18.     self.captureSession = [[BDCloudAVRecordSession sharedInstance] initWithCaptureConfig:settings];
19.     // 拍摄代理
20.     self.captureSession.delegate = self;
21.     if (self.setting.enablelandscapeLeftRecord) {
22.         self.captureSession.view.transform = CGAffineTransformMakeRotation(-M_PI_2);
23.     }
24.     self.captureSession.view.frame = self.preView.bounds;
25.
26.     // 直播支持后台推流
27.     [self.captureSession autoPauseAudioBackground:NO];
28.     // 水印
29.     if (self.setting.enableWatermark) {
30.         [self setWatermark:[UIImage imageNamed:@"pushlogo"]];
31.     }
32.
33.     // 创建特效控制UI（等相机开启回调后添加到preview）
34.     self.captrueControlView = [[BDCloudCaptureView alloc] initWithFrame:[UIScreen mainScreen].bounds
35.                                session:self.captureSession];
36.     self.captrueControlView.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
37.     self.isRunning = YES;
38.
39.     // 注意 captureSession 内含接受手势的view，会添加在 self.captureSession.view 的父view上。
40.     [self.captrueControlView insertSubview:self.captureSession.view atIndex:0];
41.     [self.preView insertSubview:self.captrueControlView atIndex:0];
42. }
43. // 2. 美颜相机配置
44. - (void)captureCameraStarted {
45.     // 添加特效效果（基础美颜）
46.     [self.captureSession applyBeautyBaseVideoFx];
47.     [self.captureSession adjustBeautyWhiteLevel:0.7];
48.     [self.captureSession adjustBeautyBlurLevel:0.7];
49.
50.     // 横屏的UI没有适配暂不开放，客户可以在configBaseUI内进行UI适配后开放此开关。
51.     if (!self.setting.enablelandscapeLeftRecord) {
52.         // 保证AR手势View在最下，防止AR手势View当住了captrueControlView的一些操作。
53.         [self.captrueControlView configBaseUI];
54.     }
```

1.1初始化推流参数并配置 这里我们以直播Demo 'BDCloudLiveContextHelper.mm'文件为例.
示例代码：

```
// 1. 创建并初始化
BDCloudAVLiveParameter *par = [[BDCloudAVLiveParameter alloc] init];
// 2. 配置音频默认编码参数
par.audioSettings = [BDCloudAVAudioOutputSettings defaultSettings];
// 3. 配置视频默认编码参数
par.videoSettings = [BDCloudAVVideoOutputSettings defaultSettings];
// 4. 修改视频编码分辨率
par.videoSettings.dimension =  self.settings.dimension;
// 5. 修改视频编码码率。
par.videoSettings.bitRate = self.settings.bitRate * 1000;
// 6. 修改视频编码帧率
par.videoSettings.frameRate = self.settings.frameRate;
// 7. 修改推流协议
par.streamType = BDCloudAVLiveTypeNormal;
// 8. 设置推流地址
par.url = [NSURL URLWithString:[settings pushUrl]];
if (settings.pushType == BDCloudPushSettingTypeLowerLatency) {
    par.streamType = BDCloudAVLiveTypeLowerLatency;
    par.url = [NSURL URLWithString:[settings pushUrl]];
    // test ip and domain mixer
    if (!par.url) {
        par.c_url = [settings pushUrl];
    }
}
```

1.2 初始化视频直播控制 类型：方法

参数：BDCloudAVLiveParameter 推流参数

返回值：BDCloudAVLiveContext实例

- (instancetype)initWithLiveParameter:(BDCloudAVLiveParameter*)parameter;

示例代码：

```
_liveContext = [[BDCloudAVLiveContext alloc] initWithLiveParameter:par];
```

1.3连接音视频数据源和推流器 示例代码：

```
// 1. BDCloudPushLiveController.mm 文件
// 连接 BDCloudAVRecordSession 和 BDCloudLivePushViewModel
[self.captureViewModel addBufferListener:self.pushViewModel];

// 2. BDCloudLivePushViewModel.mm 文件
// 连接 BDCloudLivePushViewModel 和 BDCloudAVLiveContext
self.videoOutput = [[BDCloudAVOutput alloc] initWithMediaType:AVMediaTypeVideo];
self.audioOutput = [[BDCloudAVOutput alloc] initWithMediaType:AVMediaTypeAudio];
[self.videoOutput addTarget:self.liveHelper.liveContext];
[self.audioOutput addTarget:self.liveHelper.liveContext];

// 3. BDCloudLivePushViewModel.mm 文件
// 数据传输
- (void)frame:(CMSampleBufferRef)frame type:(NSString *)type {
    if ([type isEqualToString:AVMediaTypeVideo]) {
        [self.videoOutput raiseFrame:frame];
    } else if ([type isEqualToString:AVMediaTypeAudio]) {
        [self.audioOutput raiseFrame:frame];
    }
}
```

可以直接数据源（BDCloudAVRecordSession）和推流器（BDCloudAVLiveContext）

示例代码：

```
// 伪代码
[recordSession addBufferListener: liveContext]
```

1.4 设置推流状态监听 推流状态监听必须在开始推流前设置。

类型：属性

默认值：nil

@property(nonatomic, weak) id delegate;

示例代码：

```
_liveContext.delegate = self;
```

1.5 开始推流 类型：方法

返回值：是否开起成功。当重复调用时返回NO。

- (BOOL)start;

示例代码：

```
[self.liveContext start];
```

1.6 结束推流 类型：方法

返回值：是否关闭成功。当重复调用时返回NO。

- (BOOL)stop;

示例代码：

```
[self.liveContext stop];
```

1.7 断网重连 断网重连我们通常是根据代理回调的错误码根据自身业务情况判断是否进行重连。 重连方法必须在开始推流后才能调用。

类型：方法
返回值；重连是否成功。（异步重连，返回值不作为重连的真实结果，真实结果以代理回调为主）

- (BOOL)reconnect；

示例代码：

```
// 1. 重连函数
- (void)reconnect {
    [self.liveContext reconnect];
}
// 2. 根据回调代理的错误码判断是否应该重连。
- (void)liveContextError:(RtmpSocketErrorCode)code {
    dispatch_async(dispatch_get_main_queue(), ^{
        if (self.liveContextStatusChange) {
            self.liveContextStatusChange(NO, code);
        }
    });
    if (code == ErrorCodeLicenseInvalid) {
        NSLog(@"鉴权失败，请检查自己的license文件！！！");
        return;
    } else if (code == -5) {
        NSLog(@"SRT 发送失败...开始重连");
        [self reconnect];
    } else {
        [self reconnect];
    }
}
```

1.8 开始动态码率 动态码率的本质是在推流过程中，通过数据监控判断当前的网络状态是否满足当前的码率进行数据上传，动态的调整当前视频编码的码率，使音视频数据可以高质量的上传到Server。
动态码率主要应用于网络状态不稳定的一些场景，例如户外直播等。
动态码率在开始推流前不生效，但设置后内部会在推流连接成功以后自动开启。

类型：方法
参数：
Sensibility：检测频次，检测频次与动态码率灵敏度成反相关。
maxBitRate: 动态码率的最大码率。
minBitRate: 动态码率的最小码率。
返回值：是否开启成功。当maxBitRate < minBitRate 时返回失败。

- (BOOL)startQOS:(NSUInteger)sensibility maxBitRate:(NSUInteger)maxBitRate minBitRate:(NSUInteger)minBitRate

示例代码：

```
self.liveContext startQOS:5
maxBitRate:self.settings.maxbitRate * 1000
minBitRate:self.settings.minbitRate * 1000];
```

1.9 关闭动态码率 类型：方法

- (void)stopQOS;

示例代码：

```
[self.liveContext stopQOS];
```

1.10 开关静音直播 BDCloudAVLiveContext类中的开关静音方法实际上是通过改变传入的音频数据实现的，如果对于传入的音频数据还需要二次使用，必须慎用。

示例代码：

```
'BDCloudARCaptrue.mm'文件中
- (void)switchAudioSilent:(BOOL)isSilent {
    [self.captureSession switchBufferListenerAudioSilent:isSilent];
```

1.11 自定义图片推流（后台推流） 自定义图片推流是使用指定图片进行推流，此时相机传入的视频数据失效。推流将自定义的图片进行编码打包后上报。
应用场景通常是APP进入后台后，相机无法采集数据时为了让视频数据更加的连贯和反馈给观众端的一种方法。
我们推荐如果APP长时间（超过一个gop时长）退到后台时是一定需要设置的，否则观众拉流可能会因为拉取不到视频帧导致解码器未正确初始化，推流端回到前台后拉流端仍然无法恢复。

示例代码：

```
'BDCloudARCaptrue.mm'文件中
// 退到后台时输出固定图像。
- (void)onAppEnterBackground:(NSNotification *)notification {
    [self.captureSession setBufferListenerImageOutput:[UIImage imageNamed:@"pusbackground"]];
}
// 回到前台时回复相机图像。
- (void)onAppWillEnterForeground:(NSNotification *)notification {
    [self.captureSession setBufferListenerImageOutput:nil];
}
```

1.12 发送Metadata 主播在直播流中可以发送metadata消息。可由百度云播放器解析并通过通知发送给每一个观众用户。常用于直播答题、直播换装等需要由主播添加进实时流内的消息功能。
发送metadata之前需要校验权限，此时就是需要配置该参数。

类型：方法
参数：BDCloudAVLiveMetadataConfig 鉴权及播放流信息
返回值：本地校验是否成功。当本地校验失败时返回NO（本地校验主要针对参数是否非法）

- (BOOL)setConfig:(BDCloudAVLiveMetadataConfig *)config

示例代码： [self.liveContext setConfig:self.metadataPushServer.config]

Metadata发送信息 类型：方法

参数：
NSDictionary:需要发送的json信息。
Block:发送结果和错误码

- (void)pushInfoToMetadata:(NSDictionary *)info complete:(void(^)(BOOL success, NSError* error))complete

示例代码：

```
2. [self.liveContext pushInfoToMetadata:define
3.                          complete:^(BOOL success, NSError *error) {
4.     NSLog(@"asyncPushUserDefine error = %@", error);
5. }];
```

1.13 获取直播上行速度 直播上行速度是指每秒的传送到服务端的数据量。通常用来和视频编码的码率作为对比判断当前的网速和编码码率是否匹配。
类型：参数
默认值：0
@property(nonatomic, readonly) double realtimeUploadSpeed; 示例代码；

```
1. double realtimeUploadSpeed = liveContext.realtimeUploadSpeed
```

1.14 获取直播上行帧率 直播上行帧率是指每秒的传送到服务端的视频帧数。通常用来和视频编码的帧率作为对比判断当前的推流卡顿程度。
类型：参数
默认值：0
@property(nonatomic, readonly) double realtimeUploadFPS; 示例代码：

```
1. double realtimeUploadFPS = liveContext.realtimeUploadFPS
```

1.15 获取视频编码码率 视频编码码率是指视频编码时真实每秒产出的视频数据量大小。尽管我们编码设置了目标码率，但是并不是真实产生的编码数据量的大小。通常我们会通过比较真实的编码码率和真实的上行速度来判断当前的网络宽带是否够用。
类型：参数
默认值：0
@property(nonatomic, readonly) double realtimeBitrate;

示例代码：

```
1. double realtimeBitrate = liveContext.realtimeBitrate
```

1.16 获取推流缓冲区数据占比 音视频数据编码打包后进入推流缓冲区，推流协议模块有序从推流缓冲区取出数据进行上传。推流缓冲区能够体现网络延迟和当前上行不满足当前码率等问题。
推流缓冲区占比数据当前仅适用于RTMP协议。
类型：参数
默认值：0
@property(nonatomic, readonly) double packetCacheRate; 示例代码：

```
1. double packetCacheRate = liveContext.packetCacheRate
```

1.17 获取往返延迟RTT 在使用SRT传输协议时，可以实时获取往返延迟数据。
RTT参数当前仅适用于SRT协议。
类型：参数
默认值：0
@property(nonatomic, readonly) double msRTT; 示例代码：

```
1. double msRTT = liveContext.msRTT
```

1.18 获取预估上行带宽 在使用SRT传输协议时，可以实时获取预估的上行带宽。需要注意区分预估带宽和真实上行的区别。
预估上行带宽参数当前仅适用于SRT协议。
类型：参数
默认值：0
@property(nonatomic, readonly) double mbpsBandwidth;

示例代码：

```
1. double mbpsBandwidth = liveContext.mbpsBandwidth
```

1.19 获取交互延迟 在使用SRT传输协议时，可以实时获取交互延迟。
交互延迟参数当前仅适用于SRT协议。
类型：参数
默认值：0
@property(nonatomic, readonly) double peerlatency; 示例代码：

```
1. double peerlatency= liveContext.peerlatency
```

互动直播

音视频互动直播是指两个或两个以上人通过连麦方式进行超低延迟的互动，并将互动的音视频数据传递其他的不同观众。**音视频互动直播通常是连麦互动场景。通过连麦可以让主播和连麦观众进行超低延迟的音视频通话，然后由主播将互动数据分发给直播观众。**

互动直播分为两个部分：
1、连麦加入同一个房间内进行超低延迟音视频互动。
2、将连麦房间内的音视频数据进行直播分发。

我们提供两种形式：
a) **服务端混流并分发直播**：由RTC的服务端进行音视频混流，同时将混流后的数据推送给LSS直播服务器，LSS直播服务器直接下发直播流。
i. 优点：网络消耗更少，CPU和GPU压力小，能够适配低端机型。
ii. 缺点：RTC内部为了实时性会动态压缩画面质量，直播出去的清晰度可能不足；由普通推流直播和互动直播服务端转推切换时，观众端播放器需要进行重连拉流，造成卡顿。

b) **本地混流并分发直播：**由RTC互动模块产出音视频数据，并由直播推流模块进行推流直播。
i. 优点：画面质量只与预设值相关，不会动态调整。由普通推流直播和互动直播本地混流切换时，观众端播放器可以流畅过度。
ii. 缺点：网络消耗更多，CPU和GPU压力答，需要高端机型适配。

1.1 设置鉴权信息appID 实时音视频通讯appId和token的获取：https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu 类型；参数
默认值：nil
@property (nonatomic, copy) NSString *appId;

示例代码：

```
1. BDCloudAVRtcRoomParameter *room = [[BDCloudAVRtcRoomParameter alloc] init];
2. room.appId = settings.rtcAppID;
```

1.2 设置鉴权信息token 实时音视频通讯appId和token的获取：https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu 类型；参数
默认值：nil
@property (nonatomic, copy) NSString *token;

示例代码：

```
1. room.token = settings.rtcToken;
```

1.3 设置房间信息roomName 主播和连麦观众必须加入到同一个互动房间才能进行超低延迟会话。因此roomName两个人必须保持一致。
类型；参数
默认值：nil
@property (nonatomic, copy) NSString *roomName;

示例代码：

```
1. room.roomName = settings.roomName;
```

1.4 设置互动房间内自己的信息userId 为了区分互动房间内的不同人，我们使用userId来区别房间内的不同用户。
注意：同一个房间（roomName）内，一定不能使用两个相同的userId。
类型；参数
默认值：0
@property (nonatomic, assign) NSInteger userId;

示例代码：

```
1. room.userId = settings.userID;
```

1.5 设置互动房间内自己的信息userName 房间内不同的虽然不允许相同的userId,但是允许相同的userName。是为了显示房间内的用户更加的灵活。
类型；参数
默认值：nil
@property (nonatomic, copy) NSString *userName;

示例代码：

```
1. room.userName= settings.userName;
```

1.6 设置纯音频互动 纯音频互动下，能看到对方画面和听到对方声音。但对方不能看到本端画面，只能听到本端声音。
类型；参数
默认值：NO
@property (nonatomic, assign) BOOL isOnlyAudio;

示例代码：

```
1. output.isOnlyAudio = settings.isOnlyAudio;
```

1.7 设置允许多人互动 受移动端手机性能影响，我们不建议超过2人的音视频互动直播。SDK内我们的最大上限是房间内最多6人。但是在互动直播场景下，我们希望房间内视频通道最多2条（即最多两人开起视频通道，其他人都使用纯音频互动）
类型；参数
默认值：NO
@property (nonatomic, assign) BOOL enableMutilVideoPlayer;

示例代码：

```
1. output.enableMutilVideoPlayer = NO;
```

1.8 设置输出视频混流模式 互动房间内的视频是多路视频，如果我们希望他们混合成一路视频画面，传递给下一个处理模块，此时就需要设置视频混流的模式。
当前我们支持左右、上下、内嵌三种模式将本地和远端的视频混合在一起。
类型：参数
默认值：BDCloudAVMetalBufferBlendTypeHorizontal
@property (nonatomic, assign) BDCloudAVMetalBufferBlendType outputVideoBlendType;

示例代码：

```
1. // 这里我们设置成内嵌模式
2. output.outputVideoBlendType = BDCloudAVMetalBufferBlendTypeEmbedded;
```

1.9 设置输出视频混流内嵌模式下的起点坐标 当设置成了内嵌模式，远端画面相对于本地画面是可以设置起始位置的，更加的定制化。 注意：该参数仅适用于内嵌模式
类型：参数 默认值：{ 0 , 0 } @property (nonatomic, assign) CGPoint outputVideoBlendRemotePosition;

示例代码：

```
// 这里我们设置成内左上角
output.outputVideoBlendRemotePosition = CGPointMake(0, 0);
```

1.10 设置输出视频混流内嵌模式下的缩放系数 当设置成了内嵌模式，远端画面相对于本地画面是可以设置缩放比的，更加的定制化。 注意：该参数仅适用于内嵌模式

类型：参数

默认值：0.5

@property (nonatomic, assign) float outputVideoBlendRemoteScale;

示例代码：

```
// 这里我们设置0.3倍
output. outputVideoBlendRemoteScale = 0.3;
```

超低延迟直播

概述 百度智能云超低延时直播（BRTCPlayer）是在百度智能云视频直播的基础上通过链路传输协议优化，解决传统直播延迟过高的弊端，通过集成百度智能云播放器SDK，可以支持千万级并发场景的毫秒级延时直播，并为用户提供低卡顿、秒开流畅的直播观看体验。

功能介绍 BRTCPlayer 主要功能如下：

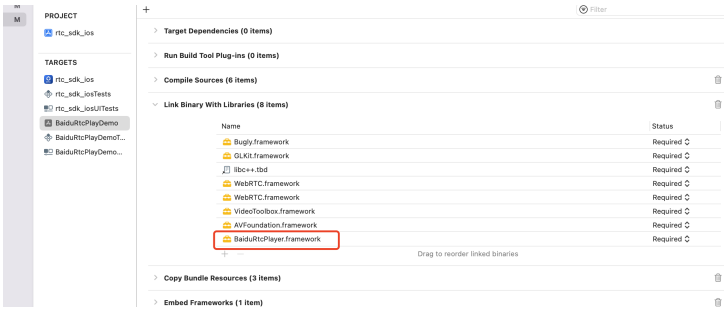
- 支持UDP信令模式：除HTTP标准信令模式外支持UDP信令模式，0 RTT 极速启播，有效提升秒开率；
- 支持AVC、HEVC视频硬解码；
- 支持视频B帧：引入B帧，可在清晰率不变的情况下降低码率；
- 支持AAC音频解码，支持MP4A-ADTS格式 48000 及 44100 采样率；
- 支持视频平滑渲染： 网络抖动时，平滑渲染视频帧, 降低卡顿；
- 支持最小延迟动态配置：可实现延迟换流畅，在允许范围内增大延迟可有效降低卡顿；
- 支持网络健康度实时检测；
- 内置HTTPDns支持；
- 支持 SEI 消息；

Demo体验 前往DEMO下载页面扫码安装DEMO

SDK 下载 前往SDK下载页面下载支持超低延时直播的智能视频SDK V1.8.0 及以后版本， 内含Demo源码；

快速集成 百度云超低延时直播由独立SDK提供支持，用户可通过在播放侧单独集成BRTCPlayer SDK实现超低延时直播，而推流侧则可复用当前使用的推流器，仅需在百度直播LSS平台针对当前域名开通低延时播放能力。 BRTCPlayer SDK具体的集成步骤如下：

- BRTCPlayer 作为独立SDK 提供低延时播放能力，包体精简，BaiduRtcPlayer.framework 包体约1.78M；
- 导入BaiduRtcPlayer.framework到工程，集成后的示例如下：



快速开始

1. 创建播放渲染视窗

```
// 在布局文件中添加渲染控件
[self.playerView addSubview:self.bdPlayer.view];
// 寻回渲染控件对象
```

2. 创建播放器对象及播放器初始化

```
// 创建播放参数集，初始化播放器
BaiduRtcPlayerParameter *param = [BaiduRtcPlayerParameter defaultParameter];
    param.externalSetActive = YES;
    param.enableDtlsSrtp = NO;
// 初始化播放器 传入播放参数及事件监听器
self.bdPlayer = [[BaiduRtcPlayer alloc] initWithParameter:param
                                    delegate:self];

// 设置播放器信令服务地址
[self._bdPlayer setSignalServer:@"http://test-pl-central.bigenemy.cn/brtc/v3/pullstream"]
// UDP 信令模式需要进行相关预设
if (bUseUdpSignaling) {
    //设置播放地址
    [self._bdPlayer setMediaServerIp:ip];
    // 设置Udp信令模式
    [self.bdPlayer setSingalMode:BaiduRtcPlayerSingalModeOverHttp];
    // 由业务侧确定是否为H265格式,默认为H264格式； udp 信令模式需设置视频编码
    [self.bdPlayer setCodecType:self->_boCodec265Type ? CODEC_H265 : CODEC_H264];
} else {
    //设置http信令模式
    [self._bdPlayer setSingalMode:BaiduRtcPlayerSingalModeOverHttp];
}
```

3. 播放设置

```
// 设置媒体流地址
[self.bdPlayer prepareToPlay:@"webrtc://test-pl-central.bigenemy.cn/myapp/test321.flv" autoPlay:NO];
```

4. 播放控制

```
// 准备播放，获取播放依赖资源
- (void)prepareToPlay;
// 开始播放
- (void)startPlay;
// 暂停播放
- (void)pausePlay;
// 恢复播放
- (void)resumePlay;
// 停止播放
- (void)stopPlay;
```

5. 释放播放器

```
// 释放播放器
```

在智能视频SDK Demo中对上述流程有详细的展示，可以参考；

接口说明 BaiduRtcPlayerParameter类 | 接口名 | 描述 | |----------|----------| | netStatus | 网络类型 wifi 4g | | audioEnable | 是否播放音频 | | videoEnable | 是否播放视频 | | | audioVolume | 设置静音启播 | | logReportEnable | 启动日志上报监控 设置最小延时， 可通过该参数增大延时降低卡顿提升播放流畅度 | | signalServer | 设置信令服务器 | | palyer_url | 设置播放地址 | | enableDtlsSrtp | 设置数据加密 | | mediaServerIP | 设置播放地址 | | videoCodectype | UDP模式使用 设置视频解码格式，由业务侧确定是否为H265格式，默认为H264格式，仅UDP信令模式 | | videoBFrame |是否开启B帧支持，由业务侧确定是否开启，默认开启 ，仅UDP信令模式
| sampleRate | 设置音频播放采样率[48000, 44100]， 仅UDP信令模式
|signalMode | 低延迟直播的信令交互模式http方式udp方式 |

BaiduRtcPlayerDelegate类 | 错误码 | 含义 | |----------|----------| | BaiduRtcPlayerErrorInvalidUrl = 10000 | URL 格式错误 | | BaiduRtcPlayerErrorIceFailed = 10001 | ICE 连接错误 | | BaiduRtcPlayerErrorIceDisconnected = 10002 | 替换预留错误码 | | BaiduRtcPlayerErrorConnection = 10003 | 连接创建失败 | | BaiduRtcPlayerErrorGetLocalSdpFailed = 10004 | 媒体描述请求失败 | | BaiduRtcPlayerErrorSetLocalSdpFailed = 10005 | 媒体描述设置失败 | | BaiduRtcPlayerErrorGetRemoteSdpFailed = 10006 | 远端媒体描述请求失败,一般由于启播放时主播侧已停播 | | BaiduRtcPlayerErrorSetRemoteSdpFailed = 10007 | 远端媒体描述设置失败 | | BaiduRtcPlayerErrorInvalidStatus = 10008 | 播放状态错误 | | BaiduRtcPlayerErrorNullVideoFrame = 10009 | 媒体流中断，一段时间未收媒体流，SDK内部检测到断流错误后会立即停止播放，一般在播放中主播停播但业务侧未退出 | | BaiduRtcPlayerErrorLoadLibaraies = 10010 | SO库文件后下载失败 | | BaiduRtcPlayerErrorTimeOut = 100011 | license错误 | | BaiduRtcPlayerErrorUdpUnsupport = 10012 | 当前license不包含低延时播放feature | |

事件码	**含义**
BaiduRtcPlayerInfoRoomRender = 1000	开始远端渲染
BaiduRtcPlayerInfoIceConnected = 1001	ICE连接成功
BaiduRtcPlayerInfoPeerConnectionClosed = 1002	对端连接关闭
BaiduRtcPlayerInfoStatsUpdated = 1003	媒体流信息更新
BaiduRtcPlayerInfoBufferingStart = 1004	Buffering start事件
BaiduRtcPlayerInfoBufferingEnd = 1005	Buffering end事件
BaiduRtcPlayerInfoIceDisconnected = 1006	ICE连接断开
BRTC_PLAYER_EVENT_NO_STREAMING_DETECTED = 1007	没有检测到媒体流
BRTC_PLAYER_EVENT_PLAY_TIME_STATISTIC = 1008	启播各阶段耗时统计
BaiduRtcPlayerInfoLocalSDPSetted = 1009	local sdp 设置完成
BaiduRtcPlayerInfoRemoteSDPRequested = 1010	remote sdp 请求完成

事件回调	**含义**
- (void)onPrepared:(BaiduRtcPlayer *)player elapse:(NSInteger)interval;	播放器准备就绪
- (void)onFirstVideoFrameRendered:(BaiduRtcPlayer *)player elapse:(NSInteger)interval;	首帧渲染事件
- (void)onResolutionChanged:(BaiduRtcPlayer *)player size:(CGSize)size;	分辨率更新回调
- (void)onPlayerError:(BaiduRtcPlayer *)player errorCode:(BaiduRtcPlayerErrorCode)errorCode error:(nullable NSError *)error;	错误错误回调，错误码定义见前文
- (void)onPlayerInfo:(BaiduRtcPlayer *)player infoCode:(BaiduRtcPlayerInfoCode)infoCode obj:(NSObject *)obj;	事件更新回调，事件码定义见前文
- (void)onPeriodRenderStatis:(BaiduRtcPlayer *)player frame:(NSInteger)frameRate;	播放状态更新
- (void)onPlayerReceivedSEI:(BaiduRtcPlayer *)player sei:(NSData *)sei;	接收到SEI 消息，回调在解码线程,不应在该回调里实现复杂业务

> SEI 数据格式(h264):[nal_type(1)|sei_type(1)|sei_size(n+1)|sei_payload(n*255+m)|trailing_bits(1)]
> sei_size: 需判断第3个字节起连续0xFF的个数n，加第1个不为0xFF的字节(如: 0x21)，该字段占字节数为 n + 1; 其值为:0xFF*n+0x21;
> sei_payload: SEI字段的值（h265）；如:[0x06 0x05 0x18 0x54 0x80 0x83 0x97 0xF0 0x23 0x47 0x4B 0xB7 0xF7 0x4F 0x32 0xB5 0x4E 0x06 0xAC 0x27 0x11 0xFE 0x69 0x79 0x01 0x00 0x00 0x80]
> 注： h265 与 h264 的 sei 格式略有不同， h264 的nal_type 为 0x06, sei_type 为0x 05; h265 的 nal_type 为 0x4e, sei_type 为两字节 0x01、0x05;

BRTCPlayer接口 | 接口名 | 描述 | |----------|----------| | initWithParameter:(BaiduRtcPlayerParameter*) delegate: | 初始化播放器。 playParameters 播放参数集 delte 播放事件监听 | |- (void)setCodecType:(CodecType) typeCodec | 设置视频编码类型 UDP模式支持 | | - (void)setScalingMode:(BaiduRtcPlayerScalingMode)mode; | 设置播放视窗缩放模式 | |- (void)prepareToPlay:(NSString *)url autoPlay:(BOOL)autoPlay; | 准备播放 | | - (void)startPlay; | 开始播放 | | - (void)pausePlay; | 暂停播放,不停止拉流，仅暂停本地媒体流渲染 | | - (void)resumePlay; | 恢复播放,从暂停状态恢复播放 | |- (void)stopPlay; | 停止播放，停止媒体拉流及渲染，再次播放需调用startPlay() | | - (void)setSingalMode:(BaiduRtcPlayerSingalMode)mode; | 设置信令模式 | |- (void)setMediaServerIp:(NSString *)mediaServerIp; | UDP模式设置播放url媒体ip | | - (void)setSignalServer:(NSString *)signalServer; | 设置信令地址，HTTP模式设置 | | - (void)setVoulme:(float)volume; | 设备播放音量 | | - (void)setBufferLevel:(NSInteger) bufferInterval; | 设置播放媒体手动延迟 | | - (void)setStatistics:(BOOL) on | 日志统计开关 |

Android开发说明

Android接入概述

本文主要介绍如何快速地将SDK（Android）集成到您的项目中，按照如下步骤进行配置，就可以完成SDK的集成工作。

1.1 适用场景

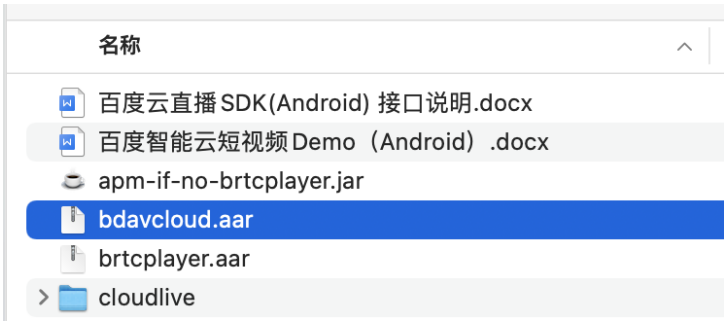
互动直播
在直播推流及RTC实时视频通话场景下，不仅可以使用美颜、美妆、滤镜等进行视频特效美化，还能通过丰富的2D/3D贴纸、人脸/肢体特效游戏、AR互动礼物等提升场景的趣味性与互动性，激发更高商业价值。

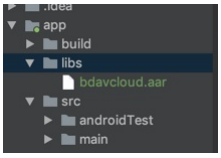
1.2 开发环境要求

Android Studio 3.2或以上版本，Gradle 4.6或以上版本。
编译环境请选择支持java8。
Android 4.4系统以上，API Level 19以上。

****1.3 下载并集成SDK**
第一步：请前往百度智能云平台下载最新版本拍摄器SDK。

注意：必须是将license申请完后，才能下载
第二步：将下载解压缩之后的SDK目录下的bdavcloud.aar文件拷贝到工程的app/libs*目录下。 注：v1.8.0 版本后，如需要集成超低延时直播SDK则新增导入brtcplayer.aar,若不集成超低延时播放SDK brtcplayer.aar，则需要同时导入apm-*.jar;





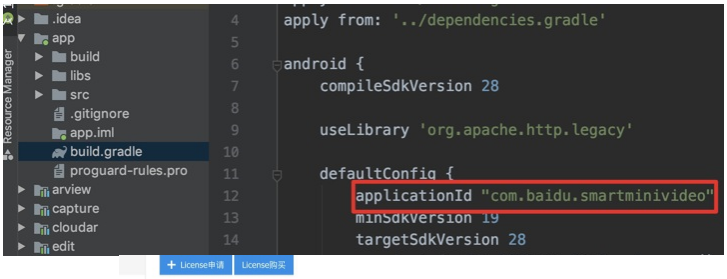
第三步：在项目build.gradle添加库依赖。

```
api fileTree(dir: 'libs', include: ['*.aar'])
// 外部第三方包
implementation 'com.google.code.gson:gson:2.7'
implementation 'com.googlecode.mp4parser:isoparser:1.0.1'
implementation 'com.googlecode.plist:dd-plist:1.16'
// retrofit
implementation 'com.squareup.retrofit2:retrofit:2.1.0'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
implementation 'com.squareup.retrofit2:adapter-rxjava:2.1.0'
// okhttp
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
```

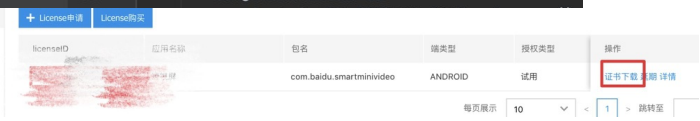
第四步：同步Sdk，单击Sync Now按钮，完成短视频sdk的集成工作。

注意：在使用拍摄器SDK，需要申请产品对应的授权文件，如无授权，产品无法正常使用。

****1.4 license申请**
** 第一步：请到百度智能云平台进行license申请。
第二步：填写应用信息、选择服务、素材选择、授权信息、立即申请，申请相应的服务（注意：在gradle中修改包名）。

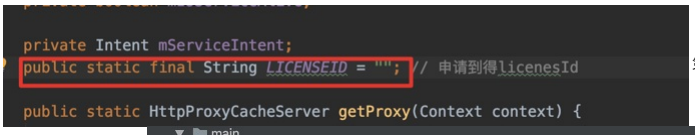


第三步：申请成功后，会得到一个licenseID和对应授权文件下载地址，下载成功后，需要手动添加到项目工程中。

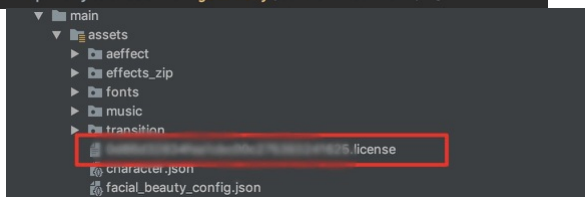


****1.5 license配置**
** 第一步：修改自定义Application类，替换自己申请的licenseID。

（可参考demo 中的SmartminivideoApplication类）



第二步：替换app/src/main/assets中后缀名为.license的文件为上面申请的license文件（licenseID对应license文件）。



注意：接入sdk必须在SmartminivideoApplication初始化sdk。

****1.6 配置APP权限**

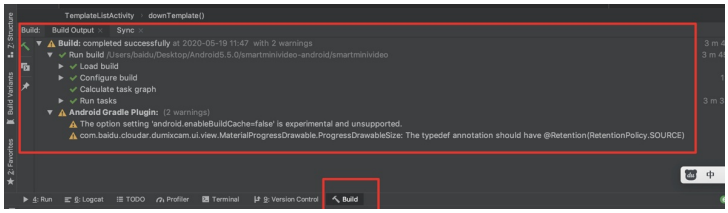
** 在 AndroidManifest.xml 中配置 App 的权限，短视屏sdk 需要以下权限：

```
<!--允许应用程序更改WIFI连接状态-->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<!--网络权限:访问网络连接，请求数据-->
<uses-permission android:name="android.permission.INTERNET" />
<!--读取手机状态和身份：统计设备信息,用于账号体系状态同步-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--写入外部存储卡：保存图片和视频至本地文件系统-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!--查看WIFI连接连接权限：获取网络状态，用于分享时判断网络状态-->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--允许应用程序访问有关网路的信息-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- Add read  camera permissions -->
<!--录音权限:视频录音-->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!--摄像头权限：获取相机权限进行图片拍照、摄像等操作-->
<uses-permission android:name="android.permission.CAMERA" />

<!-- interactive live use rtc -->
<!--允许应用程序更改网络连接状态-->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<!--修改声音设置信息-->
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

<!-- push sdk start-->
<!-- 适配Android N系统必需的ContentProvider写权限声明，写权限包含应用包名 -->
<uses-permission android:name="baidu.push.permission.WRITE_PUSHINFOPROVIDER.com.baidu.smartminivideo" />
```

****1.7 运行**** Build没有相应的错误可视为运行成功：



拉流

音视频直播拉流相对与推流场景更加的常见。主要是使用支持实时在线流的播放器进行播放。（支持HTTP-FLV;、HLS、SRT等协议的播放器）

百度云同样提供播放器SDK提供直播拉流服务，详细请参考：https://cloud.baidu.com/doc/MCT/s/2jwvz54zo

推流参数配置

LiveConfig用于配置视频采集/编码参数、音频采集/编码参数、及推流参数配置等。

视频采集/编码配置信息包括：

前后置摄像头：设置初始化拍摄器时使用前置或后置相机。

摄像头旋转角度：设置摄像头方向。

视频分辨率：视频采集及编码分辨率，视频编码场景中分辨率与清晰度正相关。

视频帧率：视频采集及编码帧率，视频编码场景中帧率与画面连贯正相关。

视频码率：视频编码码率，视频编码场景中码率与清晰度正相关。

视频gop长度：视频编码关键帧间隔，视频编码场景中关键帧间隔时间。

音频采集/编码配置信息包括：

音频采样率：音频编码采样率，音频编码场景中可以修改音频的采样率。

音频通道数：音频通道数，可配置单通道及立体声双通道。

音频码率：音频编码码率，音频编码场景中码率与清晰度成正相关。

推流配置信息包括：

横屏模式推流设置：设置横屏模式推流。

断流重连设置：设置SDK内部重连次数。

推流码率自适应：设置是否开启推流码率自适应及码率参数。

图片推流设置：设置开启推流暂停时是否推静态图片及设置图片地址。

对于不熟悉音视频产品的用户，直播SDK提供默认的参数配置，方便用户初始化。对于熟悉音视频直播推流的用户，可以自行修改参数以适配业务场景。

1.1视频采集与编码参数配置 视频采集支持设置前后置摄像头、摄像头旋转角度、分辨率、帧率、预览翻转。视频编码参数支持分辨率、帧率、码率、关键帧间隔。相关接口及说明如下：

```
 // 设置摄像头ID，默认为前置摄像头CAMERA_FACING_FRONT
public final Builder setCameraId(int cameraId)

// 设置摄像头方向，该参数最终用于Camera的setDisplayOrientation接口, 默认竖屏 0
public final Builder setCameraOrientation(int cameraOrientation)

// 设置采集及输出视频宽度, 默认 1280
public final Builder setVideoWidth(int videoWidth)

// 设置采集及输出视频高度, 默认 720
public final Builder setVideoHeight(int videoHeight)

// 设置视频采集及编码帧率, 默认25fps
public final Builder setVideoFPS(int videoFPS)

// 设置视频初始码率, 默认1024000 bps
public final Builder setInitVideoBitrate(int initVideoBitrate)

// 设置I帧间隔时长，单位为秒，默认为2秒
public final Builder setGopLengthInSeconds(int gopLengthInSeconds)

// 设置本地预览初始水平翻转
public final Builder setPreviewHFlip(boolean preivewFlip)

// 是否使能视频推流，默认为true. 若设置为false,可实现纯音频推流
public final Builder setVideoEnabled(boolean videoEnabled)
```

1.2 音频采集及编码参数配置 音频采集编码配置支持音频采集率、音频通道及音频码率设置，相关接口如下：

```
 // 设置音频采样率 默认 44100
public final Builder setAudioSampleRate(int audioSampleRate)

// 设置音频通道数 默认 2
public final Builder setAudioChannels(int channels)

// 设置音频码率
public final Builder setAudioBitrate(int audioBitrate)

// 是使能音频推流，默认为true. 若setVideoEnabled(false) 则进行纯音频推流
public final Builder setAudioEnabled(boolean audioEnabled)
```

注：音频采样率44100 是各平台普通支持的采样率，不容易出现兼容性问题。

1.3 推流参数配置 可通过如下接口设置推流及视频连麦时远端视频是否默认水平翻转：

```
 // 设置直播推流初始水平翻转
public final Builder setLiveHFlip(boolean liveFlip)
// 设置连麦互动推流水平翻转
public final Builder setInteractHFlip(boolean interactHFlip)
```

推流视频翻转

可通过如下接口设置推流及视频连麦时远端视频是否默认水平翻转：

```
 // 设置直播推流初始水平翻转
public final Builder setLiveHFlip(boolean liveFlip)
// 设置连麦互动推流水平翻转
public final Builder setInteractHFlip(boolean interactHFlip)
```

横屏推流

```
 /**
 * 设置屏幕方向
 * @param orientation 传入屏幕方向 横屏{@link ActivityInfo#SCREEN_ORIENTATION_LANDSCAPE}
 *                    或竖屏{@link ActivityInfo#SCREEN_ORIENTATION_PORTRAIT}
 */
public final Builder setScreenOrientation(int orientation)
```

注: 可通过getWindowManager().getDefaultDisplay().getRotation() 获得旋转角度分析屏幕方向。

断流重连

当直播推流遇到异常后，SDK 支持内部重连,可设置内部重连次数：

```
 // 设置内部出错重连次数 默认 3次
public final Builder setReconnectTimes(int reconnectTimes)
```

注：SDK 内部重试N次仍然失败后，用户可在监听到相关Error后再次发起外部重连。

推流码率自适应

SDK 支持根据当前推流质量动态切换推流码率，相关的设置接口包括：

```
 // 开启或关闭动态码率自动调整
public final Builder setQosEnabled(boolean qosEnabled)
```

SDK 内部通过多次探测推流质量后调整一次码率，每次探测的周期为2S,参数 qosSensitivity 为调整一次码率的推流质量探测次数。

调整一次码率的周期为：qosSensitivity*2s

```
 // 设置动态码率灵敏度-探测几次调整一次码率 范围[5, 10]
public final Builder setQosSensitivity(int qosSensitivity)
// 动态码率设置-视频最大码率 默认 1024000 bps
public final Builder setMaxVideoBitrate(int maxVideoBitrate)
// 动态码率设置-视频最小码率 默认 200000 bps
public final Builder setMinVideoBitrate(int minVideoBitrate)
```

注：最大码率不得小于100000bps, 最小码率不得小于100000bps

图片推流

推流配置支持开启及配置图片推流，若开启图片推流则可以使得在推流页进入后台时继续推流，由于推流进入后台，摄像头被动关闭，可通过配置一个静态图片，当后台推流画面呈现预设图片。若关闭后台推流，则推流进入后台时暂停音视频推流，声音、画面静止。

```
 // 设置是否开图片推流
public final Builder setPicStreamingEnabled(boolean enabled)
// 设置推流暂停图片位置
public final Builder setPausePicPath(String pausePicPath)
```

注：推流暂停图片可以是sdcard的本地图片，也可以是apk 里的resource。若为sdcard 本地图片则path 为图片文件绝对路径，若为rec/drawable 下的png 格式 resource 则path只需传入文件名称。

外部采集

直播SDK支持音视频源外部采集即外部媒体源导入，通过下面两个接口分别开启/关闭视频及音频的外部采集。

```
 /**
 * 设置视频外部采集
 * @param extAudioCaptureEnabled 是否开启视频外部采集
 * @return
 */
public final Builder setExtAudioCaptureEnabled(boolean extAudioCaptureEnabled)

/**
 * 设置音频外部采集
 * @param extAudioCaptureEnabled 是否开启音频外部采集
 * @return
 */
public final Builder setExtVideoCaptureEnabled(boolean extVideoCaptureEnabled)
```

注：当前ARMediaStreamingPusher 及InteractStreamingPusher 支持音视频外部采集。使用外部采集，除使能外部采集外，还需要通过数据导入接口输入音视频数据。

普通直播

普通推流拥有与AR直播相同的通用接口，不同的是普通推流使用SurfaceView 作为本地预览控件，视频直接通过相机获取，相机接口与AR直播使用的AR 相机接口不同。

设置本地预览视窗

```
 /**
 * 设置本地预览SurfaceHolder
 * @param surfaceHolder
 */
public void setSurfaceHolder(SurfaceHolder surfaceHolder)
```

设置远端镜像

```
 /**
 * 设置远端视频翻转
 * @param flip
 */
public void toggleRemoteFlip(boolean flip)
```

本地预览镜像

```
 /**
 * 设置预览视频水平翻转
 * @param flip
 */
public void togglePreviewFlip(boolean flip)
```

切换前后摄像头

```
 /**
 * 切换拍摄头
 * @param cameraId 相机id
 */
public void switchCamera(int cameraId)
```

设置对焦点

```
 /**
 * 设置对焦焦点
 *
 * @param x 焦点横坐标
 * @param y 焦点纵坐标
 */
public void focusToPoint(int x, int y)
```

设置焦距

```
/**
* 设置相机放大因子
*
* @param factor zoom value. The valid range is 0 to {@link #getMaxZoom}
*/
public boolean setZoomFactor(int factor)
```

开关闪光灯

```
/**
* 开关闪光灯
* @param flag 是否开启闪光灯
*/
public void toggleFlash(boolean flag)
```

互动直播

音视频互动直播是指两个或两个以上人通过连麦方式进行超低延迟的互动，并将互动的音视频数据传递其他的不同观众。**音视频互动直播通常是连麦互动场景。通过连麦可以让主播和连麦观众进行超低延迟的音视频通话，然后由主播将互动数据分发给直播观众。**

互动直播分为两个部分：
1、连麦加入同一个房间内进行超低延迟音视频互动。
2、将连麦房间内的音视频数据进行直播分发。

我们提供两种形式：
a) **服务端混流并分发直播**：由RTC的服务端进行音视频混流，同时将混流后的数据推送给LSS直播服务器，LSS直播服务器直接下发直播流。
i. 优点：网络消耗更少，CPU和GPU压力小，能够适配低端机型。
ii. 缺点：RTC内部为了实时性会动态压缩画面质量，直播出去的清晰度可能不足；由普通推流直播和互动直播服务端转推切换时，观众端播放器需要进行重连拉流，造成卡顿。
b) **本地混流并分发直播：**由RTC互动模块产出音视频数据，并由直播推流模块进行推流直播。
i. 优点：画面质量只与预设值相关，不会动态调整。由普通推流直播和互动直播本地混流切换时，观众端播放器可以流畅过度。
ii. 缺点：网络消耗更多，CPU和GPU压力答，需要高端机型适配。

1.1 互动通道基本配置 互动通道基本配置包括互动模块鉴权信息（appId、token）、用户标识（userId、userName）及互动通道信息（roomName、mediaServer）等必选设置。

鉴权信息appID
实时音视频通讯appId和token的获取：https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu

鉴权信息token
实时音视频通讯appId和token的获取：https://cloud.baidu.com/doc/RTC/s/Qjxbh7jpu

互动房间内自己的信息userId
为了区分互动房间内的不同人，我们使用userId来区别房间内的不同用户。 注意：同一个房间（roomName）内，一定不能使用两个相同的userId。

互动房间内自己的信息userName
房间内不同的虽然不允许相同的userId,但是允许相同的userName。是为了显示房间内的用户更加的灵活。

房间信息roomName
主播和连麦观众必须加入到同一个互动房间才能进行超低延迟会话。因此roomName两个人必须保持一致。

信令服务地址 mediaServer
互动通道信令服务器地址，用于客户端与媒体服务器的信令交互。 默认信令服务地址：wss://rtc.exp.bcelive.com:8989/janus

互动通道基本配置接口：

```
// 设置互动通道appid
public final Builder setAppId(String appId)
// 设置互动通道token
public final Builder setTokenStr(String tokenStr)
// 设置用户id
public final Builder setUserId(int userId)
// 设置用户名
public final Builder setUserName(String userName)
// 设置互动通道房间
public final Builder setRoomName(String roomName)
// 设置互动通道信令服务器地址
public final Builder setMediaServer(String mediaServer)
```

1.2 互动通道媒体参数及转推配置 设置纯音频互动
纯音频互动下，能看到对方画面和听到对方声音。但对方不能看到本端画面，只能听到本端声音。

```
// 音频互动
public final Builder setAudioOnly(boolean audioOnly)
```

音频外部采集
互动模块音频外部采集是相对外部直播模块而言，互动模块由部具备独立音频采集功能，同时直播模块也具备打开音频设置采集音频的能力， 由于音频设备的独占性，不能在互动直播时同时开启互动模块及直播模块的音频采集功能，所以在互动直播时需要设置互动模块音频外部采集，即从外部直播模块采集音频传入互动模块。

```
// 是否开启音频外部采集 默认开启
public final Builder setExternalAudioRecord(boolean enableExternalAudioRecord
```

)

互动通道音频采集率设置

```
// 设置互动通道音频采样率
public final Builder setAudioSampleRate(int sampleRate)
注意：当开启音频外部采集，互动通道的音频采集自动与直播音频采集率一致。
 互动通道音频通道数设置
// 设置互动通道音频通道数
public final Builder setAudioChannels(int channels)
```

注意：当开启音频外部采集，互动通道的音频通道自动与直播音频通道一致。

视频分辨率及码率设置

```
 // 设置互动通信视频分辨及码率 格式: WxH-kbps,如:544x960-1000kbps
public final Builder setVideoResolution(String videoResolution)
```

码率还可以通过如下接口设置，上面方法的码率设置优化级高。

```
 // 设置互动通道视频比特率
public final Builder setVideoBitrate(int videoBitrate)****
```

互动通道视频帧率设置

```
 // 设置互动通道视频帧率
public final Builder setVideoFramerate(int videoFramerate)
```

服务转推使能

```
 // 设置服务端转推使能
public final Builder setEnableTransfer(boolean rtcMix)
```

互动混流模式

```
 public enum FlowMixMode {
   FLOW_MIX_NONE,
   // 服务端混流
   SERVER_FLOW_MIX,
   // 本地音频混流
   LOCAL_AUDIO_FLOW_MIX,
   // 本地音视频混流
   LOCAL_AV_FLOW_MIX,
}

// 设置混流模式
public final Builder setFlowMixMode(FlowMixMode mode)
```

1.3 互动推流器初始化与直播 互动通道参数配置

参考上节互动直播参数配置，主要涉及鉴权信息（appId、token）、用户标识（userId、userName）及互动通道信息（roomName、mediaServer）等基本配置 及 互动媒体通道与转推配置。

互动推流器初始化

```
 /**
 * 互动直播推流器构造方法
 * @param context 上下文环境
 * @param surfaceView 本地预览视窗
 * @param liveConfig 推流配置
 * @param interactConfig 互动配置
 */
public InteractStreamingPusher(Context context, GLSurfaceView surfaceView, LiveConfig liveConfig, InteractConfig interactConfig)
```

设置连麦远端视频视窗

```
 /**
 * 设置连麦远端视频视窗
 * @param remoteDisplay 远端视频显示控件
 */
public void setRemoteDisplay(RTCVideoView remoteDisplay)
```

直播创建与销毁

直播创建与销毁接口与调用流程与AR直播相同，此处略。 同时，退出互动直播时需要销毁互动通道：

```
 /**
 * 销毁互动连麦通道
 */
public void releaseInteractSession()
```

1.4 互动通道建立与销毁 建立多媒体互动

当主播端接收到连麦互动请求后调用如下接口建立连麦。

```
 //建立连麦互动
public void setupInteractChannel()
```

注意：若设置为服务转推模式，则需要在建立连麦前，先关闭当前的直播，示例：

```
 // 服务转推模式下先 关闭直播通道
if (mPushMode == Constraints.MODE_LIVE_INTERACT_TRANSFORM) {
    mInteractSession.stopStreaming();
    mInteractSession.destroyRtmpSession();
    mIsLiveConnected = false;
}
// 建立连麦互动通道
mInteractSession.setupInteractChannel();
```

退出多媒体互动

当主播主动或被动挂断连麦互动时调用如下接口退出互动连接。

```
 // 退出连麦互动
public void destroyInteractChannel()
```

注意:同样的，若设置为服务转推模式，则需要在退出连麦后，再开启直播，示例：

```
 // 销毁连麦互动通道
mInteractSession.destroyInteractChannel();
// 服务转推模式下恢复直播通道
if (mPushMode == Constraints.MODE_LIVE_INTERACT_TRANSFORM) {
    mInteractSession.configRtmpSession(mPushUrl);
}
```

1.5 其它互动接口 设置本地混流布局模板

```
 /**
 * 通用混流布局模板初始化配置
 * @param width 视窗宽度
 * @param height 视窗高度
 * @param positionX 视窗X位标相对比例 [0, 1], 对等布局模式下positionX 无效
 * @param positionY 视窗Y位标相对比例 [0, 1]
 * Note: 若为画中画{@link MixTemplate#PIC_IN_PIC} 布局，则视窗表示远端视频小窗
 *       若为对 等{@link MixTemplate#SIDE_BY_SIDE_H} 布局，则视窗表示本地与远端组合后的视频小窗
 */
public MixStreamingConfig(MixTemplate template,  int width, int height, float positionX, loat positionY)

/**
 * 混流布局模板初始化配置
 * NOTE: 该构造函数主要用于画中画模板, 可直接设置小窗位置到： 左上、右上、左下、右下
 * @param width 视窗宽度
 * @param height 视窗高度
 * @param location 位置  {@link LOCATION} [LEFT_TOP,RIGHT_TOP,LEFT_BOTTOM,RIGHT_BOTTOM]
 */
public MixStreamingConfig(MixTemplate template, int width, int height, LOCATION location)
/**
 * 设置本地混流布局模板
 * @param mixStreamingConfig 模板配置
 */
public void setMixStreamingConfig(MixStreamingConfig mixStreamingConfig)
```

设置左上角画中画布局示例：

```
 MixStreamingConfig mixStreamingConfig =
        new MixStreamingConfig(MixStreamingConfig.MixTemplate.PIC_IN_PIC, 180, 320, MixStreamingConfig.LOCATION.LEFT_TOP);
mInteractSession.setMixStreamingConfig(mixStreamingConfig);
```

设置任意垂直位置及视频高度对等布局示例：

```
 MixStreamingConfig mixStreamingConfig =
        new MixStreamingConfig(MixStreamingConfig.MixTemplate.SIDE_BY_SIDE_H
, 0, 240, 0.0f, 0.5f);
mInteractSession.setMixStreamingConfig(mixStreamingConfig);
```

互动连麦本端静音

连麦时本端静音，区别与setMuteAudio(boolean isMute) 推流静音。

```
 /**
 * 连麦静音
 * @param muted 是否静音
 */
public void interactMute(boolean muted)
```

预设开启扬声器

连麦时默认开启扬声器， 可通过该接品预设是否开启扬声器输出音频。

```
 /**
 * 预设开启扬声器
 * @param enable 是否预设扬声器
 */
public void presetLoudSpeaker(boolean enable)
```

切换互动连麦扬声器

```
 /**
 * 预设开启扬声器
 * @param enable 是否预设扬声器
 */
public void presetLoudSpeaker(boolean enable)
```

连麦异常重连

```
 /**
 * 连麦异常重连 默认重连间隔3s
 */
public void interactReconnect()

/**
 * 连麦异常重连
 * @param interval 重连间隔 不小于3s
 */
public void interactReconnect(int interval)
```

注意：由于涉及互动模块内部及服务端状态的恢复， 重连间隔建议不小于3s.

设置互动直播水印

设置互动直播水印方法与AR直播一样，不同的，互动直播时小主播端视频及观众端视频可同时展示水印。

特效直播

ARMediaStreamingPusher类控制AR直播推流的执行。内部封装这一套完整的音视频编码，打包，上传等功能。
其中AR直播管理类包含通用推流接口、及AR控制接口两部分。

1.1 AR直播实现流程 整个推流流程涉及的调用包括：AR推流器初始化、直播创建与销毁、直播状态控制、拍摄器与AR设置、及推流体验优化及状态访问五部分。

AR 直播关键步骤示例，具体可参考demo工程AnchorARActivity.java文件。

```
 // 初始化AR直播推流器
mSession = new ARMediaStreamingPusher(this, mGlSurfaceView, liveConfig);
// 设置直播事件监听
mSession.setLiveEventListener(this);

// 初始化音视频采集设备
mSession.setupDevice(mOnCaptureReadyCallback);
// 配置推流地址、创建推流通道
mSession.configRtmpSession(mPushUrl);
// 在拍摄器就绪(onVideoCaptureReady)且推流通道连接后开始推流
public void onSessionConnected() {
    ... ...
if (mIsVideoCaptureReady && mIsLiveConnected) {
        // 开始推流
       mSession.startStreaming();
  }
}
    ... ...
// 退出直播 释放直播通道
// 停止推流
mSession.stopStreaming();
// 销毁推流通道
mSession.destroyRtmpSession();
// 释放音视频采集设备
mSession.releaseDevice();
```

1.2 AR推流器初始化 配置推流参数

```
 LiveConfig.Builder builder = new LiveConfig.Builder();
builder.setCameraId(Camera.CameraInfo.CAMERA_FACING_FRONT) // 设置前置相机
       .setCameraOrientation(90) // 设置相机旋转角度
       .setVideoWidth(mWidth) // 设置视频采集与输出宽度（像素）
       .setVideoHeight(mHeight) // 设置视频采集与输出高度（像素）
       .setOrientation(mOrientation) // 设置屏幕方向
       .setVideoFPS(mFps) // 设置视频采集与输出帧率
       .setInitVideoBitrate(mBitrate) // 设置视频码率初始比特率
       .setQosEnabled(mEnableAutoBitrate) // 设置是否开启码率自适应
       .setMaxVideoBitrate(mMaxBitrate) // 设置码率自适应最大比特率
       .setMinVideoBitrate(mMinBitrate) // 设置码率自适应最小比特率
       .setVideoEnabled(true) // 设置是否推视频流
       .setAudioEnabled(true) // 设置是否推音频流
       .setAudioSampleRate(44100) // 设置音频采样率
       .setAudioBitrate(64000); // 设置音频比特率

builder.setPicStreamingEnabled(true); // 使能后台推流
if (mOrientation == LiveConfig.PORTRAIT) {
    builder.setPausePicPath("pause_streaming_portrait"); // 设置竖向暂停图片
} else {
    builder.setPausePicPath("pause_streaming_land"); // 设置横向暂停图片
}
```

初始化AR直播推流器

```
 /**
 * AR媒体推流器构造函数
 * @param context 上下文环境
 * @param surfaceView 本地预览视窗
 * @param liveConfig 推流配置
 */
public ARMediaStreamingPusher(Context context, GLSurfaceView surfaceView, LiveConfig liveConfig)
```

注：AR 推流器使用GLSurfaceView 作用本地预览控件

设置直播事件监听

```
 // 直播事件监听
public interface OnLiveEventListener {
    /**
     * 直播通道已经建立
     */
    void onSessionConnected();
    /**
     * 直播过程中出错
      *
     * @param errorCode 错误码
     */
    void onError(int errorCode);
}

// 错误码定义：
public interface BidirectRtmpEventListener {
    int ErrorCodeBase = -10000;
    //与服务器建立rtmp连接过程出错
    int ErrorCodeConnectToServerFailed = -10000;
// 创建rtmp推流通道出错
    int ErrorCodePulishStreamFailed = -20000;
    //Disconnect过程中出错
    int ErrorCodeDisconnectFromServerFailed = -30000;
    // 推流过程中，遇到未知错误导致推流失败
    int ErrorCodeUnknownStreamingError = -60000;
    /**
     * 推流过程中，遇到弱网情况导致推流失败
     * 收到此错误后，建议提示用户当前网络不稳定，
     * 如果反复收到此错误码，建议调用停止推流
     */
    int ErrorCodeWeakConnection = -70000;
    // Try again 重试
    int ErrorCodeWeakConnection_EAGAIN = -70011;
    // No buffer space available 没有可用的缓存空间
    int ErrorCodeWeakConnection_ENOBUFS = -70105;
    // Interrupted system call 中断的系统调用
    int ErrorCodeWeakConnection_EINTR = -70004;
    // Connection timed out 连接超时
    int ErrorCodeWeakConnection_ETIMEDOUT = -70110;
    /**
     * 推流过程中，遇到服务器网络错误导致推流失败
     * 收到此错误后，建议调用立即停止推流，并在服务恢复后再重新推流
     */
    int ErrorCodeServerNetworkError = -80000;
    // Connection reset by peer 连接被对方复位
    int ErrorCodeServerNetworkError_ECONNRESET = -80104;
    /**
     * 推流过程中，遇到设备断网导致推流失败，
     * 收到此错误后，建议提示用户检查网络连接，然后立即停止推流
     */
    int ErrorCodeLocalNetworkError = -90000;
    // Broken pipe 管道破裂
    int ErrorCodeLocalNetworkError_EPIPE = -90032;
    // Bad file number 错误文件编号
    int ErrorCodeLocalNetworkError_EBADF = -90009;
    // Network is down 网络已关闭
    int ErrorCodeLocalNetworkError_ENETDOWN = -90100;
    // Network is unreachable 网络不可达
    int ErrorCodeLocalNetworkError_ENETUNREACH = -90101;
    // License 失效
    int ErrorCodeLicenseInvalid = -100000;
                ... ...
}

// 设置直播事件监听
public void setLiveEventListener(OnLiveEventListener listener)
```

1.3 直播创建与销毁 初始化音视频采集设备

```
 /**
 * 拍摄器初始化回调
 */
public interface OnCaptureStateListener {
    //拍摄头尺寸变化
    void onCameraSizeChange(int width, int height);

   // 摄像头开启状态
    void onCameraOpenResult(boolean var1);

   // 拍摄器就绪回调接口
    void onVideoCaptureReady(boolean success);
}

// 初始化音视频采集设备
public void setupDevice(VideoCaptureSession.OnCaptureStateListener callback)
```

配置推流地址、创建推流通道

```
 /**
 * 创建rtmp推流通道
 * @param pushUrl 推流地址
 * @return 推流通道创建是否成功
 */
public boolean configRtmpSession(String pushUrl)
```

销毁直播通道

```
 // 销毁推流通道
public void destroyRtmpSession()
```

释放媒体采集设备

```
 // 释放音视频采集设备
public void releaseDevice()
```

1.4 直播状态控制 开始推流

```
 // 启动音视频编码 开始推流
public void startStreaming()
```

暂停推流

```
 // 暂停推流, 如配置使能后台推流则开启图片推流否则暂停推流
public void pauseStreaming()
```

注：如使能了后台推流，则调用该接口进行图片推流，否则暂停音视频推流。

恢复推流

```
 //恢复推流
public void resumeStreaming()
```

注：若如使能后台推流则停止图片推流恢复正常推流

结束推流

```
 //关闭音视频编码器、结束推流
public void stopStreaming()
```

1.5 推流体验优化及其它 推流重连

```
 // 推流重连
public void sessionReconnect()

示例： 在推流事件监听错误回调里根据错误类型发起推流重连。
public void onError(int errorCode) {
            ... ...
    // 非直播授权失效异常下发起外部重连
    if (errorCode != BidirectRtmpEventListener.ErrorCodeLicenseInvalid) {
        mSession.sessionReconnect();
    }
}
```

配置推流水印

```
 WaterMarkConfig waterMarkConfig = new WaterMarkConfig("baidu_watermark", 100, 100, 0.04f, 0.98f);
mSession.setWaterMarkConfig(waterMarkConfig);
```

背景音乐

推流状态开启/关闭背景音乐,相关接口：

```
 /**
 * 开始播放背景音乐
 * @param bgmPath 背景音乐本地路径
 * @param isLooping 是否循环播放
 */
public void startBGM(String bgmPath, boolean isLooping)

/**
 * 选择背景音乐区间, 默认从0 播放整首音乐
 *
 * @param clipStartInUSec    微秒
 * @param clipDurationInUSec 微秒
 */
private void configBGMClip(long clipStartInUSec, long clipDurationInUSec)

/**
 * 设置本地mic采集音量的大小，默认是 1f
 *
 * @param gain 范围在 0f 到 1f 之间
 */
public void setRecordTrackGain(float gain)

/**
 * 设置背景音量的大小，默认是 1f
 *
 * @param gain 范围在 0f 到 1f 之间
 */
public void setBGMTrackGain(float gain)

// 暂停播放背景音乐
public void pauseBGM()

// 恢复背景音乐播放
public void resumeBGM()

// 停止播放背景音乐
public void stopBGM()
```

音视频源外部采集

直播SDK支持从外部导入音视频数据源，使用外部采集首先须使能音视频外部采集，然后分别使用如下接口开启视频外部采集、更新外部视频纹理（当前仅支持视频纹理输入方式）、更新音频数据（pcm 格式）。

```
 /**
 * 开始外部视频采集, 有外部渲染控制
 * @param eglContext 外部视频渲染EglContext, sdk内部依赖eglconect 获理视频纹理
* @note 该接口一般在外部渲染器创建成功回调函数里调用
 */
@Override
public void startExtVideoDevice(EGLContext eglContext)

/**
 * 外部纹理更新导入接口
 * @param textureId 外部视频纹理id
 * @param width 预览视频宽度
 * @param height 预览视频高度
 */
public void onExtTextureUpdate(int textureId, int width, int height)

/**
 * 外部音频pcm 数据导入接口
 * @param bufferData 音频pcm数据
 * @param bufferInfo 音频数据信息
 */
public void onExtAudioFrameUpdate(ByteBuffer bufferData, MediaCodec.BufferInfo bufferInfo)
```

媒体采集数据回调

推流器支持将内部采集的音频数据（pcm）及视频数据（yuv nv21 / textureId）对外输出。

静音推流

推流过程关闭/开启音频流

```
 /**
 * 是否静音推流
 * @param isMute 是否静音
 */
public void setMuteAudio(boolean isMute)
```

获取推流帧率

```
 // 获取视频上传帧率
public double getUploadFps()
```

获取推流码率

```
 // 获取媒体发送码率
public double getUploadBindwidthInKBps()
```

获取推流缓冲区水位

```
 // 获取推流缓冲区百分比
public double getPacketCacheRate()
```

1.6 推流器（协议）选择 Rmtp推流器：ARMediaStreamingPusher

SRT推流器：SrtMediaStreamingPusher

SRT + AR推流器：ARSrtMediaStreamingPusher

超低延时直播

概述 百度云超低延时直播（BRTCPlayer）是在百度智能云视频直播的基础上通过链路传输协议优化，解决传统直播延迟过高的弊端，通过集成百度智能云播放器SDK，可以支持千万级并发场景的毫秒级延时直播，并为用户提供低卡顿、秒开流畅的直播观看体验。

功能介绍 BRTCPlayer 主要功能如下：

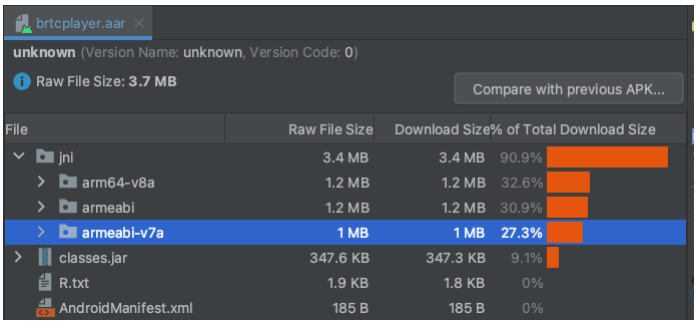
- 支持UDP信令模式：除HTTP标准信令模式外支持UDP信令模式，0 RTT 极速启播，有效提升秒开率；
- 支持AVC、HEVC视频硬解码；
- 支持视频B帧：引入B帧，可在清晰率不变的情况下降低码率；
- 支持AAC音频解码，支持MP4A-ADTS格式 48000 及 44100 采样率；
- 支持视频平滑渲染： 网络抖动时，平滑渲染视频帧, 降低卡顿；
- 支持最小延迟动态配置：可实现延迟换流畅，在允许范围内增大延迟可有效降低卡顿；
- 支持网络健康度实时检测；
- 内置HTTPDns支持；
- 支持 SEI 消息；

Demo体验 前往DEMO下载页面扫码安装DEMO

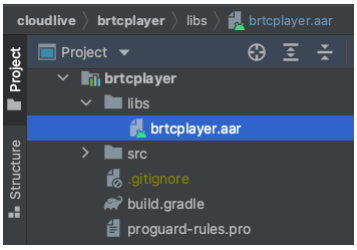
SDK 下载 前往SDK下载页面下载支持超低延时直播的智能视频SDK V1.8.0 及以后版本， 内含Demo源码；

快速集成 百度云超低延时直播由独立SDK提供支持，用户可通过在播放侧单独集成BRTCPlayer SDK实现超低延时直播，而推流侧则可复用当前使用的推流器，仅需在百度直播LSS平台针对当前域名开通低延时播放能力。 BRTCPlayer SDK具体的集成步骤如下：

- BRTCPlayer 作为独立SDK 提供低延时播放能力，包体精简，classes.jar 约350KB, 单架构 so （如 armeabi-v7a） 1M；



- 导入BRTCPlayer aar到工程，集成后的示例如下：



- 修改 build.gradle

```
 dependencies {
    implementation fileTree(dir: 'libs', include: ['brtcplayer.aar'])
    implementation 'com.squareup.okhttp3:okhttp:3.10.0'
    ...
}
```

- 添加混淆规则

```
 -keepclasseswithmembernames,allowshrinking,allowoptimization class * {
    native <methods>;
}

-keep interface * {
    public <fields>;
    public <methods>;
}

-keep class com.baidu.rtc.** {*;}
-keep class com.webrtc.** {*;}
```

快速开始

1. 创建播放渲染视窗

```
 // 在布局文件中添加渲染控件
<com.baidu.rtc.RTCVideoView
        android:id="@+id/brtc_video_view"
    android:layout_width="match_parent"
        android:layout_height="match_parent" />

// 寻回渲染控件对象
mVideoView = (RTCVideoView) findViewById(R.id.brtc_video_view);
```

2. 创建播放器对象及播放器初始化

```
 // 创建播放器对象
mBRTCPlayer = new BRTCPlayerImpl(this);
// 创建播放参数集，初始化播放器
mPlayerParameters = new BRTCPlayerParameters();
// 设置播放器信令服务地址
mPlayerParameters.setPullUrl(mSignalUrl);
// UDP 信令模式需要进行相关预设
if (bUseUdpSignaling) {
    // 设置Udp信令模式
    mPlayerParameters.setEnableSignalOverUdp(bUseUdpSignaling);
    if (bEnableHevc) {
        // 由业务侧确定是否为H265格式,默认为H264格式； udp 信令模式需设置视频编码
        mPlayerParameters.setVideoDecodeFormat(BRTCPlayerParameters.H265);
    }
    // 音频默认格式为 MP4A-ADTS, udp 信令模式需设置采样率
    mPlayerParameters.setAudioSampleRate(mAudioSampleRate);
}
// 初始化播放器 传入播放参数及事件监听器
mBRTCPlayer.initPlayer(mPlayerParameters, this);
```

3. 播放设置

```
 // 设置播放渲染视图
mBRTCPlayer.setSurfaceView(mVideoView);
// 设置媒体流地址
mBRTCPlayer.setStreamUrl(mStreamUrl);
```

4. 播放控制

```
 // 准备播放，获取播放依赖资源
mBRTCPlayer.prepareAsync();
// 开始播放
mBRTCPlayer.startPlay();
// 暂停播放
mBRTCPlayer.pausePlay();
// 恢复播放
mBRTCPlayer.resumePlay();
// 停止播放
mBRTCPlayer.stopPlay();
```

5. 释放播放器

```
 // 释放播放器
mBRTCPlayer.releasePlayer();
```

在智能视频SDK Demo中对上述流程有详细的展示，可以参考；

接口说明 BRTCPlayerParameters类 | 接口名 | 描述 | |----------|----------| | void setEnableDebug(boolean enable) | 设置是否开启debug日志 | | void setPullUrl(String pullUrl) | 设置信令服务地址，格式为http[:s]//domain/brtc/v3/pullstream | | void setAutoPlay(boolean autoPlay) | 设置是否自动启播，默认false | | | void void setEnableSilentStart(boolean enable) | 设置静音启播 | | void setBaseDelayMs(int baseDelayMs) | 设置最小延时， 可通过该参数增大延时降低卡顿提升播放流畅度 | | void setCpuType(String cpuType) | 指定下载的SO库CPU架构 | | void setVideoDecodeFormat(int videoDecodeFormat) | 设置视频解码格式，由业务侧确定是否为H265格式，默认为H264格式，仅UDP信令模式 | | void setAudioSampleRate(int sampleRate) | 设置音频播放采样率[48000, 44100]， 仅UDP信令模式
| void setEnableVideoBFrame(boolean bFrame) | 是否开启B帧支持，由业务侧确定是否开启，默认开启 ，仅UDP信令模式 |

BRTCPlayerEvents类 | 错误码 | 含义 | |----------|----------| | BRTC_PLAYER_ERROR_INVALID_URL = 10000 | URL 格式错误 | | BRTC_PLAYER_ERROR_ICE_CHANNEL = 10001 | ICE 连接错误 | | BRTC_PLAYER_ERROR_RESERVED = 10002 | 替换预留错误码 | | BRTC_PLAYER_ERROR_CONNECTION = 10003 | 连接创建失败 | | BRTC_PLAYER_ERROR_LOCAL_SDP_REQUEST = 10004 | 媒体描述请求失败 | | BRTC_PLAYER_ERROR_LOCAL_SDP_SET = 10005 | 媒体描述设置失败 | | BRTC_PLAYER_ERROR_REMOTE_SDP_REQUEST = 10006 | 远端媒体描述请求失败,一般由于启播放时主播侧已停播 | | BRTC_PLAYER_ERROR_REMOTE_SDP_SET = 10007 | 远端媒体描述设置失败 | | BRTC_PLAYER_ERROR_INVALID_STATE = 10008 | 播放状态错误 | | BRTC_PLAYER_ERROR_STREAMING_INTERRUPT = 10009 | 媒体流中断，一段时间未收媒体流，SDK内部检测到断流错误后会立即停止播放，一般在播放中主播停播但业务侧未退出 | | BRTC_PLAYER_ERROR_LOAD_LIBRARIES = 10010 | SO库文件后下载失败 | | BRTC_PLAYER_ERROR_INVALID_LICENSE = 100011 | license错误 | | BRTC_PLAYER_ERROR_LICENSE_FEATURE_INVALID = 10012 | 当前license不包含低延时播放feature | | BRTC_PLAYER_ERROR_DECODER_OPEN_FAILED = 10013 | 解码器打开失败 | | int BRTC_PLAYER_ERROR_MTU_SIZE_CHECK_FAILED = 10014 | 用户网络MTU Size 检测失败（< 1300 Byte）无法支持RTC媒体传输，约2s后返回,建议业务侧降级为普通播放 |

事件码	**含义**
BRTC_PLAYER_EVENT_REMOTE_RENDER = 1000	开始远端渲染
BRTC_PLAYER_EVENT_ICE_CONNECTED = 1001	ICE连接成功
BRTC_PLAYER_EVENT_PEER_CONNECTION_CLOSED = 1002	对端连接关闭
BRTC_PLAYER_EVENT_STATS_UPDATED = 1003	媒体流信息更新
BRTC_PLAYER_EVENT_BUFFERING_START = 1004	Buffering start事件
BRTC_PLAYER_EVENT_BUFFERING_END = 1005	Buffering end事件
BRTC_PLAYER_EVENT_ICE_DISCONNECTED = 1006	ICE连接断开
BRTC_PLAYER_EVENT_NO_STREAMING_DETECTED = 1007	没有检测到媒体流
BRTC_PLAYER_EVENT_PLAY_TIME_STATISTIC = 1008 BRTC_PLAYER_EVENT_LOCAL_SDP_SET = 1009 BRTC_PLAYER_EVENT_REMOTE_SDP_ACQUIRED = 1010 BRTC_PLAYER_EVENT_LIBS_DOWNLOAD_COMPLETED = 1011 BRTC_PLAYER_EVENT_LIBS_LOADED_SUCCESS = 1012 BRTC_PLAYER_EVENT_RENDERVIEW_VISIBLE = 1013	启播各阶段耗时统计
BRTC_PLAYER_EVENT_DECODER_OPENED = 1014	解码器成功打开
BRTC_PLAYER_EVENT_START_RECONNECT = 1015	内部播放重连
BRTC_PLAYER_EVENT_LINK_DOWN = 1016	播放链路带宽不足，建议业务侧尝试低码率流播放
BRTC_PLAYER_EVENT_FIRST_AUDIO_PLAYOUT = 1007	音频首帧播放出事件, 若业务侧设置初始静音播放，则可在该事件恢复音频播放

事件回调	**含义**
void onPrepared()	播放器准备就绪
void onFirstFrameRendered()	首帧渲染事件
void onResolutionChanged(int w, int h)	分辨率更新回调
void onError(int errCode, String msg)	错误错误回调，错误码定义见前文
void onInfoUpdated(int event, Object msg)	事件更新回调，事件码定义见前文
void onPlayerStateChanged(BRTCPlayer.PlayerState currentState)	播放状态更新
void onSEIRecv(ByteBuffer data)	接收到SEI 消息，回调在解码线程,不应在该回调里实现复杂业务

> SEI 数据格式(h264):[nal_type(1)|sei_type(1)|sei_size(n+1)|sei_payload(n*255+m)|trailing_bits(1)]
> sei_size: 需判断第3个字节起连续0xFF的个数n，加第1个不为0xFF的字节(如: 0x21)，该字段占字节数为 n + 1; 其值为:0xFF*n+0x21;
> sei_payload: SEI字段的值（h265）；如:[0x06 0x05 0x18 0x54 0x80 0x83 0x97 0xF0 0x23 0x47 0x4B 0xB7 0xF7 0x4F 0x32 0xB5 0x4E 0x06 0xAC 0x27 0x11 0xFE 0x69 0x79 0x01 0x00 0x00 0x80]
> 注： h265 与 h264 的 sei 格式略有不同， h264 的nal_type 为 0x06, sei_type 为0x 05; h265 的 nal_type 为 0x4e, sei_type 为两字节 0x01、0x05;

RTCVideoView类 | 枚举类型 | 描述 | |----------|--------| | ScalingType.SCALE_ASPECT_FIT | 缩放模式：适应 | | ScalingType.SCALE_ASPECT_FILL | 缩放模式：拉伸 |

BRTCPlayer接口 | 接口名 | 描述 | |----------|----------| | long initPlayer(BRTCPlayerParameters playParameters, BRTCPlayerEvents events) | 初始化播放器。
playParameters 播放参数集

events 播放事件监听 | | void setEventObserver(BRTCPlayerEvents events) | 设置播放事件监听 | | void setSurfaceView(RTCVideoView surfaceView) | 设置播放渲染视窗,当前只支持RTCVideoView | | void setScalingType(RTCVideoView.ScalingType scaleType) | 设置播放视窗缩放模式 | | void prepareAsync() | 准备播放 | | void setPlayWhenReady(boolean autoPlay) | 设置是否自动启播，默认false | | void startPlay() | 开始播放 | | void pausePlay() | 暂停播放,不停止拉流，仅暂停本地媒体流渲染 | | void resumePlay() | 恢复播放,从暂停状态恢复播放 | | void stopPlay() | 停止播放，停止媒体拉流及渲染，再次播放需调用startPlay() | | boolean hasVideo() | 媒体流是否包含视频流 | | boolean hasAudio() | 媒体流是否包含音频流 | | void releasePlayer() | 释放播放器 | | void setVolume(double volume) | 设备播放音量 | | void setStreamUri(String streamPath) | 设置播放媒体流地址，媒体流URL格式为webrtc://domain/app/stream | | PlayerState getPlayerState() | 获取播放状态 |

SDK体验

SDK下载

直播SDK

- 开发者：北京百度网讯科技有限公司
- 版本号：见SDK下载列表
- 主要功能：提供包括美颜美颜、连麦互动、超低延时、高质量的直播推流服务
- 个人信息处理规则： 见 百度直播SDK隐私政策
- 合规使用说明：见 百度直播SDK开发者个人信息保护合规指引

直播SDK中集成了AR特效的能力，您可以选择下载您申请的license中的AR版本对应的SDK版本

Android直播SDK下载列表

SDK信息	集成AR版本	更新说明	更新时间	SDK下载
版本号： V1.8.0 包名：com.baidu.cloudvideo MD5：3f5767a371c77954b12d7a295ef51e18	V5.1	1.新增BRTCPlayer超低延时直播；	2023-07-10	下载
版本号：V1.7.1 包名：com.baidu.cloudvideo MD5：50502fc6b6d1c873dd0a5fb4d6a2d6d6	V5.1	1.解决部分Android 12机型推流异常； 2.修复若干Bug;	2022-09-21	下载

IOS直播SDK下载列表

SDK版本	集成AR版本	更新说明	更新时间	SDK下载
版本号：V1.8.0 MD5：66cad763942258c7aa578b5bfae29c6f	V5.1	1.新增BRTCPlayer超低延时直播；	2023-07-10	下载

Demo体验

1 Demo下载

直播APP 我们提供直播demo APP供您体验直播能力，您可扫描以下二维码下载安装体验。
下载demo需要密码，密码为：bdcloud

- **Android**

- **iOS**

2 AR特效

（1）互动特效case-人脸特效

- 多人脸贴纸特效

（2）互动特效case-手势互动（7款手势）

- 手势控雨特效

（3）互动特效case-肢体互动

- 肢体互动特效-雷电

（4）互动特效case-身体美化

- 头发染色特效

（5）互动特效case-环境特效

- 手势切换粒子+天空顶特效

（6）互动特效case-SLAM放置

- 在贴纸中支持，放置人物，动物，产品等3D模型

常见问题

常见问题总览

开发类问题

- 开发环境要求
- 提示license过期
- 美颜、美妆、贴纸等无效果
- 需要录音权限
- 本地添加贴纸

内容制作类

- 怎样制作特效贴纸？
- 有音乐资源吗？

常见错误码

以下为集成SDK中可能遇到的常见错误码

内容制作类问题

怎样制作特效贴纸？

可提供适用于SDK的特效贴纸制作工具（支持人脸、手势、人像分割等）。工具面向设计师，采用可视化的界面、操作成本比较低，设计师即可独立完成内容的开发制作，极大的节省了技术开发资源。
如无设计团队，可采购我们的贴纸素材库，贴纸库里已有上百款内容。

有音乐资源吗？

接入了太合版权库，有20万首版权音乐可供选择

常见错误码

常见错误码

1、错误码BDCloudAVAuthFailure = 20000，可能是因为licenseID与.license不对应 ,请去官网比对
2、错误码BDCloudAVAuthLicenseMiss = 20004，项目中没有找到.license文件
3、错误码BDCloudAVAuthLicenseExpired = 20006，可能是licenseId配置成了包名，请区分licenseId与包名不同
4、错误码BDCloudAVAuthLicenseStickerNoPermission = 20007，没有获取对应贴纸的权限，请重新编辑服务请求，选取对应的贴纸
5、错误码BDCloudAVAuthControlFailure = 20001，
BDCloudAVAuthSessionTokenFailure = 20002，
BDCloudAVAuthResponseNone = 20003，
BDCloudAVAuthLicenseSignFailure = 20005，
BDCloudAVAuthParamError = 20008，
可能与请求参数配置有关，可以联系我们

开发类问题

开发环境要求

1、Android Studio 3.2或以上版本，Gradle 4.6或以上版本。编译环境请选择支持java8， Android 4.4系统以上，API Level 19以上；
2、iOS9及以上系统，Xcode 9.0+

提示license过期

1、app的build.gradle中的ApplicationId需要和申请license时的包名一致
2、和包名匹配的license文件以及和包名匹配的 licenseID
注意：三者不能填错（须自查）；开发前请阅读demo中ReadMe文件

美颜、美妆、贴纸等无效果

请查看license是否设置正确，具体请参考工程搭建文档。
如果设置正确，请查看设置美颜、美妆、贴纸的时机，是否在onSetup方法回调后，具体请参考上方美颜、美妆、贴纸设置文档。

添加本地贴纸

若需要使用自定义本地贴纸，需要把资源放到工程项目Sources/LocalStickers文件夹中，同一个贴纸资源命名一致（包括贴纸图片，贴纸资源，贴纸模型资源）。

需要录音权限

请在授予相机权限，麦克风权限，存储空间权限后再录制。

相关协议

移动直播SDK开发者个人信息保护合规指引

亲爱的开发者：

感谢您在所开发的移动应用中集成并使用百度旗下软件开发工具包（SDK）。

百度非常重视用户个人信息保护，包括集成百度旗下直播推流软件开发工具包（SDK）（以下简称“百度SDK”）的移动应用的最终用户（以下简称“最终用户”）个人信息保护，特制定《移动直播SDK个人信息保护合规开发者指引》，以供您在所开发的移动应用（以下简称“移动应用”）中集成并使用百度SDK时参照执行。

一、 个人信息保护合规基本要求

在所开发的移动应用中集成并使用百度SDK，您需要首先遵守以下个人信息保护合规基本要求：

1. 您应遵守收集、使用最终用户个人信息有关的所有可适用法律法规及规范性文件要求，包括但不限于《个人信息保护法》、《网络安全法》、《App违法违规收集使用个人信息行为认定方法》、《工业和信息化部关于开展APP侵害用户权益专项整治工作的通知》（工信部信管函［2019］337号）、《工业和信息化部关于开展纵深推进APP侵害用户权益专项整治行动的通知》（工信部信管函〔2020〕164号）、《工业和信息化部关于进一步提升移动互联网应用服务能力的通知》（工信部信管函〔2023〕26号）等，保护用户个人信息安全。
2. 您的APP需要制定一份独立的隐私政策。隐私政策的内容建议通俗易懂，对您的APP收集、使用个人信息的目的、方式、范围，清晰、完整、准确地向个人信息主体进行明示告知，并充分给予最终用户独立的选择权，确保在获得个人信息主体授权同意后方可进行个人信息的处理活动。《隐私政策》应由用户自主选择是否同意，不应以默认勾选同意的方式或是以欺骗诱导的方式取得用户授权。但请您注意，仅是改善服务质量、提升用户体验、定向推送信息、研发新产品还不足以能成为要求用户同意收集其个人信息的理由。
3. 您应将在移动应用中集成并使用百度SDK服务的情况，以及百度SDK对最终用户必要个人信息的收集、使用和保护规则（具体请见移动直播SDK隐私政策），在移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（具体请参考本指引第二部分 “如何告知最终用户”及第三部分“告知文案示例”），并获得最终用户对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。如果最终用户是未满14周岁的未成年人，请您务必确保获得最终用户的父母或其他监护人对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。
4. 您应向最终用户提供易于操作的访问、更正、删除其个人信息，撤销或更改其授权同意、注销其个人账号等用户权利实现机制。
5. 您应确保在移动应用首次运行时，应在最终用户阅读并同意移动应用隐私政策之后，方可初始化百度SDK进行最终用户信息采集。
6. 百度SDK会根据产品升级优化、提升安全性能、法律及监管要求等原因，不断升级迭代SDK版本，不同版本的SDK获取的字段信息可能会有差异。为了保证合法合规开展合作，并切实履行保护用户个人信息的责任与义务，请您务必确保您已将APP中的百度SDK升级至官方最新版本，以避免因使用旧版本SDK而出现违法违规的问题，导致您的APP遭受监管处罚的风险。百度SDK更新后会及时通过官网通知、站内信、公告、邮件、短信等有效方式对您进行告知，请您及时关注并尽快更新。

除了上述个人信息保护合规基本要求外，您还应遵守您所开发的移动应用所集成并使用的移动直播SDK隐私政策。

SDK基本业务功能与拓展业务功能：

移动直播SDK基本业务功能为提供直播推流、基础拍摄、连麦互动、水印、数据统计、质量监控。移动直播SDK扩展业务功能为拉流播放、视频混流、美颜、背景音乐、背景分割、媒体审核等。

以下是移动直播SDK获取您的APP的最终用户的个人信息以及权限，由于不同SDK版本采集的字段与是否可选可能存在一定差异，具体采集情况以实际接入的SDK版本为准：

SDK收集个人信息情况：

功能及服务	个人信息类型	个人信息字段（区分必选信息和可选信息）
区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌	必选
区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号	必选
区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本	必选
区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息	必选
区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况	必选
区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息	必选
区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率	必选

安卓操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集视频数据，进行直播推流	直播推流时
录音	开启麦克风	用于采集音频数据，进行直播推流	直播推流时
录屏	录屏功能	用于采集屏幕数据，共享屏幕给远端	共享屏幕时

IOS操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相机	开启摄像头	用于采集视频数据，进行直播推流	直播推流时
录音	开启麦克风	用于采集视频数据，进行直播推流	直播推流时
录屏	录屏功能	用于采集屏幕数据，共享本地屏幕给远端	共享屏幕时

SDK初始化设置：

请务必确保您已将移动直播SDK升级到最新版。上述版本调用初始化函数不会采集用户个人信息，也不会向百度联盟后台上报数据。请确保用户同意《隐私政策》后再进行start/autoTrace采集配置，方可采集用户个人信息并上报数据。

调用初始化函数文档： https://cloud.baidu.com/doc/VideoCreatingSDK/s/Ckih8a9o5#12-ar%E6%8E%A8%E6%B5%81%E5%99%A8%E5%88%9D%E5%A7%8B%E5%8C%96

二、 如何告知最终用户

为帮助您明确地告知最终用户百度SDK个人信息收集、使用和保护相关事宜，我们为您提供了以下告知方式，供您参考执行：

1. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并显示相应**百度SDK隐私政策链接**以告知最终用户，百度SDK收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。
2. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并参考相应百度SDK隐私政策内容，**以条款或表格等形式列明**收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。
3. 当最终用户在移动应用中首次打开/使用相应功能/服务时，以**弹窗、页面提示**方式显示相应**百度SDK隐私政策链接**，以告知最终用户相应功能/服务由百度SDK提供，百度SDK为提供相应功能/服务而收集、使用的最终用户个人信息类型、目的及用途，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

三、 告知文案示例

为帮助您明确地告知最终用户百度SDK个人信息保护规则相关事宜，我们为您提供了以下告知方文案示例，供您参考执行：

文案示例A

为向您提供直播推流功能/服务，我们集成了北京百度网讯科技有限公司的移动直播SDK。在为您提供直播推流功能/服务时，北京百度网讯科技有限公司移动直播SDK需要收集、使用您必要的个人信息，包括设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息，用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用直播推流、连麦互动及美颜特效等功能。关于北京百度网讯科技有限公司移动直播SDK收集、使用的个人信息类型、目的及用途，以及移动直播SDK将如何保护所收集、使用的个人信息，请您仔细阅读《移动直播SDK隐私政策》了解。

文案示例B

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	个人信息类型	使用目的	官网链接/隐私政策链接
移动直播SDK	北京百度网讯科技有限公司	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用直播推流、连麦互动及美颜特效等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/Skifpoa0z

文案示例C

为向您提供直播推流功能/服务，我们集成了北京百度网讯科技有限公司的移动直播SDK。在为您提供直播推流功能/服务时，北京百度网讯科技有限公司的移动直播SDK收集、使用您的设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息，用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用直播推流、连麦互动及美颜特效等功能，具体请您仔细阅读《移动直播SDK隐私政策》了解。

文案示例D

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	业务功能	个人信息类型	调用权限类型	具体目的/用途	隐私政策链接
移动直播SDK	北京百度网讯科技有限公司	直播推流	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	相机、录音、录屏、相册	用于通信质量和设备性能检测，对兼容/崩溃问题进行适配和故障排查，保障用户正常使用直播推流、连麦互动及美颜特效等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/Mmcwzji5g

四、 使用SDK服务的合规注意事项

1. 您接入百度SDK前，应当仔细阅读SDK的协议约定、本合规规范、用户协议、隐私政策等内容，并依据相关内容对您APP的《隐私政策》及您APP收集、存储、使用、共享等处理个人信息的情况进行合规自查。
2. 您知悉并认可百度SDK具备收集个人信息的功能，并认可该等信息的收集均为双方合作之必要目的所需。
3. 您承诺已制定并按照相关要求公示您APP的《隐私政策》，并已清晰、明确、显著地说明有关通过SDK收集个人信息的必要性、收集数据的范围、方式以及用途。同时，您应确保在APP首次运行时以弹窗等合规方式显著提示用户阅读您APP的《隐私政策》并取得用户的合法授权同意，经过合法授权后再初始化百度SDK进行个人信息的收集与处理。
4. 您已认真阅读并理解百度SDK平台协议、合作规范、隐私政策、接入文档等约定和要求，并承诺在您使用百度SDK服务期间，针对百度SDK收集、使用、处理、共享、转让相关个人信息，您已取得了用户持续有效的授权和同意，并保证您不会违反国家相关法律法规、相关国家标准以及双方约定的目的。
5. 如果您的APP面向不满十四周岁的儿童及未成年人用户提供服务，您承诺遵守儿童个人信息保护及未成年人保护相关的法律法规，您承诺已采取相关措施并保证已获得其监护人的授权同意。
6. 如因您违反百度SDK的平台协议、合作规范、隐私政策、接入文档等约定，导致您的用户或第三方对百度主张任何形式的索赔或权利要求，或导致百度因此产生任何法律纠纷的，您将负责解决并承担全部责任，如因此给百度及其关联主体造成损失的，您应赔偿因此给百度及其关联主体造成的全部损失。
7. 您保证对于您从百度SDK获取的数据，无论是在合作期间或是合作停止后，均承担保密义务，不擅自提供、泄露、透露给任何第三方，并应采取一切合法措施以使上述数据免于散发、传播、披露、复制、滥用及被无关人员接触，避免导致相关数据被超出双方合作目的使用。

移动直播SDK隐私政策

欢迎使用移动直播软件开发工具包（SDK）（简称"移动直播SDK"）服务！

移动直播SDK 是一款为移动应用开发者（以下简称"开发者"）提供移动端直播推流服务的软件开发工具包（SDK）。开发者在其移动应用内集成移动直播SDK后，可通过移动直播SDK平台向其移动应用（以下简称"开发者应用"）的最终用户（以下简称"最终用户"）提供本隐私政策所说明的功能及服务。开发者在其移动应用集成并使用移动直播SDK服务时，委托移动直播SDK处理开发者应用相关数据信息，其中可能包括开发者应用最终用户（以下简称"最终用户"）的个人信息。此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策"附录1：名词解释"来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用移动直播SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用移动直播SDK服务的情况，以及移动直播SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于移动直播SDK收集、使用最终用户相关个人信息的完整、合法、在使用移动直播SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于移动直播SDK收集、使用最终用户相关个人信息的完整、合法、在使用移动直播SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

（4）关于上述承诺的具体落地执行可参考《移动直播SDK开发者个人信息保护合规指引》。

2、我们希望集成并使用移动直播SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的移动直播SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，移动直播SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用移动直播SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订

8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用移动直播SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌（必选）	采用加密传输的安全处理方式	iOS及Android
2	区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号（必选）	采用加密传输的安全处理方式	iOS及Android
3	区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本（必选）	采用加密传输的安全处理方式	iOS及Android
4	区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息（必选）	采用加密传输的安全处理方式	iOS及Android
5	区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况（必选）	采用加密传输的安全处理方式	iOS及Android
6	区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息（必选）	采用加密传输的安全处理方式	iOS及Android
7	区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率（必选）	采用加密传输的安全处理方式	iOS及Android

2. 设备权限调用

为了保证最终用户能正常使用移动直播SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能，各项功能及功能对设备权限的调用情况如下：

Android系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
录屏	录屏功能，采集屏幕数据	同上

iOS系统版本

设备权限	功能及服务	权限授权方式
相机	开启摄像头，采集视频数据	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启
录音	开启麦克风，采集音频数据	同上
录屏	录屏功能，采集屏幕数据	同上

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用移动直播SDK产品和服务时遇到的问题，改进及优化移动直播SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用移动直播SDK服务时遇到的问题，改进及优化移动直播SDK的服务体验，提高移动直播SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用移动直播SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与移动直播SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器

都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息：

仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；
2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；
3. 如果我们或我们的关联公司涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；
2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；
6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；
8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在移动直播SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。
2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。
3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系，**并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。
4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过移动直播SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加密，注意确保您个人信息的安全。**
5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**

6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案，**由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。
2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。
3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

移动直播SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成移动直播SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及**《百度儿童个人信息保护声明》**，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过工单联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用移动直播SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；
2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；
3. 如果我们停止提供产品或者服务，或者保存期限已届满；
4. 如果最终用户撤回同意；
5. 法律、行政法规规定的其他情形。

当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

移动直播SDK愿一直陪伴开发者和最终用户，若因特殊原因导致移动直播SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过【八、如何联系我们】中的方式与我们取得联系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在移动直播SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用移动直播SDK产品和服务，若开发者继续集成并使用移动直播SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用移动直播SDK服务期间持续有效的授权同意。**若最终用户不同意该等变更，可以停止使用移动直播SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用移动直播SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；
2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；
3. 个人信息共享、转让或公开披露的主要对象发生变化；
4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；
5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；

6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

移动直播SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对移动直播SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过工单反馈开发者及最终用户对移动直播SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台(http://help.baidu.com/personalinformation) 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。 敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行移动直播SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等唯一设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问移动直播SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的**精确位置信息**，也包括通过IP地址或其他网络信息等获取的**粗略地理位置信息**。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。

匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度手机助手及其他百度系产品https://www.baidu.com/more/)等网站、程序、服务、工具及客户端。

关联公司是指移动直播SDK的经营者【北京百度网讯科技有限公司】及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对移动直播SDK的信任和使用！

北京百度网讯科技有限公司

【2023】年【12】月【15】日更新

【2023】年【12】月【15】日生效

播放器SDK

产品简介与下载

百度智能云提供Web、Android、iOS及鸿蒙平台的播放器SDK，为开发者提供简单、便捷的开发接口，帮助开发者在各类终端设备上实现媒体播放功能。

Web端播放器 （cyberplayer）在标准版SDK之外，提供了高级版SDK，功能包含有H.265 编码格式软硬解播放、AV1编码格式软硬解播放、MPEG-TS播放。为用户带来更丰富的音视频体验。

Android、iOS、鸿蒙平台在标准版SDK之外，提供了高级版SDK，包含有全景声（WANOS）音频格式解码与音效处理、HDR多标准视频解码与渲染、超低延时直播、VR视频播放、智能防挡弹幕、投屏、绿幕抠图、端上超分等高级功能，这些高级功能也可以作为独立组件单独集成，为用户带来更丰富的音视频体验。

在Android、iOS平台，我们还提供了基于Unity框架的播放器SDK，帮助开发者在元宇宙、游戏、VR/AR等场景中快速实现高性能且丰富的媒体播放功能。不仅如此，借助全景声技术，我们在Unity框架上还提供了6DoF空间音频能力，为用户带来视觉、听觉上完全的3D体验。

平台类别	Demo体验	下载地址	API参考	帮助文档
Web	web播放器演示	SDK + Demo	控制接口	播放器Web SDK
Android		SDK + Demo	接口速查	播放器Android SDK
iOS		SDK + Demo	接口速查	播放器iOS SDK
Unity	Unity Android端  Unity iOS端 	SDK + Demo	接口速查	播放器Unity SDK

Android、iOS高级版SDK功能简介

全景声（WANOS）音频格式解码与音效处理

全景声（WANOS）是我国完全自主知识产权的音频编码技术，结合多声道和多声音对象编码，可以提供沉浸式的全景听觉体验。在高级版SDK中，我们提供了全景声（WANOS）音频格式的解码播放能力。

不仅如此，我们还基于全景声（WANOS）空间音效处理算法，针对包括扬声器和耳机在内的不同输出设备，提供了原声模式、电影模式、音乐模式、全景环绕模式等音效处理和切换能力，为用户带来更多样的音频玩法。

百度智能云音视频处理（MCP）同时支持全景声（WANOS）音频内容的生产，为用户提供完整的端到端全景声解决方案。

- Android端全景声功能接入
- iOS端全景声功能接入

HDR多标准视频解码与渲染

HDR视频具有高动态范围、宽色域、高位深的特点，可以呈现更细腻真实的视觉体验。在高级版SDK中，我们不仅提供了HDR10和HLG标准的支持，还支持了我国自主知识产权的HDR Vivid标准，其具备动态元数据、色调映射和饱和度调节能力。

除此之外，我们还通过高性能的后处理算法，让HDR视频在不支持HDR显示的低端机型上也能呈现出正确的色彩，让更多用户感受到HDR画面带来的震撼影像体验。

百度智能云音视频处理（MCP）同时支持HDR视频内容的生产，为用户提供完整的端到端HDR解决方案。

- Android端HDR功能接入
- iOS端HDR功能接入

超低延时直播

百度智能云超低延时直播利用百度RTC技术，实现端到端延迟低于1s的直播观看体验，适用于电商直播、秀场直播等对实时互动性有要求的业务场景。

在高级版SDK中，我们不仅提供了超低延时直播的播放端支持，还利用UDP信令方案进一步优化首屏时间，同时支持H264/HEVC视频编码和AAC音频编码，还提供了对B帧的支持。

百度智能云音视频直播（LSS）支持超低延时直播的推流、分发，如您有接入需求，请提交工单或联系您的客户经理。

- Android端超低延时直播功能接入
- iOS端超低延时直播功能接入

VR视频播放

在高级版SDK中，提供了VR全景视频的高性能渲染能力，支持点播、直播流，同时支持基于陀螺仪的视角控制。

- Android端VR功能接入
- iOS端VR功能接入

智能防挡弹幕

利用百度智能云音视频处理（MCP)对视频中的人体、人脸等重要信息进行预先分析并生成蒙版，可以在高级版SDK中实现防挡弹幕效果，保留弹幕互动性的同时不遮挡画面重要内容，提升用户体验。

投屏

在高级版SDK中，提供了DLNA投屏能力，允许用户将手机端的多媒体内容投送到盒子、投影、电视等大屏设备上，并且可以在手机端控制大屏端的媒体播放。

- Android端投屏功能接入
- iOS端投屏功能接入

绿幕抠图

在高级版SDK中，提供了高精度、高性能的绿幕抠图能力，可实现对绿色或其他纯色背景的自动识别和抠像，背景可以实时替换为2D视频画面或虚拟3D场景，适用于电商直播、虚拟主播、元宇宙直播等场景。绿幕抠图功能可以配合播放内核使用，也支持作为独立组件单独使用。

- Android端绿幕抠图功能接入
- iOS端绿幕抠图功能接入

端上超分

在高级版SDK中，提供了客户端超分辨率能力，利用端侧推理实现对低分辨率画面的清晰度提升、噪声和块效应去除，适用于视频播放场景和RTC通话场景。端上超分能力可以配合播放内核使用，也支持作为独立组件单独使用。

- Android端超分功能接入
- iOS端超分功能接入

Web高级版SDK功能简介

H.265 编码格式软硬解播放

播放器SDK支持H.265/HEVC 编码格式通过软硬解进行播放。优先使用硬解播放，不支持H.265硬解的浏览器，内部优先开启软解模式。

- H.265/HEVC播放功能接入

AV1 编码格式软硬解播放

播放器SDK支持AV1 编码格式通过软硬解进行播放。优先使用硬解播放，不支持AV1硬解的浏览器，内部优先开启软解模式。

- AV1播放功能接入

MPEG-TS播放

播放器SDK支持MPEG-TS格式文件播放，对.ts 文件播放做了全面支持。

不同SDK版本证书的区别

使用Android、iOS、Unity、Web、鸿蒙播放器SDK均需申请License，您需要登录百度智能云控制台申请获取播放器SDK License。

不同版本SDK License的区别如下：

Android、iOS、Unity、鸿蒙：

License版本	可用功能	测试证书可用时长
标准版	除高级版SDK功能以外的SDK功能	30天
高级版	包含所有标准版和高级版SDK功能	60天

Web：

License版本	可用功能	测试证书可用时长
标准版	除高级版SDK功能以外的SDK功能	30天
高级版	包含所有标准版和高级版SDK功能	30天
高级版-永久	包含所有标准版和高级版SDK功能	30天

个人信息处理规则

播放器SDK隐私政策

合规使用说明

播放器SDK开发者个人信息保护合规指引

功能详情

百度智能云提供Web、Android、iOS及鸿蒙平台的播放器SDK，为开发者提供简单、便捷的开发接口，帮助开发者在各类终端设备上实现媒体播放功能。本文为您详细介绍各个终端播放器 SDK 支持的功能。

> 说明：
>
> 1、Web 端的功能支持情况还取决于系统和浏览器，详见 Web 浏览器功能详情。

功能模块	功能项	功能简介	Web	iOS & Android	uni-app	HarmonyOS Ne
播放协议与格式	点播或直播支持	同时支持点播播放和直播播放能力	✔	✔	✔	✔
	支持的直播播放格式	支持 RTMP、FLV、HLS、DASH 和 WebRTC 等直播视频格式	WebRTC、FLV、HLS、DASH	RTMP、FLV、HLS、（WebRTC 仅高级版支持）	RTMP、FLV、HLS	RTMP、FLV、HI 高级版支持）
	支持的点播播放格式	支持 HLS、FLV、DASH、TS、MP4 和 MP3 等点播音视频格式	HLS、FLV、DASH、MP4、MP3、（TS 仅高级版支持）	HLS、FLV、MP4、MP3、TS	HLS、FLV、MP4、MP3、TS	HLS、FLV、MP
	URL 播放	支持在线视频、本地视频以URL的方式播放	✔	✔	✔	✔
	H.264播放及软硬解	支持播放 H.264 视频源，并支持软硬解切换。	✔	✔	✔	✔
	H.265 播放	支持对 H.265 视频源的解码播放	✔（仅高级版支持）	✔	✔	✔
	AV1 播放	支持对 AV1 视频源的解码播放	✔（仅高级版支持）	✔	✔	✔

	AV1 播放	支持对 AV1 视频源的解码播放	✔（仅高级版支持）	✔	×	×
	H.266 播放	支持对 AV1 视频源的软解码播放	✔（仅高级版支持）	✔	×	×
	HDR 视频	支持播放 HDR10/HLG 等多种 HDR（High Dynamic Range，高动态范围）视频。	×	✔（仅高级版支持）	×	×
	全景VR视频	支持播放全景VR视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，web端支持鼠标在界面上拖动画面查看	✔（仅高级版支持）	✔（仅高级版支持）	×	×
	纯音频播放	支持 MP3 等文件纯音频播放	✔	✔	✔	✔
播放控制	基础播放控制	支持开始、结束、暂停和恢复等播放控制功能	✔	✔	✔	✔
	设置封面	支持设置播放视频的封面	✔	✔	✔	✔
	Seek	支持拖动到指定位置。	✔	✔	✔	✔
	精准Seek	支持以帧级别的精确度拖动到指定位置。	✔	✔	✔	✔
	缓存内Seek	在进行 Seek 操作时，已缓存的视频内容不被清除且能够快速进行 Seek。	✔	✔	✔	✔
	重播	支持视频播放结束后手动触发重播。	✔	✔	✔	✔
	续播	支持设置续播起播时间点。	✔	✔	✔	✔
	循环播放	支持视频播放结束后自动重播。	✔	✔	✔	✔
	倍速播放	支持变速播放，与此同时音频变速不变调。	✔	✔	✔	✔
	后台播放	支持界面切到后台后继续播放音频和视频	×	✔	✔	✔
	清晰度切换	支持用户流畅无卡顿地切换视频的多路清晰度流。	✔	✔	✔	✔
	音轨切换	支持播放含多音轨的视频文件，播放时可切换音轨，如从英文切换到中文	✔	✔	×	×
	截图	支持截取播放画面的任意一帧。	✔	✔	✔	✔
	SEI 回调	解析视频流中的 SEI 帧，并进行事件回调。	✔	✔	×	✔
	直播时移	支持直播时移视频流播放，可设置开始、结束和当前支持时间，支持拖动	✔	×	×	×
网络协议	自定义 Header	支持在请求音视频资源时，自定义 HTTP Header。	✔	✔	×	×
	支持 HTTPS	支持播放 HTTPS 的视频资源	✔	✔	✔	✔
	HTTP 2.0	支持 HTTP 2.0协议	✔	✔	✔	✔
音频效果	音量调节	支持调节视频音量。	✔	✔	✔	✔
	全景声播放	支持原声模式、电影模式、音乐模式、全景环绕模式等音效处理和切换能力	×	✔（仅高级版支持）	×	×
	静音	支持开启和关闭静音。	✔	✔	✔	✔
视频效果	填充模式	支持画面裁剪和填充。	✔	✔	✔	✔
	镜像	支持水平、垂直等方向的镜像	✔	✔	✔	✔
	设置播放器尺寸	支持自定义设置播放器的宽高	✔	✔	✔	✔
	亮度调节	支持播放视频时调节系统亮度	×	✔	✔	✔
	缩略图预览（雪碧图）	支持进度条缩略图预览。	✔	✔	×	✔
	全屏	支持设置全屏/退出全屏	✔	✔	✔	✔
	锁定屏幕	支持锁屏功能，包含锁定旋转和隐藏界面元素	×	✔	✔	✔
	画中画（小窗）	画中画以小窗形式播放。	✔	✔	×	✔
播放性能	多实例	支持在同一界面添加多个播放器并同时播放	✔	✔	✔	✔
	播放失败重试	播放失败时自动重试。	✔	✔	✔	✔
	预下载	支持 MP4 格式预下载	×	✔	×	×
	边播边缓存	支持在播放过程中同时缓存后续内容，降低网络占用，并可设置缓存策略。	✔	✔	✔	×
	视频下载	支持 HLS 离线下载	×	✔	×	✔
	自适应码率	支持播放 HLS、DASH可根据网络带宽自动选择合适的码率进行播放	✔	✔(仅支持hls)	×	×
互动功能	外挂字幕	支持的字幕格式有srt/ass/ssa/webvtt	✔(仅支持srt、webvtt格式)	✔	×	✔
	弹幕	支持普通弹幕播放	✔	✔	×	✔
	蒙版弹幕	播放弹幕时不遮挡人物。	✔	✔	×	✔
视频安全	时间戳防盗链	支持视频的播放地址仅能在鉴权有效时间内播放。	✔	✔	✔	✔
	百度智能云加密视频	支持百度智能云私有 DRM 加密视频播放	✔	✔	✔	✔
	HLS 标准加密	支持播放标准 AES-128 加密视频。	✔	✔	✔	✔
	动态水印	支持在播放界面添加不规则跑动的文字水印，有效防盗录	✔	×	×	×
质量上报	日志上报	支持上报播放器 SDK 日志，统计点播视频播放的埋点数据。	✔	✔	✔	✔
开发支持	事件回调	支持对播放状态回调、首帧回调、播放完成或失败回调。	✔	✔	✔	✔
	UI 组件	提供完整播放器 UI，您可以根据自身需求选用。	✔	✔	✔	✔
增值功能	全景声（WANOS）	全景声（WANOS）音频格式解码与音效处理	×	✔（仅高级版支持）	×	×
	HDR 视频	支持播放 HDR10/HLG 等多种 HDR（High Dynamic Range，高动态范围）视频。	×	✔（仅高级版支持）	×	×
	全景VR视频	支持播放全景VR视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，web端支持鼠标在界面上拖动画面查看	✔（仅高级版支持）	✔（仅高级版支持）	×	×
	超低延时直播	支持端到端延迟低于1s的WebRTC直播	✔	✔（仅高级版支持）	×	✔（仅高级版支
	投屏	支持DLNA投屏能力	×	✔（仅高级版支持）	×	×
	绿幕抠图	支持对绿色或其他纯色背景的自动识别和抠像，背景可以实时替换为2D视频画面或虚拟3D场景	×	✔（仅高级版支持）	×	×
	端上超分	支持端侧推理实现对低分辨率画面的清晰度提升、噪声和块效应去除	×	✔（仅高级版支持）	×	✔（仅高级版支

Web 播放器

简介

百度智能云播放器 Web SDK (以下简称“播放器 SDK”) 是百度官方推出的用于开发网页播放器的软件开发工具包。

百度智能云播放器 SDK Web端（cyberplayer）自 4.4.0.1 版本起需获取 License 授权后方可使用。若您无需使用高级功能，可直接申请标准版 License 继续免费使用；若您需要使用高级功能则需购买高级版

功能特性

播放器SDK为您提供丰富的特性：

- 低门槛、高灵活度

  利用SDK提供的API接口，轻松创建专业级播放应用。

- 支持HEVC/H.265浏览器硬解，适用于点播与直播场景

  对于浏览器不支持硬解场景(如，Firefox浏览器)，使用wasm软解能力(missile.js)

- 支持广泛的流式视频格式

- 支持加密视频播放

  播放器SDK支持对AES128加密的HLS视频进行解密和播放，便于企业用户对视频内容进行加密保护。

  播放器SDK支持播放通过使用百度智能云服务进行DRM加密处理过的视频。

  播放器SDK支持播放open-token透明加密的视频（注：仅限使用百度CDN）

- 字幕支持以及字幕样式定制

  播放器SDK支持SRT、VTT格式字幕的显示和字幕样式的定制。

- 支持列表播放

  播放器SDK提供了二选一的方式实现媒体资源的列表播放：

  - 在控制栏上添加“上一个”和“下一个”按钮实现媒体的切换。
  - 添加列表栏，通过鼠标点击实现指定条目的播放。

- 支持图片广告功能

  播放器Web SDK支持用户配置开场、暂停及结束广告，满足个性化视频播放需求。

- 支持多码率视频

  播放器Web SDK支持多码率视频的播放及平滑切换。

- 支持视频打点及缩略图展现

  通过视频打点和缩略图展现能提升观看者的观看体验，提前获知整个视频在不同时间段的播放内容。

- 支持添加水印

  播放器SDK支持在播放界面添加静态或者不规则跑动的文字或者图片水印，有效防盗录

- 支持画中画

  播放器SDK支持切换到画中画以小窗形式播放

- 支持自适应码率

  播放器SDK支持播放HLS自适应码流，可根据网络带宽自动选择合适的码率进行播放

- 支持纯音频播放

  播放器SDK支持纯音频播放包括格式：mp3、aac、flac、ogg、wav、opus

- 支持WebRTC

  播放器SDK支持WebRTC协议播放

- 支持移动端

  对于移动端在手机自带浏览器、Chrome浏览器、微信内置浏览器做了全面兼容

- 支持AV1浏览器硬解和软解播放

  播放器SDK支持mp4封装的AV1视频播放。优先使用硬解，对于不支持硬解（例如：Safari浏览器）的浏览器，使用wasm软解能力

- 支持直播时移

开启直播时移后，在直播期间观众可以拖动进度条跳转至任意过去时间点观看直播内容。直播时移链接通过百度智能云直播服务中配置生成

- 自定义header配置
- 支持多语言配置

  播放器SDK支持配置多语言
- 支持记忆播放、从指定位置播放

  播放器SDK支持记忆播放、从指定位置播放
- 支持视频镜像、视频旋转

  播放器SDK支持设置视频镜像、视频旋转
- 支持DASH播放

  播放器SDK支持DASH协议播放
- 支持VR播放

  播放器SDK支持VR全景视频播放，播放中可以通过陀螺仪转动或手势操作来改变视角
- 支持H.266播放

  播放器SDK支持H.266软解播放
- 支持HLS、DASH多音轨切换

  播放器SDK支持HLS、DASH多音轨切换
- 支持SEI信息解析

  播放器SDK支持MP4、HLS、MPEG-TS、FLV封装格式的SEI信息解析
- 支持HEVC/H.265自动降级

  播放器SDK支持同时传入 HEVC 和其它视频编码格式。 当浏览器不支持 HEVC 格式时，自动降级为配置的其它编码格式（如： H.264 ）的视频播放。

优势

播放器SDK有诸多优势：

- 内嵌百度自主研发的Cyberplayer内核，支持目前所有主流的视频格式，支持点播MP4/FLV/M3u8/MPEG-TS/MPEG-PS/DASH播放，以及直播HLS/DASH/HTTP-FLV/WebSocket-FLV/HTTP-TS/WebRTC的播放。弥补了原生播放器在媒体支持格式上的不足，并在兼容性、稳定性和响应速度上有了明显的提高。
- 提供简单、快捷的接口，帮助开发者高效创建媒体播放应用，有效降低了开发多媒体应用的技术门槛。
- 提供安全易用的轻量级版权保护功能，通过视频转码平台用AES128加密算法对视频文件进行加密，防止非法用户对视频内容进行复制和扩散，为您的音视频版权提供安全保障。

组件及资源

播放器SDK的完整下载包中包含：

- player：主要存放播放器SDK的cyberplayer.js

解压后的目录结构如下所示:

```
cyberplayer-<version>.zip
|   ├── skins
|   |   ├── {skin}.css (播放器样式)
|   ├── cyberplayer.js
|   ├── missile.js (用于H.265 、AV1、H.266软解使用，无需在html中单独引入)
|   ├── missile.wasm
|   ├── missile-simd.js (用于H.265 、AV1、H.266软解使用，无需在html中单独引入)
|   ├── missile-simd.wasm
|   ├── missile-min.js (用于H.265 、AV1、H.266软解使用，无需在html中单独引入)
|   ├── missile-min.wasm
|   ├── missile-min-simd.js (用于H.265 、AV1、H.266软解使用，无需在html中单独引入)
|   ├── missile-min-simd.wasm
|   ├── license.js (用于License鉴权，无需在html中单独引入)
|   ├── license.wasm
```

为便于用户便捷开发，百度智能云提供了功能完备的播放器Demo，详见web播放器演示。

协议支持

播放协议	pc浏览器
HLS	支持
MP4	支持
FLV	支持
TS	支持
WebRTC	支持
DASH	支持

功能支持

功能列表	Chrome	Safari	FireFox	Edge	IOS Safari	IOS 微信浏览器	IOS Chrome	Android Chrome	Android 自带浏览器	Android 微信浏览器
设置播放器尺寸	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
封面设置	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
倍速播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
自动播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
循环播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
多实例播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
音频播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
视频截图	支持	支持	支持	支持	支持	支持	支持	支持	部分支持	支持
全屏	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
音量调节	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
视频录制	支持	支持	支持	支持	支持	支持	支持	支持	部分支持	支持
缩略图预览	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
进度条标记	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
VID 播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持 UI 自定义	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
弹幕	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
蒙版弹幕	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
列表播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
HLS 标准加密视频播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
私有 DRM 加密视频播放	支持	支持	支持	支持	部分支持	部分支持	部分支持	部分支持	部分支持	部分支持
外挂字幕	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持 H.265 编码格式**（仅高级版支持）**	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
水印	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
画中画	支持	支持	支持	支持	支持	支持	支持	部分支持	部分支持	部分支持
自适应码率	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
清晰度切换	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持WebRTC协议播放	支持	支持	支持	支持	支持	支持	支持	支持	部分支持	支持
支持AV1软硬解播放**（仅高级版支持）**	支持	支持	支持	支持	支持	支持	支持	支持	部分支持	支持
直播时移	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
自定义header	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
多语言配置	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
记忆播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
从指定位置播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
视频镜像	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
视频旋转	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持MPEG-TS播放 **（仅高级版支持）**	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持DASH播放	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持VR播放 **（仅高级版支持）**	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持H.266播放 **（仅高级版支持）**	支持	支持	支持	支持	支持	支持	支持	支持	支持	支持
支持HLS、DASH多音轨切换	支持	支持	支持	支持	不支持	不支持	不支持	支持	部分支持	支持
支持SEI信息解析	支持	支持	支持	支持	部分支持	部分支持	部分支持	支持	部分支持	支持
支持HEVC/H.265自动降级	支持	支持	支持	支持	部分支持	部分支持	部分支持	支持	部分支持	支持

为了帮助用户快速了解如何使用播放器Web SDK，我们提供 cyberplayer Demo 展示播放器的基本功能，直播支持和相关个性化需求等。

手机和移动端的设置注意事项请参考播放器常见问题。

使用指南

本章以播放器 Web SDK v4.4.0.1版本为例，适用于源视频文件存于BOS、VOD及HLS/FLV/WebRTC直播等场景，对于HLS/FLV/WebRTC直播，请您在初始化播放器（即调用setup方法）时，添加配置参数'isLive: true'。

1.准备工作 播放器 SDK Web端（cyberplayer）自 4.4.0.1 版本起需获取 License 授权后方可使用。若您无需使用新增的高级功能，可直接申请标准版 License 以继续免费使用 cyberplayer；若您需要使用新增的高级功能则需购买高级版 cyberplayer。具体信息如下：

cyberplayer功能	功能范围	所需 License	定价	授权形态	授权单位
标准版功能	标准版功能，详见 产品功能支持	播放器 Web 端标准版 License	0元 免费申请	有限期授权	1个license最多授权1个泛域名
高级版功能	标准版功能、H265软硬解播放、Av1软硬解播放、MPEG-TS播放	播放器 Web 端高级版 License**（试用期30天）**	3999元/个/年 立即购买	有限期授权	1个license最多授权1个泛域名
高级版功能-永久	标准版功能、H265软硬解播放、Av1软硬解播放、MPEG-TS播放	播放器 Web 端高级版 License**（试用期30天）**	40000元/个 立即购买	无限期授权	1个license最多授权1个泛域名

说明

- 播放器 Web 端标准版 License 可免费申请,申请后有效期默认1年；到期后可免费申请续期
- 播放器 Web 端高级版 License ,申请后试用期30天；到期后可申请购买
- 为方便本地开发，播放器不会校验 localhost 或者 127.0.01，因此申请 License 时不需要申请这类本地服务域名
- 为方便客户使用，播放器会在License过期7天内提示客户去申请续期，超过7天播放器将会终止使用

2. 集成SDK

cyberplayer 的接入步骤包括：引入依赖、添加播放器容器、实例化播放器、使用其他功能。下面为您介绍 cyberplayer 的接入指引。通过接入 cyberplayer，您可以在网页上添加一个视频播放器。

1. 引入依赖

cyberplayer 支持以下 3 种资源获取方式。不同获取方式下，引入依赖的操作方法存在差异。

UMD引入

请您在本地的项目工程内新建 index.html 文件，在 html 页面内引入 cyberplayer 的脚本文件。 代码如下所示:

```
<script src="https://bce-cdn.bj.bcebos.com/jwplayer/4.4.0.1/cyberplayer.js"></script>
```

离线包引入

点击cyberplayer.zip，下载 SDK 压缩包至本地。 将 SDK 压缩包解压到项目文件目录下。例如，解压到 lib 目录下。

解压后的目录结构如下所示:

```
cyberplayer-<version>.zip
    |   ├─── skins
    |   |   ├─── {skin}.css (播放器样式)
    |   ├─── cyberplayer.js
    |   ├─── missile.js (用于H.265 、AV1软解使用，无需在html中单独引入)
    |   ├─── missile.wasm
    |   ├─── missile-simd.js (用于H.265 、AV1软解使用，无需在html中单独引入)
    |   ├─── missile-simd.wasm
    |   ├─── missile-min.js (用于H.265 、AV1软解使用，无需在html中单独引入)
    |   ├─── missile-min.wasm
    |   ├─── license.js (用于License鉴权，无需在html中单独引入)
    |   ├─── license.wasm
```

在您的 Web 应用代码的 html 文件中，通过`<script src="${文件路径}/cyberplayer.js"></script>` 引入依赖。例如，解压在 lib 目录下，引入依赖的代码示例如下所示。

```
<script src="./lib/cyberplayer.js" type="text/javascript"></script>
```

NPM引入

通过包管理工具将 SDK 的依赖安装到项目中。

`npm install @baiduplayer/cyberplayer`

在项目中引入 cyberplayer

`import cyberplayer from '@baiduplayer/cyberplayer';`

2. 添加播放器容器

在需要展示播放器的页面添加播放器容器，例如，在 index.html 中加入以下代码。

```
<div id="playerContainer"></div>
```

3. 实例化播放器

根据业务使用在setup方法传入相关参数使用，代码示例如下所示。

```
const playerSdk = cyberplayer('playerContainer').setup({
        width:640,
        height:360,
        autostart:true,
        stretching:'uniform',
        volume:100,
        controls:true,
        appid:'XXXXp', // 参考准备工作部分，appid对应百度智能云控制台申请 License 后的licenseID
        licenseUrl:'./lib/XXX.license', // License文件路径，百度智能云控制台申请后，将下载下来的.license文件存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
        file:"xxx.mp4"
    });
```

4. 使用其他功能

支持使用其他功能：

- 支持直播（flv、hls、WebRTC）、点播（mp4、flv、hls、ts）播放
- 支持封面设置、画中画、打点及缩略图、截图等基础功能设置

接入API具体请参考开发指南，同时为便于用户便捷开发，百度智能云提供了功能完备的播放器Demo，详见web播放器演示。

开发指南

更换Logo

通过在`header`中设置Logo url，您可以将默认的百度Logo替换为个人/组织的个性化Logo。Logo图片的大小建议为：15*15像素。

```
<style>
  .jwplayer .jw-icon-barlogo-new:before {
      content: none;
  }
  .jw-icon-barlogo-new {
      background: url('<newLogo>') no-repeat;
      background-position: center;
  }
</style>
```

下面是一段自定义Logo的简易Web播放器的完整代码：

```
 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>简单的html示例</title>
    <style>
        .jwplayer .jw-icon-barlogo-new:before {
            content: none;
        }
        .jw-icon-barlogo-new {
            background: url('http://cyberplayer.bcelive.com/icon/logo.png') no-repeat;
            background-position: center;
        }
    </style>
</head>
<body>
    <div id="playercontainer"></div>
    <script type="text/javascript" src="https://bce-cdn.bj.bcebos.com/jwplayer/4.4.0.1/cyberplayer.js"></script>
    <script type="text/javascript">
        var player = cyberplayer("playercontainer").setup({
        width : 580,
        height : 433,
        backcolor : "#FFFFFF",
        stretching : "uniform",
        appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
        licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改。

        file : "<Media_Source_In_BOS_Bucket>",
        autoStart : true,
        repeat : false,
        volume : 100,
        controls : "over"
        });
        </script>
</body>
</html>
```

设置播放类型

播放器 SDK 支持通过 `type` 参数设置播放类型。默认情况下，播放器 SDK 会根据视频文件的后缀名判断视频类型，进而按照视频文件本身的类型进行播放；设置 `type` 参数后，播放器 SDK 强制按照设置类型播放视频。

语法：

```
 type: "{mp4 | flv | m3u8 }"
```

下面的代码示例表示用mp4格式播放存于"http://domain/path/file"的音视频文件：

```
 file: "http://domain/path/file",
type: "mp4",
```

实现HEVC/H.265播放

播放器SDK支持H.265/HEVC 编码格式通过软解进行播放，添加软解配置useSoftDecoding参数，即可播放。

- 播放器内部优先使用硬解播放；
- 针对不支持H.265硬解的浏览器，内部已优先开启软解模式；
- 可以通过配置enableDecoderDegrade开启或关闭解码兼容模式；
- 软解播放的解码操作依赖设备 CPU，播放高码率、高分辨率的视频时，对 CPU 占用率非常高。

开发者可以通过在setup方法中配置**enableDecoderDegrade**开启或关闭H.265解码兼容模式。

开启解码兼容模式：（默认）

`enableDecoderDegrade: true`

- 注意：开启H.265兼容模式后，播放器会在不支持 H.265 硬解的浏览器环境下，自动降级为软解兼容模式。

关闭解码兼容模式：

`enableDecoderDegrade: false`

- 注意：关闭解码兼容模式后，对于不支持 H.265 硬解的浏览器会抛出错误，开发者可以根据错误信息进行相关的容错处理。

```
player.onError((error)=>{
        console.log('解码错误,播放失败',error)
})
```

开发者可以通过在setup方法中添加下面的代码直接使用软解功能：

```
 useSoftDecoding: true, //是否使用软解，默认为fasle
liveInfoPanel: {
    showInfo: true  //是否显示面板信息，默认为false
},
```

可通过监听performance获取的面板信息，其中面板信息参数包括：

- frameRate：帧率，单位为fps
- buffer：水位线，单位为s
- bitRate：码率，单位为kbps
- decodeFrameRate：解码帧率，单位为fps

- playStyle：播放格式：flv_live,hls_live,hls_vod,flv_vod
- height：分辨率，高度
- width：分辨率，宽度
- code：编码格式
- broken：loading记录，数组类型，其参数包括如下：

broken参数说明

- pts：当前loading pts
- time：loading时长

performance事件监听代码示例：

```
player.on('performance', (info) => {
        console.log(info);
 })
```

代码示例：

```
    var player = cyberplayer('mp4-container').setup({
      width: 640,
      height: 360,
      autostart: true,
      stretching: 'uniform',
      repeat: false,
      volume: 100,
      controls: true,
      file: 'https://bdcloud-player.cdn.bcebos.com/testvideo/hls/265/1080p/liulangdiqiu/liulangdiqiu-265-1080.m3u8',
      useSoftDecoding: true,
      liveInfoPanel: {
          showInfo: true
      },
     appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID。 注意：此功能为高级版，请使用高级版appid和对应.license文件
     licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
    });
   player.on('performance', (info) => {
      console.log(info);
   })
```

实现VVC/H.266播放

播放器SDK支持VVC/H.266 编码格式通过软解进行播放，添加软解配置useSoftDecoding参数，即可播放。

开发者可以通过在setup方法中添加下面的代码直接使用软解功能：

```
  useSoftDecoding: true, //是否使用软解，默认为fasle
liveInfoPanel: {
    showInfo: true  //是否显示面板信息，默认为false
},
```

可通过监听performance获取的面板信息，其中面板信息参数包括：

- frameRate：帧率，单位为fps
- buffer：水位线，单位为s
- bitRate：码率，单位为kbps
- decodeFrameRate：解码帧率，单位为fps
- playStyle：播放格式：flv_live,hls_live,hls_vod,flv_vod
- height：分辨率，高度
- width：分辨率，宽度
- code：编码格式
- broken：loading记录，数组类型，其参数包括如下：

broken参数说明

- pts：当前loading pts
- time：loading时长

performance事件监听代码示例：

```
player.on('performance', (info) => {
        console.log(info);
 })
```

代码示例：

```
    var player = cyberplayer('mp4-container').setup({
      width: 640,
      height: 360,
      autostart: true,
      stretching: 'uniform',
      repeat: false,
      volume: 100,
      controls: true,
      file: 'XXX-vvc.mp4',
      useSoftDecoding: true,
      liveInfoPanel: {
          showInfo: true
      },
     appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID。 注意：此功能为高级版，请使用高级版appid和对应.license文件
     licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
    });
   player.on('performance', (info) => {
      console.log(info);
   })
```

实现字幕功能

使用SDK，可以为视频添加字幕以及进行字幕样式的定制。

添加字幕

播放器Web SDK支持SRT、VTT格式的字幕。开发者可以通过在setup方法中添加下面的代码实现字幕功能。

```
  tracks : [{
    //字幕文件的URL，注意：需要考虑SDK跨域访问的局限性。
    file: "中文.srt",
    //字幕的标签
    label: "Chinese",
    //true表示默认显示字幕，false则相反。注意：default要有双引号。
    "default": true
}]
```

定制字幕样式

播放器 SDK 提供两种定制方式：

- captions选项块
- HTML标签

在setup方法中添加captions选项块

开发者可以通过在setup方法中添加captions选项块实现对字幕样式的定制。代码如下：

```
  captions: {
    //false表示背景透明，true表示背景黑色
    back: false,
    //设置字幕字体颜色
    color: 'cc0000',
    //设置字幕字体大小
    fontsize: 20
}
```

在字幕文件中添加基本的HTML标签

开发者可以通过在字幕文件中添加基本的HTML标签实现对字幕样式的定制。下面以SRT格式字幕为例，代码如下：

```
 1
00:01:11,050 --> 00:01:12,300
<b>Duke</b>?

2
00:01:13,380 --> 00:01:15,290
<font color="#3333CC">Jack, <i>how</i> are you ? </font>

3
00:01:15,290 --> 00:01:17,370
Well, <font size="30" color="#FF0000">sorry to bother you, but Sam Verdreaux called.</font>
```

示例代码：

```
var player = cyberplayer('mp4-container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.mp4',
    tracks: [
        {file: "XXX-ch.srt", label: "Chinese"}, // 字幕srt配置文件
        {file: "XXX-fa.srt", label: "Farsi"},
        {file: "XXX-gr.srt", label: "Greek"}
    ],
    captions: {
        back: true, // false表示背景透明，true表示背景黑色
        fontSize: 18, // 字体大小
        fontFamily: 'Arial,sans-serif', // 字体样式
        fontOpacity: 100, // 字体透明度
        color: '#666', // 字体颜色
        backgroundColor: '#000', // 字幕背景颜色
        backgroundOpacity: 100 // 字幕背景透明度
    },
    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
});
```

实现列表播放

播放器SDK支持中添加多个资源实现列表播放。实现了列表播放，以及列表栏显示等诸多功能。具体的参数配置如下：

可配置sources为多个链接如下配置即可实现线路、清晰度切换。其中label和title为自定义。

```
playlist: [
        {
            sources: [   // 播放源信息,以数组的形式可传入多个链接，其参数包括file、label
                {
                    file: '01-1.mp4'  // 播放源链接，例如：XXX.mp4
                    label:'高清'
                },
                {
                    file: '01-2.mp4'  // 播放源链接，例如：XXX.mp4
                    label:'标清'
                }
            ],
            title: '线路1'   //播放列表中显示的标题
        },
        {
            sources: [
                {
                    file: '02-1.mp4',
                    label:'高清'
                },
                {
                    file: '02-2.mp4',
                    label:'标清'
                }
            ],
            title: '线路2'
        }
    ]
```

如果sources数组仅有一个值时，则可以已播放列表形式展示，即没有清晰度切换按钮

```
playlist: [
        {
            sources: [   // 播放源信息
                {
                    file: '01-1.mp4'  // 播放源链接，例如：XXX.mp4
                }
            ],
            title: '线路1'   //播放列表中显示的标题
        },
        {
            sources: [
                {
                    file: '02-1.mp4',
                    label:'高清'
                }
            ],
            title: '线路2'
        }
    ]
```

接口及事件说明：

接口：

```
cyberplayer::getPlaylist()
  参数列表：无
  返回类型：Array
  接口功能：获取当前的播放列表。
  代码示例：
  var playlist = myPlayer. getPlaylist();

cyberplayer::getPlaylistIndex()
  参数列表：无
  返回类型：Number
  接口功能：获取当前正在播放视频在播放列表中的位置，0代表第一个，依次类推。
  代码示例：
  var playlistIndex = myPlayer. getPlaylistIndex();

cyberplayer::getPlaylistItem
  参数列表：无
  返回类型：Object
  接口功能：获取当前正在播放的视频对象。
  代码示例：
  var playlistItem = myPlayer. getPlaylistItem();

cyberplayer::playlistItem
  参数列表：Number
  返回类型：Object
  接口功能：播放指定的播放条目，并返回该项对象，条目从0开始计数。
  代码示例：
  var item = myPlayer.playlistItem(2);

cyberplayer::playlistNext
  参数列表：无
  返回类型：Object
  接口功能：播放当前播放条目的后一项，并返回该项对象。
  代码示例：
  var item = myPlayer.playlistNext();

 cyberplayer::playlistPrev
  参数列表：无
  返回类型：Object
  接口功能：播放当前播放条目的前一项，并返回该项对象。
  代码示例：
  var item = myPlayer.playlistPrev()
```

事件:

```
cyberplayer::onPlaylist
参数列表：Function
返回类型：无
接口功能：设置播放器播放列表接收函数。
代码示例：
  player.onPlaylist(function(event){
      console.log(event)
  });
  或使用on监听：
  player.on('playlist',(event)=>{
     console.log(event)
  })

cyberplayer::onPlaylistItem
  参数列表：Function
  返回类型：无
  接口功能：设置播放器当前播放条目变化监听函数。
  代码示例：
    player.onPlaylistItem(function(event){
        console.log(event)
    });
    或使用on监听：
    player.on('playlistItem',(event)=>{
        console.log(event)
    })
```

代码示例：

```
 var player = cyberplayer('mp4-container').setup({
   width: 640,
   height: 360,
   autostart: true,
   stretching: 'uniform',
   repeat: false,
   volume: 100,
   controls: true,
   starttime: 0,
   playlist: [
      {
         sources: [{file: 'XXX.mp4'},
         title: '线路1'
      },
      {
         sources: [{file: 'XXX.mp4'}],
         title: '线路2'
      }
   ],
   appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
   licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
   });
    // 获取播放列表信息
   const Playlist = player.getPlaylist(); //获取当前的播放列表
   const PlaylistIndex = player.getPlaylistIndex();//获取当前正在播放视频在播放列表中的位置，0代表第一个，依次类推。
   const getPlaylistItem = player.getPlaylistItem();//获取当前正在播放的视频对象
   // 设置
    var item = player.playlistItem(1);  //播放指定的播放条目，并返回该项对象，条目从0开始计数。
    var item = player.playlistNext();  //播放当前播放条目的后一项，并返回该项对象。
    var item = player.playlistPrev(); // 播放当前播放条目的后一项，并返回该项对象。
   // 事件监听
   player.on('playlist', (event) => {
      console.log(event)
   })
   player.on('playlistItem', (event) => {
      console.log(event)
   })
   player.onPlaylist(function (event) {
      console.log(event)
   });
   player.onPlaylistItem(function (event) {
      console.log(event)
   });
```

清晰度切换

播放器SDK支持清晰度切换，支持用户自行切换清晰度。

示例代码：

```
var player = cyberplayer('container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
     playlist:[{"sources": [
            {"file":"XXX.m3u8","label":"高清"},
            {"file":"XXX.m3u8","label":"标清"}],
    }],

    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
});
```

显示贴片广告

在视频片头、片尾和播放暂停时显示图片广告，支持gif、png、jpeg格式图片。

其中start是片头广告参数，pause是暂停广告参数，end是片尾广告参数。Image参数表示显示图片地址，url表示链接地址，time表示片头广告显示时长，单位为秒。

暂停广告图片显示时会被缩放到400x300，推荐使用宽高比为4:3的图片作为暂停广告。

```
imageAdvs : {
  start : {
     image : "http://xxxx/abc.gif",
     url : "http://xxxx",
     time : 10
  },
  pause : {
     image : " http://xxxx/abc.gif ",
     url : " http://xxxx"
  },
  end : {
     image : " http://xxxx/abc.gif ",
     url : " http://xxxx"
  }
}
```

实现跑马灯功能

代码示例如下：

```
marquee: {
  show: false,      // 显示与否，默认不显示
  text:'中国善良的解放路的时间',      // 跑马灯文字，默认'百度智能云'
  fontSize:20,  //   字体大小
  color:' #990033'   //   字体颜色
}
```

设置视频buffer控制

您可通过maxBufferLength配置项设置最大缓存长度，单位为秒。

- 视频类型：HLS点播视频

示例代码：

```
var player = cyberplayer('mp4-container').setup({
      width: 500,
      height: 260,
      autostart: true,
      repeat: false,
      volume: 0,
      primary: 'flash',
      startparam: 'start',
      maxBufferLength: 30,
      file: 'http://demo/test.m3u8'
  });
```

弹幕功能

播放器SDK支持显示弹幕、智能蒙版弹幕。通过在setup中配置参数支持此功能。配置弹幕后可以在播放器内部显示弹幕样式，可以设置弹幕样式，比如字体大小，字体透明度，弹幕速度，弹幕显示区域。

蒙版弹幕：可以配置支持蒙版弹幕，弹幕的播放源需要经过百度智能云服务合成的带有SEI字幕信息的MP4格式。

使用过程中可以配置如下参数：

- show（布尔类型）：是否显示弹幕开关按钮，默认false
- useMask（布尔类型）：是否配置使用蒙版弹幕，默认false
- opacity（Number类型）：配置弹幕透明度，默认为1， 取值为0-1
- fontSize（Number类型）：配置弹幕字体大小，默认为24
- speed（Number类型）：配置弹幕速度：默认为1，可配置0.25，0.5，0.74，1
- area（Number类型）：配置弹幕显示区域：默认为0.5 可配置0.25，0.5，0.74，1
- active（布尔类型）：配置默认是否开启弹幕，默认为false
- items（数据）：配置弹幕列表bulletList

bulletList配置弹幕列表，其参数为：

- show（String类型）：设置默认发送的文本
- time（String类型）：设置默认弹幕显示的时间，单位为秒

提供发送弹幕方法：

```
player.sendDanmu({text:'欢迎使用cyberplayer',time:10});
```

示例代码：

```
var player = cyberplayer('mp4-container').setup({
      width: 640,
      height: 360,
      autostart: true,
      stretching: 'uniform',
      repeat: false,
      volume: 100,
      controls: true,
      file:'XXX.mp4'
      danmaku: {
 show: true,
 fontSize: 24,
 speed: 1,
 area: 0.5,
 opacity: 1,
 items: {text:'XXX',time:'1'},
 active: false,//是否开启，默认为false
         useMask:false
   },
      appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
      licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
  });
```

播放HLS加密视频

播放器SDK支持百度智能云DRM 加密视频播放。支持播放通过百度智能云视频版权保护服务（https://cloud.baidu.com/doc/MCT/s/Skmylkuy2）进行加密的HLS视频源。其中加密方式有3种：

- Open：开放密钥，系统自动生成加密密钥，密钥公开，不设访问控制。
- PlayerBinding：绑定播放器，系统自动生成加密密钥，密钥设有访问控制。 PlayerBinding模式下密钥设有访问控制，安全性比较高，推荐使用PlayerBinding模式。

- Token: 临时口令播放授权，系统根据userKey生成密钥加密视频；播放时按照规则生成Token并发送给密钥服务验证，校验通过才能播放，安全性比较高。

其中Open和PlayerBinding模式对于播放器不需要传参数，播放器内部实现解密播放。对于Token模式，需要给播放器传入token口令用作校验，通过后才可以播放。同时需要在setup方法中传入tokenEncrypt:true作为使用token加密标识。

示例代码：

```
var player = cyberplayer('mp4-container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.m3u8',
    tokenEncrypt:true, // 说明使用token加密
    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
});
// token需要从服务器端获取
var token = '485aa70XXXec3ac_74c18025680XXXb81d53_169XXX0273';
player.on('beforePlay', function (e) {
    player.setToken(e.file, token);
});
```

UI自定义

播放器SDK支持在setup中配置参数skin控制按钮及进度条进行背景颜色、激活或者悬浮颜色、未激活颜色的设置。配置参数controlbar控制是否显示进度条上的logo、logo跳转地址、进度条是否允许拖动功能。

开发者可以通过配置controlbar来控制控制条配置，代码如下：

```
controlbar: {
  barLogo: true, // 进度条上的logo显示与否,默认true
  barLogoUrl: "https://www.baidu.com/", // 进度条上的logo的跳转地址可配置
  canDrag: true, // 进度条是否允许拖动,默认true
}
```

开发者可以通过配置skin来控制控制条样式的显示，代码如下：

```
skin: {
  name: "bce", // 默认皮肤-bce，其他可选项有beelden, bekle, five, glow, roundster, seven, six, stormtrooper, vapor
  background: "#108cee", // 背景色设置
  inactive: "#FFF", // 未激活时的颜色
  active: "red", // 悬浮或激活的严责
}
```

示例代码：

```
var player = cyberplayer('mp4-container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.m3u8',
    controlbar: {
        barLogo: true, // 进度条上的logo显示与否,默认true
        barLogoUrl: "https://www.baidu.com/", // 进度条上的logo的跳转地址可配置
        canDrag: true, // 进度条是否允许拖动,默认true
    },
    skin: {
        name: "bce", // 默认皮肤-bce，其他可选项有beelden, bekle, five, glow, roundster, seven, six, stormtrooper, vapor
        background: "#108cee", // 背景色设置
        inactive: "#FFF", // 未激活时的颜色
        active: "red", // 悬浮或激活的严责
    },
    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
});
```

水印

播放器SDK支持水印的显示，通过在setup中配置参数支持此功能。配置水印内容及样式后可以在播放器内部显示水印，可以对水印的类型、位置、字体颜色、透明度、速度等进行配置。参数说明及配置如下：

开发者可以通过配置watermark来控制水印样式显示，参数配置及说明如下：

- type（String类型）：水印类型，值可为：‘text’ or ‘image，默认为‘text’
- speed（Number类型）：水印运行速度，0-1 默认为0可理解为静态
- content（String类型）：水印内容 默认为空。type等于‘text’时，值为文本信息，type等于‘image’时，值为在线图片链接。格式可 为：’png‘、‘jpg’、‘svg’等浏览器支持的图片格式
- fontSize（Sting类型）：字体大小 默认为12px,格式为浏览器支持设置的字体大小可以为：‘px’、‘em’、‘rem’
- color（String类型）：水印颜色 默认为‘#fffff’
- left（String类型）：水印位置距离左侧位置 可以设置百分比或“Xpx”距离
- top（String类型）：水印位置距离顶部位置 可以设置百分比或“Xpx”距离
- rotate（String类型）：水印旋转角度 值为：'Xdeg'

示例代码：

```
var player = cyberplayer('container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.m3u8',
    watermark:{
        type:"text",
        speed:0.5,
        content:"这里是水印",
        opacity:1,
        fontSize:"12px",
        color:"#fff",
        left:"10px",
        bottom:"100px",
        rotate:"0deg"
    } ,
   appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
   licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
 });
```

画中画

播放器SDK支持画中画配置，配置画中画后，可通过点击播放器画中画按钮进行画中画模式切换。

开发者可以通过在setup方法中添加pictinpict属性来配置显示画中画切换icon。代码如下：

```
var player = cyberplayer('container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.mp4',
    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
    pictinpict:true
 });
```

自适应码率

播放器SDK支持播放HLS自适应码流，可根据网络带宽自动选择合适的码率进行播放，并且支持用户手动切换 HLS 视频的多路清晰度流，其中切换到自动模式时自动选择合适的码率进行播放，默认为自动。

示例代码：

```
var player = cyberplayer('container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.m3u8',
    appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
 });
```

提供分辨率更新时事件触发，代码如下：

```
player.on('hls_level_updated',(data)=>{
    console.log(data,'hls_level_updated');
    console.log(player.getHlsCurrentLevel(),'hls_level_updated')
})
```

提供获取当前hls level接口可通过调用：player.getHlsCurrentLevel()获取 其返回参数及说明如下：

- bitrate（Number类型）：当前选择的播放文件对应的bitrate
- height（Number类型）：当前选择的播放文件对应的分辨率
- id（String类型）：当前选择的播放文件对应的id
- label（Sting类型）：当前选择的播放文件对应的播放器显示label
- width（Number类型）：当前选择的播放文件对应的分辨率

纯音频播放

播放器SDK支持纯音频播放包括格式：mp3、aac、flac、ogg、wav、opus。

（注意：其兼容性取决于浏览器是否可以解码播放）

示例代码：

```
var player = cyberplayer('container').setup({
    width: 640,
    height: 360,
    autostart: true,
    stretching: 'uniform',
    repeat: false,
    volume: 100,
    controls: true,
    file: 'XXX.mp3',
    appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
    licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

WebRTC拉流播放

播放器SDK支持WebRTC协议播放，通过在setup中配置RtcPlugin对象，其中参数包括serverUrl、usedatachannel、reConnectTimeOut、reConnectTime。参数说明及配置如下：

开发者可以通过配置RtcPlugin来进行WebRTC拉流播放，参数配置及说明如下：

参数	类型	是否必填	说明
serverUrl	String	是	播放链接
usedatachannel	Boolean	否	是否开启datachannel传输数据，默认为false
reConnectTimeOut	Number	否	断线重连超时时间，默认30s
reConnectTime	Number	否	断线重连次数，默认6次
usehttps	Boolean	否	是否开启https访问，默认为false

事件监听：

开发者可通过player.on('rtcEvent',callback)来监听WebRTC事件。其中callback返回数据对象参数包括event、type。其中type值为'rtcEvent',

event值及说明如下：

参数值	说明
onDataChannelOpen	使用datachannel时触发、DataChannel通道已连接
onDataChannelMsg	使用datachannel时触发、收到对等端发送的消息
onDataChannelError	使用datachannel连接失败
onDataChannelClose	使用datachannel连接关闭
canplay	可播放
pause	播放已暂停
play	已播放
volumechange	音量发生改变
reConnecting	rtc断线正在重连中
reConnectFailed	rtc重连失败
reConnectSuccess	rtc播放错误
destroy	rtc被销毁

示例代码：

```
function defaultOpt() {
    return {
        title:'RTC',
        image:'http://cyberplayer.bcelive.com/thumbnail.jpg',
        width:640,
        height:360,
        autostart:true,
        stretching:'uniform',
        repeat:false,
        volume:100,
        controls:true,
        appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
        licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
        RtcPlugin:{"serverUrl":"webrtc:XXX","usedatachannel":true}
    }
}
const player = cyberplayer('container').setup(defaultOpt());
player.on('rtcEvent',(data)=>{
    console.log(data,'rtcEvent');
})
```

实现AV1播放

播放器SDK支持mp4封装的AV1视频播放。优先使用硬解播放，对于不支持硬解（例如：safari浏览器）的浏览器，使用wasm软解能力，在setup方法中添加软解配置useSoftDecoding参数，即可播放。

- 播放器内部优先使用硬解播放；
- 部分浏览器不支持 AV1 编码格式，如果需要播放该编码格式的视频，请开启软解；
- 可以通过配置enableDecoderDegrade开启或关闭解码兼容模式；
- 针对不支持 AV1 编码格式的浏览器，内部已优先开启软解模式；
- 软解播放的解码操作依赖设备 CPU，播放高码率、高分辨率的视频时，对 CPU 占用率非常高。

开发者可以通过在setup方法中配置**enableDecoderDegrade**开启或关闭AV1解码兼容模式。

开启解码兼容模式：（默认）

`enableDecoderDegrade: true`

- 注意：开启AV1兼容模式后，播放器会在不支持AV1硬解的浏览器环境下，自动降级为软解兼容模式。

关闭解码兼容模式：

`enableDecoderDegrade: false`

- 注意：关闭解码兼容模式后，对于不支持AV1硬解的浏览器会抛出错误，开发者可以根据错误信息进行相关的容错处理。

```
player.onError((error)=>{
        console.log('解码错误,播放失败',error)
    })
```

开发者可以通过在setup方法中配置`useSoftDecoding:true`使用软解功能：

参数	类型	是否必填	说明
useSoftDecoding	Boolean	否	是否使用软解，默认为fasle

代码示例：

```
var player = cyberplayer('container').setup({
  title:'AV1播放',
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.mp4',
  useSoftDecoding: true,
  appid:'XXXXp'，// appid对应百度智能云控制台申请 License 后的licenseID。注意：此功能为高级版，请使用高级版appid和对应.license文件
  licenseUrl:'./lib/XXX.license'，// license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
});
```

直播时移

播放器支持时移功能，开启直播时移后，在直播期间观众可以拖动进度条跳转至任意过去时间点观看直播内容。通过在setup中配置timeShift对象来控制时移时间，其中参数包括liveStartTime。

- 直播时移链接需要通过百度智能云直播服务中配置生成。

开发者可以配置timeShift时移时间，参数配置及说明如下：

参数	类型	是否必填	说明
liveStartTime	Number	是	时移的开始时间绝对时间：Unix 时间，例如：1698836590； 例如：Date.now() / 1000 - 60, 则表示观众可以看 1 分钟前的直播回放

*liveStartTime 需使用浏览器本地时间

代码示例：

```
var player = cyberplayer('container').setup({
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.m3u8',
  timeShift:{  //直播时移参数配置
    liveStartTime:Date.now() / 1000 - 60，//则表示观众可以看 1 分钟前的直播回放
  },
  appid:'XXXXp'，// appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license'，// license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

自定义header

播放器支持添加自定义header，可通过在setup方法中添加`headers:Object`，其中参数为自己定义的对象及值。

示例代码：

```
function defaultOpt() {
    return {
        title:'自定义header',
        image:'http://cyberplayer.bcelive.com/thumbnail.jpg',
        width:640,
        height:360,
        autostart:true,
        stretching:'uniform',
        repeat:false,
        volume:100,
        controls:true,
        file: 'https:XXX.m3u8',
        headers:{  //自定义header，可传入多个参数。
          authToken:'XXXX',
          Token:'XXXXX'
        },
       appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
       licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
     }
}
const player = cyberplayer('container').setup(defaultOpt());
```

多语言

播放器支持对可视文字进行多语言配置，并且提供切换语言API接口。 使用多语言功能可以在setup方法中配置如下参数：

参数	是否必填	类型	说明
lang	必填	String	默认选择的语言：值可以为:'zh'、'en'
languages	选填	Object	配置的多语言项：值见languages配置项
showLang	选填	Boolean	是否在控制栏显示多语言切换选项，默认为false。 根据languages属性中的语言配置进行选择，默认为lang对应的配置。

languages配置项说明:

参数	类型	说明
zh	Object	多语言的配置项，包含自定义的多语言中文配置项及 播放列表中lineTextKey线路、definitionTextKey清晰度 对应的配置项
en	Object	多语言的配置项，包含自定义的多语言英文配置项及 播放列表中lineTextKey线路、definitionTextKey清晰度 对应的配置项

languages配置项可配置一项也可以配置多项

对于多线路和多清晰度切换可自定义设置多语言值

例如可以自定义definitionTextKey 、lineTextKey 对应的值用来指定多语言的 key，其值对应languages配置的参数。

示例代码如下：

```
const player = cyberplayer('container').setup({
  title: "多语言",
  width: 640,
  height: 360,
  volume: 10,
  controls: true,
  appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
  lang: "en",
  showLang: true,
  playlist: [
    {
      sources: [
        {
          file: "XXX.m3u8",
          definitionTextKey: "HDKEY",
        },
        {
          file: "XXX.m3u8",
          definitionTextKey: "SDKEY",
        },
      ],
      lineTextKey: "LINENAME_ONE",
    },
    {
      sources: [
        {
          file: "XXX.m3u8",
          definitionTextKey: "HDKEY",
        },
        {
          file: "XXX.m3u8",
          definitionTextKey: "SDKEY",
        },
      ],
      lineTextKey: "LINENAME_TWO",
    },
  ],
  languages: {
    en: {
      LINENAME_ONE: "Line 1",
      LINENAME_TWO: "Line 2",
      SDKEY: "SD",
      HDKEY: "HD",
      PLAYLIST_TITLE: "play list",
      en: "english",
      zh: "chinese",
      fa: "fa",
    },
    zh: {
      LINENAME_ONE: "线路一",
      LINENAME_TWO: "线路二",
      SDKEY: "标清",
      HDKEY: "高清",
      PLAYLIST_TITLE: "播放列表",
      en: "英文",
      zh: "中文",
      fa: "法语",
    },
    fa: {
      LINENAME_ONE: "Ligne un",
      LINENAME_TWO: "Ligne deux",
      SDKEY: "Le standard Clear",
      HDKEY: "Haute définition",
      PLAYLIST_TITLE: "Playlist",
      en: "Anglais",
      zh: "Chinois",
      fa: "Français",
    },
  },
})
```

记忆播放

播放器支持记忆播放功能，记忆播放开启后，可以在您上次观看视频离开后的时间点继续播放。通过在setup方法中配置isMemoryPlay开启记忆播放。

- 播放器默认使用浏览器的 localStorage API 存储播放时间，即默认情况下不能实现跨端跨平台共享存储记忆的播放时间点。
- 与从指定位置播放配置冲突，如果都配置的情况下默认选择记忆播放

开发者可以配置isMemoryPlay，参数配置及说明如下

参数	类型	是否必填	说明
isMemoryPlay	Boolean	否	是否开启记忆播放，默认为false

代码示例：

```
var player = cyberplayer('container').setup({
  title:'记忆播放',
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.m3u8',
  isMemoryPlay:true,
  appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

从指定位置播放

播放器支持从指定位置播放，通过在setup中配置starttime用来指定开始播放的时间。

- 与记忆播放配置冲突，如果都配置的情况下，默认选择记忆播放
- 设置的播放开始时间不合法，例如：配置starttime参数值大于视频总时长，播放器会选择从头开始播放

开发者可以配置starttime，参数配置及说明如下

参数	类型	是否必填	说明
starttime	Number	否	配置起播开始时间，默认为0。单位：s

代码示例：

```
var player = cyberplayer('container').setup({
  title:'从指定位置播放',
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.m3u8',
  starttime:10,
  appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

视频镜像

播放器支持配置视频镜像功能，通过在setup方法中设置isMirror参数，配置开启后会在播放器内显示镜像切换按钮，点击后进行切换。

开发者可以配置isMirror，参数配置及说明如下

参数	类型	是否必填	说明
isMirror	Boolean	否	是否开启镜像，默认为false

代码示例：

```
var player = cyberplayer('container').setup({
  title:'视频镜像',
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.m3u8',
  isMirror:true,
  appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

视频旋转

播放器支持配置开启视频旋转功能，可以在setup设置rotate参数用来指定视频旋转。其中rotate参数包括clockwise、innerRotate。配置开启后会在播放器内显示旋转按钮，点击后会按照配置进行旋转。

开发者可以配置rotate，其参数包括clockwise、innerRotate，配置说明如下

参数	类型	是否必填	说明
clockwise	Boolean	否	指定旋转方向，true表示顺时针，false表示逆时针。默认为true。每次操作旋转90度
innerRotate	Boolean	否	是否只旋转内部 video或canvas（软解使用）。默认为true。为false时则与控制条一起旋转

代码示例：

```
var player = cyberplayer('container').setup({
  title:'视频旋转',
  width: 640,
  height: 360,
  autostart: true,
  stretching: 'uniform',
  repeat: false,
  volume: 100,
  controls: true,
  file: 'https:XXX.m3u8',
  rotate:{"clockwise":false,"innerRotate":true}
  appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
  licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入
});
```

实现DASH播放

播放器SDK支持DASH协议播放，通过在setup中设置file值为.mpd格式链接。

示例代码：

```
function defaultOpt() {
  return {
      title:'dash点播',
      width:640,
      height:360,
      autostart:true,
      stretching:'uniform',
      appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID
      licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
      repeat:false,
      volume:100,
      controls:true,
      file:"XXX.mpd"
   }
}
let player = cyberplayer('playerContainer').setup(defaultOpt());
```

实现VR播放

播放器SDK支持VR全景视频播放，播放中可以通过陀螺仪转动或手势操作来改变视角。

支持视频封装格式

播放协议	浏览器
HLS	支持
MP4	支持
FLV	支持
TS	支持
DASH	支持

初始化播放器实例时，可通过声明 VrPlugin 插件的方法开启 VR 播放能力。

示例代码：

```
`
function defaultOpt() {
        return {
            width:640,
            height:360,
            stretching:'uniform',
            VRPlugin:{
                yaw:0, // 初始化左右视角角度，单位为度。
                ... //其他参数详见：插件属性说明
            },
            appid:'XXXXp', // appid对应百度智能云控制台申请 License 后的licenseID。 注意：此功能为高级版，请使用高级版appid和对应.license文件
            licenseUrl:'./lib/XXX.license', // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
          }
}
window.player = cyberplayer('playerContainer').setup(defaultOpt());
`
```

插件属性说明如下

属性名	类型	默认值	说明
initialYaw	Number	0	初始化左右视角角度，单位为度。取值范围-360~360。
initialPitch	Number	0	初始化上下视角角度，单位为度。取值范围-180~180。
fov	Number	90	水平视场 (FOV) 角度。改变前后视角。取值范围0~180。
yawRange	Object	{min:-360,max:360}	限制左右视角活动的范围。单位为度。
pitchRange	Object	{min:-180,max:180}	限制上下视角活动的范围。单位为度。
initialZoom	Number	1	相机变焦范围。例如：zoom: 2将图像水平放大 200%。
zoomChange	Object	{min:0.6,max:10}	限制相机变焦范围。

API animateTo 在一段时间通过动画形式移动到特定角度的视角。

```
player.VR().animateTo({ yaw: 30,pitch:20,fov:30,zoom:5 }, 1000)
```

enterVR 进入VR，可通过陀螺仪转动或手势操作来改变视角。

```
player.VR().enterVR()
```

exitVR 退出VR

```
player.VR().exitVR()
```

gyro(陀螺仪控制)

isAvailable 获取陀螺仪是否可用，返回值true为可用，false为不可用。

```
const gyroAvailable =  await player.VR().gyro().isAvailable();
```

requestSensorPermission 请求陀螺仪权限。

```
await player.VR().gyro().requestSensorPermission();
```

enable 陀螺开启，进入后可以通过陀螺仪转动或手势操作来改变视角。

```
player.VR().gyro().enable()
```

disable

陀螺关闭，退出将以手势操作来改变视角。

```
player.VR().gyro().disable()
```

说明

- 在浏览器劫持播放的环境，无法支持 VR 视频的播放。
- Android 端播放器初始化后会默认进入 VR 模式，并开启陀螺仪。
- iOS 端根据系统版本不同，表现会有差异：
  - 系统版本13+，需要手动点击页面，触发人机交互并获取权限后，才能打开陀螺仪。
  - 系统版本12.2 - 13，需进入系统设置手动开启运动传感器。通常来说开启路径一般为设置 > Safari > 动作与方向访问，开启传感器后刷新页面，即可打开。
- Gyroscope陀螺仪传感器仅在安全上下文中工作。也就是本地环境：localhost，http://127.0.0.1/ ，及https:// 环境。 其余环境例如： http://baidu.com/XX/XX 不生效。

支持打点及缩略图

播放器SDK支持设置打点及缩略图，可在监听onReady后通过setCues()接口进行配置。

setCues()

设置打点及缩略图

参数说明

参数	类型	是否必填	默认值	说明
begin	Number	是	无	打点的时间，单位：s
text	String	否	无	打点标题内容
describe	String	否	无	打点描述内容
img	String	否	无	缩略图路径

代码示例

```
 function defaultOpt() {
   return {
        title:'打点及缩略图',
        width:640,
        height:360,
        autostart:true,
        stretching:'uniform',
       file: 'https:XXX.mp4',
        appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
        licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
        repeat:false,
        volume:100,
        controls:true,
     }
}
let player = cyberplayer('playerContainer').setup(defaultOpt());
player.onReady(function () {
  // 动态设置打点及缩略图。
    player.setCues([
        { begin: 19.5, text: '我的信息1',img:'./assets/img1.jpg',describe:'describe1'  },
        { begin: 50, text: '我的信息2',img:'./assets/img2.jpg',describe:'describe2' },
    ]);
})
```

实现HLS、DASH多音轨切换

播放器SDK支持HLS、DASH多音轨切换，通过在setup中设置file值为带多音轨.m3u8/.mpd格式链接。

代码示例：

```
function defaultOpt() {
  return {
      title:'多音轨切换',
      width:640,
      height:360,
      autostart:true,
      stretching:'uniform',
      appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
      licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
      repeat:false,
      volume:100,
      controls:true,
      file:"XXX.mpd" //带多音轨.m3u8/.mpd格式链接
    }
}
let player = cyberplayer('playerContainer').setup(defaultOpt());
```

支持HEVC/H.265自动降级

播放器SDK支持同时传入 HEVC 和其它视频编码格式。 当浏览器不支持 HEVC 格式时，自动降级为配置的其它编码格式（如： H.264 ）的视频播放。

```
function defaultOpt() {
  return {
      title:'HEVC/H.265自动降级',
      width:640,
      height:360,
      autostart:true,
      stretching:'uniform',
      appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
      licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
      repeat:false,
      volume:100,
      controls:true,
      playlist:[{"sources":[{"file":"XXX/265.mp4","label":"h265"},{"file":"XXX/h264.mp4","label":"h264"}]}],
      enableDecoderDegrade:false,
    }
}
let player = cyberplayer('playerContainer').setup(defaultOpt());
```

支持SEI 信息解析

播放器SDK支持MP4、HLS、MPEG-TS、FLV封装格式的SEI信息解析。

SEI信息解析事件监听代码

```
player.on('sei_parsed', (sei) => {
        console.log(sei,'这里是SEI信息');
```

}) **sei信息返回参数说明** | 参数 | 说明 | | --- |--- | | uuid | 返回sei信息的uuid | | pts | sei信息插入时间戳。单位：s | | userData | sei信息内容 | | userDataBytes | sei信息内容，格式为Uint8Array | | payloadType | sei信息类型，5为自定义sei信息|

代码示例：

```
function defaultOpt() {
  return {
      title:'SEI解析',
      width:640,
      height:360,
      autostart:true,
      stretching:'uniform',
      file: 'https:XXX.mp4',
      appid:'XXXXp',  //  appid对应百度智能云控制台申请 License 后的licenseID
      licenseUrl:'./lib/XXX.license',  // license文件路径，百度智能云控制台申请后，将下载下来的.license存放到项目目录中，以静态资源方式传入。注意：.license文件名不能更改
      repeat:false,
      volume:100,
      controls:true,
    }
}
let player = cyberplayer('playerContainer').setup(defaultOpt());
player.on('sei_parsed',(data)=>{
    console.log(data,'sei_parsed');
})
```

切换播放地址

播放器支持在不销毁播放器的情况下更换视频源，可调用播放器实例的 playNext 方法

普通地址切换

```
playerSdk.playNext({url:'XXX.mp4',starttime:10});
```

参数说明：

参数	类型	是否必填	说明
url	String	是	播放链接
starttime	Number	否	切换链接从starttime开始播放，单位：S

webrtc拉流地址切换

```
player.playNext({RtcPlugin:{"serverUrl":"webrtc://by-test.bj-webrtc-pl001.bigenemy.cn/XXX/XXX","usedatachannel":true}});
```

RtcPlugin参数说明：

参数	类型	是否必填	说明
serverUrl	String	是	webrtc播放链接
usedatachannel	Boolean	否	是否开启datachannel传输数据，默认为false
reConnectTimeOut	Number	否	断线重连超时时间，默认30s
reConnectTime	Number	否	断线重连次数，默认6次
usehttps	Boolean	否	是否开启https访问，默认为false

API说明

详情请参见API参考。

接口速查

事件响应接口

接口	参数列表	返回类型	接口功能	代码示例
cyberplayer::onConnection	Function	无	监听直播流的状态，成功建立连接时触发。	myPlayer.onConnection(function(){log("onConnection");});
cyberplayer::onCoverImageReady	Function	无	监听 视频起播图加载成功，可以点击播放。	myPlayer.onCoverImageReady(function(){log("onCoverImageReady");});
cyberplayer::onAlive	Function	无	监听直播流的状态，有直播流时触发。	myPlayer.onAlive(function(){log("onAlive");});
cyberplayer::onNoLiveStream	Function	无	监听直播流的状态，无直播流时触发。	myPlayer.onNoLiveStream(function(){log("onNoLiveStream");});
cyberplayer::onLiveStop	Function	无	监听直播流的状态，直播暂停时触发。	myPlayer.onLiveStop(function(){ log("onLiveStop");});
cyberplayer::onBuffer	Function	无	设置播放器缓存事件监听函数。	myPlayer.onBuffer(function(event){log("onBuffer");});
cyberplayer::onBufferChange	Function	void	设置播放器缓存变化监听函数。	myPlayer.onBufferChange(function(event){log("buffer percent : " + event.bufferPercent + "; position : " + event.position);});
cyberplayer::onBufferFull	Function	无	设置播放器缓存满载状态监听函数。	myPlayer.onBufferFull(function(event){log("onBufferFull");});
cyberplayer::onComplete	Function	无	设置播放器完成事件监听函数。	myPlayer.onComplete(function(event){log("onComplete");});
cyberplayer::onError	Function	无	设置播放器出错监听函数。	myPlayer.onError(function(event){log("onError");});
cyberplayer::onFullscreen	Function	无	设置播放器全屏状态变化监听函数。	myPlayer.onFullscreen(function(event){log("onFullscreen");});
cyberplayer::onIdle	Function	无	设置播放器缓空闲事件监听函数。	myPlayer.onIdle(function(event){log("onIdle");});
cyberplayer::onMeta	Function	无	设置播放器元数据接收函数。	myPlayer.onMeta(function(event){ log("onMeta");});
cyberplayer::onMute	Function	无	设置播放器音频开关变化监听函数。	myPlayer.onMute(function(event){log("onMute");});
cyberplayer::onPause	Function	无	设置播放器暂停事件监听函数。	myPlayer.onPause(function(event){log("onPause");});
cyberplayer::onPlay	Function	无	设置播放器播放事件监听函数。	myPlayer.onPlay(function(event){log("onPlay");});
cyberplayer::onPlaylist	Function	无	设置播放器播放列表接收函数。	myPlayer.onPlaylist(function(event){log("buffer percent : " + event.bufferPercent + "; position : " + event.position);});
cyberplayer::onPlaylistItem	Function	无	设置播放器当前播放条目变化监听函数。	myPlayer.onPlaylistItem(function(event){log("onPlaylistItem");});
cyberplayer::onReady	Function	无	设置播放器就绪事件监听函数。	myPlayer.onReady(function(event){log("onReady");});
cyberplayer::onResize	Function	无	设置播放器窗口大小变化监听函数。	myPlayer.onResize(function(event){log("onResize");});
cyberplayer::onSeek	Function	无	设置播放器拖动事件监听函数。	myPlayer.onSeek(function(event){log("onSeek");});
cyberplayer::onTime	Function（回调参数列表：Object - Time 事件）	无	设置播放器播放进度变化事件监听函数。	myPlayer.onSeek(function(event){log("onSeek");});
cyberplayer::onVolume	Function	无	设置播放器音量变化事件监听函数。	myPlayer.onVolume(function(event){log("onVolume");});
cyberplayer::onScreenshot	Function	canvas对象,通过canvas对象获取id，寻找页面元素，使用toDataURL方式转化图片地址	设置播放器截图事件监听函数。	myPlayer.onScreenshot(function(canvas){});
cyberplayer::onCaptureFrameFinished	Function	视频录制地址(blob:)	设置播放器视频录制事件监听函数。	myPlayer.onCaptureFrameFinished(function(event){});

控制接口

接口	参数列表	返回类型	接口功能	代码示例
cyberpl	String - 播放器DIV容器的	Object	在指定的DIV容器内创建	var myPlayer

ayer	器的id，用于盛放播放器	Object	创建一个播放器实例。	=cyberplayer("playerContainer").setup({width:600,height:450,file:"http:/, "<Image_File>"});
cyberplayer::setup	Object	无	创建一个播放器。	var myPlayer = cyberplayer("playerContainer").setup({flashplayer:"playe width:600,height:450,file:" http://ip:port/playlist.m3u8",image: "<Ima false}});参数详解见下表
cyberplayer::remove	无	无	移除当前播放器。	myPlayer.remove();
cyberplayer::getPlaylist	无	Array	获取当前的播放列表。	var playlist = myPlayer. getPlaylist();
cyberplayer::getPlaylistIndex	无	Number	获取当前正在播放视频在播放列表中的位置，0代表第一个，依次类推。	var playlistIndex = myPlayer. getPlaylistIndex();
cyberplayer::getPlaylistItem	无	object	获取当前正在播放的视频对象。	var playlistItem = myPlayer. getPlaylistItem();
cyberplayer::playlistItem	Number	object	播放指定的播放条目，并返回该项对象，条目从0开始计数。	var item = myPlayer.playlistItem(2);
cyberplayer::playlistNext	无	Object	播放当前播放条目的后一项，并返回该项对象。	var item = myPlayer.playlistNext();
cyberplayer::playlistPrev	无	Object	播放当前播放条目的前一项，并返回该项对象。	var item = myPlayer.playlistPrev();
cyberplayer::getBuffer	无	Object	获取当前正在播放视频的缓冲时长，单位为秒。	var buffer = myPlayer. getBuffer();
cyberplayer::getState	无	String	获取当前播放器状态。	var state = myPlayer.getState();取值范围：{“playing”、“paused”、“idle”
cyberplayer::play	无	无	开始播放当前媒体内容。	myPlayer.play();
cyberplayer::pause	无	Number	暂停播放当前媒体内容。	myPlayer.pause();
cyberpl			停止播	

ayer::stop	无	无	放当前媒体内容。	`myPlayer.stop();`
cyberplayer::getDuration	无	Number - 当前播放的媒体文件的总时长	获取当前播放的媒体文件的总时长。	`var duration = myPlayer.getDuration();`
cyberplayer::getPosition	无	Number	获取当前播放的媒体文件的播放位置。	`var position = myPlayer.getPosition();`
cyberplayer::seek	Number - 目标播放时间	无	定位当前媒体内容开始播放的位置。	`myPlayer.seek(102);`
cyberplayer::getMute	无	Boolean - true : 声音关闭； false : 声音打开	获取当前播放器声音是否打开。	`var mute = myPlayer.getMute();`
cyberplayer::getVolume	无	Number	获取当前播放器的音量。	`var volume = myPlayer.getVolume();`
cyberplayer::setMute	Boolean - true : 关闭声音； false : 打开声音	无	设置当前播放器的声音开关。	`myPlayer.setMute(true);`
cyberplayer::setVolume	Number - 音量大小（0-100）	无	设置播放器音量。	`myPlayer.setVolume(90);`
cyberplayer::getWidth	无	Number	获取当前播放器窗口宽度。	`myPlayer.setVolume(90);`
cyberplayer::getHeight	无	Number - 播放器高度	获取当前播放器显示窗口高度。	`var height = myPlayer.getHeight();`
cyberplayer::getFullscreen	无	Boolean - true : 全屏 ；false : 非全屏	获取当前播放器是否处于全屏状态。	`var fullscreen = myPlayer.getFullscreen();`
cyberplayer::resize	Number - 播放器宽, Number - 播放器高	无	调整播放器大小。	`myPlayer.resize(600, 400);`
cyberplayer::setFullscreen	Boolean - true : 全屏； false : 非全屏	无	设置当前播放器是否全屏。	`myPlayer.setFullscreen(true);`
cyberplayer::getQualityLevels	无	Array	获取当前视频的所有码率。	`var qualityLevels = myPlayer.getQualityLevels();`
cyberplayer::ge	无	Number	获取当前正在播放视频的码	`var currentQuality = myPlayer.getCurrentQuality();`

tCurrentQuality			率索引，从0开始。	
cyberplayer::getControls	无	Boolean	获得是否能展现播放器的control bar。	var controls = myPlayer.getControls();
cyberplayer::getSafeRegion	无	Object	获得播放器的位置对象。	var region = myPlayer.getSafeRegion();
cyberplayer::setControls	Boolean - true：显示；false：不显示	Object	设置播放器是否显示control bar。	myPlayer.setControls();
cyberplayer::getMeta	无	Object - 元数据对象	获取当前播放的媒体文件的元数据对象。	var meta = myPlayer.getMeta();
cyberplayer::getRenderingMode	无	String - 渲染模式	获取当前播放器的渲染模式。	var mode = myPlayer.getRenderingMode();取值范围：{“flash”、“html5”}
cyberplayer::getCaptionsList	无	Array	获取当前视频的所有字符对象。	var captionList = myPlayer.getCaptionList();
cyberplayer::getCurrentCaptions	无	Number	获取当前正在使用的字符文件索引，用0开始。	var currentCaptions = myPlayer.getCurrentCaptions();
cyberplayer::setCurrentCaptions	Number	Object	设置使用某种特定字幕。	myPlayer.setCurrentCaptions(1);
cyberplayer::setScreenshot	无	无	播放器视频截图。	myPlayer.setScreenshot();
cyberplayer::setCaptureFrameStart	无	无	播放器开启视频录制。	myPlayer.setCaptureFrameStart();
cyberplayer::setCaptureFrameEnd	无	无	播放器结束视频录制。	myPlayer.setCaptureFrameEnd();
cyberplayer::sendDanmu	Object	无	发送弹幕消息。	myPlayer.sendDanmu({text:'欢迎使用cyberplayer',time:10});
cyberplayer::getHlsCurrentLevel	Object	无	获取当前播放hls分辨率。	myPlayer.getHlsCurrentLevel();
cyberplayer::setWatermark	Object	无	设置水印样式	myPlayer.setWatermark({type: 'text', speed: 0.5, content: "这里是水印" '12px',color: '#fff', left: '10px',rotate: '0deg'});
cyberpl	String			

ayer::setLang	or Object	无	设置多语言	myPlayer.setLang('en');
cyberplayer::getMedialnfo	无	{video: {width:1920,height:1080,code:'avc1.64001f',frameRate:30,playStyle:'flv_live',demuxStyle:'flv',bitRate:500},audio: {bitRate:45,sampleRate:48000,code:''aac}}	获取音视频分辨率、码流(单位：kbps)、帧率(单位：fps)、编码格式、封装格式	myPlayer.setLang('en');
cyberplayer::playNext	Object	无	不重新创建播放器内核，设置播放下一个。url为播放链接，starttime设置下一个播放视频起播时间,默认为0	myPlayer.playNext({url:'XXX.mp4',starttime:10});
cyberplayer::VR	无	无	获取VR播放API	例如：设置移动到特定角度 myPlayer.VR().animateTo({ yaw: 30,pitch:20 他更多API请参见开发指南实现VR播放章节
cyberplayer::setCues	Array	无	设置打点及缩略图，更多使用说明请参见开发指南支持打点及缩略图章节	player.setCues([{ begin: 50, text: '我的信息',img:'./assets/img.jpg',des
cyberplayer::on	-	-	播放器事件监听方法，当执行了某个动作后激活。	myPlayer.on('play', function () {alert('视频已经播放了');}); 类似事件有ready, setupError, playlist, playlistItem, playlistComplete, bu buffer, idle, complete, error, seek, seeked, time, mute, volume, fullscre levelsChanged, captionsList, captionsChange, controls, displayClick, m hls_level_updated,rtcEvent,sei_parsed 等。

cyberplayer::setup接口参数详解：setup只有一个Object类型的参数，该参数是一个参数集合，该参数集合所包含的参数元素及使用方法如下描述

参数名称	参数解释	备注
width	指定要创建的播放窗口的宽度。	必选
height	指定要创建的播放窗口的高度。	必选
playlist	请参见实现列表播放	可选

imageAdvs	在视频片头、片尾和播放暂停时显示图片广告，支持gif、png、jpeg格式图片。
	start是片头广告参数，pause是暂停广告参数，end是片尾广告参数。Image参数表示显示图片地址，url表示链接地址，time表示片头广告显示时长，单位为秒。
	暂停广告图片显示时会被缩放到400x300，推荐使用宽高比为4:3的图片作为暂停广告。
	imageAdvs : {
	start : {
	image : "http://xxxx/abc.gif",
	url : "http://xxxx",
	time : 10
	},
	pause : {
	image : " http://xxxx/abc.gif ",
	url : " http://xxxx"
	},
	end : {
	image : " http://xxxx/abc.gif ",
	url : " http://xxxx"
		可选

	} }	
autostart	设置是否在播放器载入后自动播放：true：自动播放；false：不自动播放。	可选
repeat	设置视频的重复播放模式，重复模式分为： 1.false:无重复； 2.true:重复播放	可选
file	设置媒体流名称或文件名或M3U8播放列表地址	必选
image	设置媒体流的预览图	可选
screenshot	设置是否支持剪辑，布尔类型，默认false	可选
volume	设置播放器音量大小，范围（0 - 100）	可选
controls	设置播放器控制条的显示模式，显示模式分为： 1.none:不显示； 2.over:悬浮（鼠标无操作时自动隐藏）	可选
barLogo	设置是否显示Logo，controls的子参数，可选值为true (显示)，false（不显示）	可选
skin	设置播放器皮肤包	可选
useSoftDecoding	是否使用软解播放	可选
liveInfoPanel	是否显示面板信息，Object类型，参数包括showInfo，默认为false	可选
stretching	设置播放器缩放方式，缩放方式分为： 1.none:不缩放； 2.uniform:添加黑边缩放； 3. exactfit:改变宽高比缩到最大； 4.fill:剪切并缩放到最大（默认方式为uniform）	可选

watermark	设置水印样式，
	watermark:{
	type:"text",
	speed:0.5,
	content:"这里是水印",
	opacity:1,
	fontSize:"12px",
	color:"#fff",
	left:"10px",
	bottom:"100px",
	rotate:"0deg"
	}
		可选

pictinpict	设置使用画中画功能，值可为：true or false，默认为false，不显示画中画切换按钮	可选
maxBufferLength	设置最大缓存长度，单位为秒，类型为Number	可选
playRate	点播播放时是否展示倍速选择 ，默认为true，可设置为false，不展示倍速选择	可选
playRateConfig	点播播放时，可以配置倍速选择项 。例如：playRateConfig:[{"label":"×1"},{"label":"×2"},{"label":"×3"},{"label":"×4"}]	可选
RtcPlugin	rtc拉流播放配置项，详情请参见开发指南中WebRTC拉流播放章节	可选
timeShift	直播时移配置项，详情请参见开发指南中直播时移章节	可选
header	自定义headers配置项	可选
lang	默认选择的语言：值可以为:'zh'、'en' 。默认为'zh'	可选
showLang	是否在控制栏显示多语言切换选项，默认为false	可选
languages	配置的多语言项，详情请参见开发指南中多语言章节	可选
isMemoryPlay	记忆播放配置项	可选
starttime	从指定位置播放配置项，用来指定开始播放的时间 。单位：s	可选
isMirror	视频镜像配置项	可选
rotate	视频旋转配置项	可选
appid	百度智能云控制台申请 License 后的licenseID	必填
licenseUrl	百度智能云控制台申请后，将下载下来的.license文件存放到项目目录中，以静态资源方式传入	必填
VRPlugin	VR播放配置参数。例如：VRPlugin:{initialYaw:0,initialPitch:90} 更多参数请参见开发指南实现VR播放章节	选填
withLog	是否开启质量日志上报配置，默认为true	选填
isScrollShowMinScreen	是否开启滚动鼠标小屏幕展示视频，默认为false	选填

版本更新记录

版本	功能描述
v4.4.5.1	新增iOS端浏览器支持token私有 DRM 加密视频播放 、支持浏览器鼠标滚动小屏播放
v4.4.4.6	支持外部父容器绑定video节点播放、支持rtc拉流地址切换、支持配置日志开关、修复部分已知问题
v4.4.4.1	新增HLS、DASH多音轨切换，新增SEI信息解析，新增HEVC/H.265自动降级，新增WebSocket-FLV播放，优化软解倍速播放，优化视频缩略图打点接口，支持同时设置标题、打点图片、描述内容等信息
v4.4.3.2	新增播放质量监控
v4.4.3.1	新增H.266播放，优化播放器体验，并修复一些问题
v4.4.2.1	新增VR播放，优化播放器体验，并修复一些问题
v4.4.1.1	新增dash播放，软解支持倍速、seek、循环播放， 优化播放器体验，并修复一些问题
v4.4.0.1	新增license鉴权，并修复一些问题
v4.3.3.1	新增记忆播放、从指定位置播放功能，支持设置视频镜像、视频旋转，新增解码自动降级配置项，优化播放器体验，并修复一些问题
v4.3.2.1	新增WebRTC播放、AV1软硬解播放、直播时移、自定义header、多语言配置，优化播放器体验，兼容移动端播放，并修复一些问题。
v4.3.1.2	新增水印及动态水印、画中画，自适应码率、纯音频播放、清晰度切换功能，新增支持VTT格式字幕展示，新增flv直播拉流重试失败错误通知，并修复一些问题。
v4.3.0.1	新增弹幕、蒙版弹幕功能，外挂字幕、播放列表功能，提升H265视频播放体验。
v4.2.1.6	修复多实例场景视频播放失败，导致视频下载异常问题及优化手动播放场景下起播体验。
v4.2.1.2	精简内核，修复若干问题，提升H265视频播放体验。
v4.1.1.1	解决播放H264的HTTP-FLV时，部分无法起播的问题。
v4.1.0.1	解决播放H264 MP4文件时偶发onCoverImageReady事件不触发的问题。
v4.0.0	解决H264的HTTP-FLV部分流无法Resize窗口问题；解决H265 MP4播放器释放后，重创后无onCoverImageReady事件问题。
v3.9.0	对于不支持的视频格式，增加错误提示；对于HTTP-FLV直播流请求后无数据情况下，进行重试。
v3.8.0	新增直播流请求失败后重试功能；并修复一些问题。
v3.7.1	修复一些H265模式下的问题。
v3.7.0	hevc:true配置(H265播放内核)内核下,兼容H264点播直播。
v3.6.0	增加配置 hevc:true 开启H265播放内核；支持H265编码的MP4点播、FLV点播、M3U8点播、MPEG-TS点播、HTTP-FLV直播。
v3.5.3	修复Rtmp直播无法自动隐藏滚动条。
v3.5.2	修复 腾讯X5内核 自动全屏播放的问题；修复 主动引入 HLS 无效的问题；修复 Flash内核 SPS数据帧解析错误；修复 videojs 多次初始化的问题；升级 flvjs 到 1.5- 升级 adobe-air-sdk 到 31.0。
v3.4.0	chrome66 不能自动播放导致的一个bugFix；部分编码的hls视频出现跳帧的bugFix。
v3.3.0	360浏览器展示问题优化；时长展示不准确bugFix；倍速播放的一个bugFix； token加密播放使用升级。
v3.2.0	解决部分token加密模式下不能自动播放的问题；记录播放速率；其他一些已知问题的修复。
v3.1.0	解决部分浏览器兼容性问题；自动根据浏览器环境来加载videojs从而实现h5播放hls； 增加异常处理来提高其可用性。
v3.0.0	3.0版本全面支持 H5 播放flv和hls格式的视频，同时兼容flash播放器的功能，在低版本浏览器中依旧加载flash进行播放；支持 H5 播放hls直播及点播视频；优化 hls token解密播放 ，支持 adpative hls和多个视频文件多码率视频的播放；提供新版皮肤；H5播放器支持缓冲池长度设置；iphone浏览器中支持内联播放；提升稳定性。
v2.3.0	融入videojs, 支持h5 video播放加密/不加密的hls格式视频；融入flvjs, 支持h5 video播放flv格式直播和点播；h5播放器倍速播放功能；bugFix，提升稳定性。
v2.2.0	支持html5播放flv视频；支持动态设置打点及controlbar上的预览图；bugFix。
v2.1.7	支持 IE8 下使用 dom控制播放器，与高版本浏览器环境下功能保持一致；默认皮肤改成蓝色风格；可自定义邮件内容；控制条上展示当前码率；bugFix，提升稳定性。
v2.1.6	浏览器兼容性bugFix，提升稳定性。
v2.1.5	Bug Fix
v2.1.4	百度开放云更名为百度智能云。
v2.1.3	使用Flash播放MP3视频时，显示声音波形；支持自定义多码率视频列表；支持使用token方式播放加密视频；RTMP直播切换码率时进行seek操作导致黑屏bugFix；观看HLS直播较长时间后自动断开bugFix。
v2.1.2	支持直播状态回调；延长hls直播时的重试时间；播放mp4/flv过程中切换码率时视频时长显示不正常bugFix。
v2.1.1	支持加密视频自定义playerId和playerKey；增加disableSeekForward参数和onSeekForwardForbidden时间，支持禁用向前seek；支持打开GPU渲染；规范报错信息；直播暂停bugFix；视频缓冲时点击暂停出错bugFix。
v2.1.0	支持视频buffer控制；兼容IE8以下版本的浏览器。
v2.0.3	增加重连与备用机制用于处理直播异常；支持设置提示点cuepoint用于插入广告等功能。
v2.0.2	增加跑马灯功能；增加码率记忆功能；老版本参数兼容；时间过短的音视频文件播放失败bugFix。
v2.0.1	兼容不支持localstorage浏览器播放失败bugFix；编码过的视频播放失败bugFix； 调用stop接口后，播放器进度条未归零bugFix；视频播放结束后，已播放时长和视频总时长不一致bugFix。
v2.0	支持播放器皮肤的个性化设置；优化声音调节体验；优化多码率播放视频功能；提供更健壮的API。
v1.7	播放器添加 Baidu Logo； 播放器添加 AK 鉴权；Bug fix：Adaptive hls子视频流url二次encode。

license指引

概述

播放器 SDK 提供直播播放和点播播放能力，web播放器分标准版和高级版：其中标准版免费；高级版按license/年收费，1个license最多授权1个泛域名。您需要获取对应 License 后方可使用对应功能。 License 获

取和使用方式如下：

1、License申请

您可参考产品功能选择您需要申请的 License，包括标准版和高级版。 其中License与您需要申请域名进行绑定。试用周期1个月，结束后正式使用请购买License。

2、License购买

点击**购买License**后，您需将您的申请License与您需要授权的域名进行绑定。指引见下文。

3、播放器License配置

完成申请或者绑定后，您可以在百度智能云控制台下载**对应的证书**和**licenseID**用于播放器传值参数。详见集成文档

License申请和购买

web播放器分标准版和高级版：标准版免费；高级版按license/年收费，1个license最多授权1个泛域名。

cyberplayer功能	功能范围	所需 License	定价	授权形态	授权单位
标准版功能	标准版功能，详见 产品功能支持	播放器 Web 端标准版 License	0元 免费申请	有限期授权	1个license最多授权1个泛域名
高级版功能	标准版功能、H265软硬解播放、Av1软硬解播放、MPEG-TS播放	播放器 Web 端高级版 License **（试用期30天）**	3999元/个/年 立即购买	有限期授权	1个license最多授权1个泛域名
高级版功能-永久	标准版功能、H265软硬解播放、Av1软硬解播放、MPEG-TS播放	播放器 Web 端高级版 License **（试用期30天）**	40000元/个 立即购买	无限期授权	1个license最多授权1个泛域名

说明：Web 端播放器 SDK （cyberplayer）支持使用版本4.4.0.1及以上

申请License

申请播放器SDK license：您需要登录百度智能云控制台申请获取播放器SDK license。

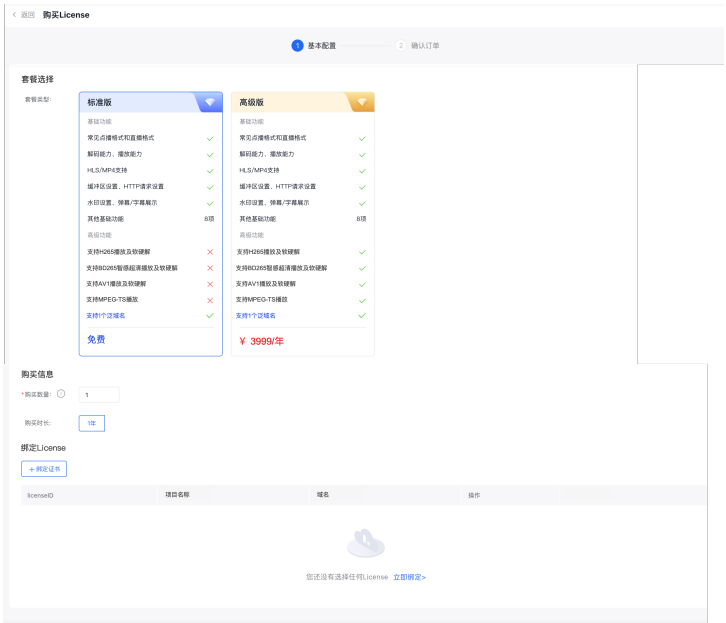
购买、绑定License 申请License后可在百度智能云控制台-web播放器页面查看License申请列表，可免费试用1个月，结束后正式使用请按照如下步骤购买、绑定License。

购买、绑定License需要您登录**百度智能云控制台 > 智能视频 SDK > 播放器SDK > web播放器 > License购买/申请购买**

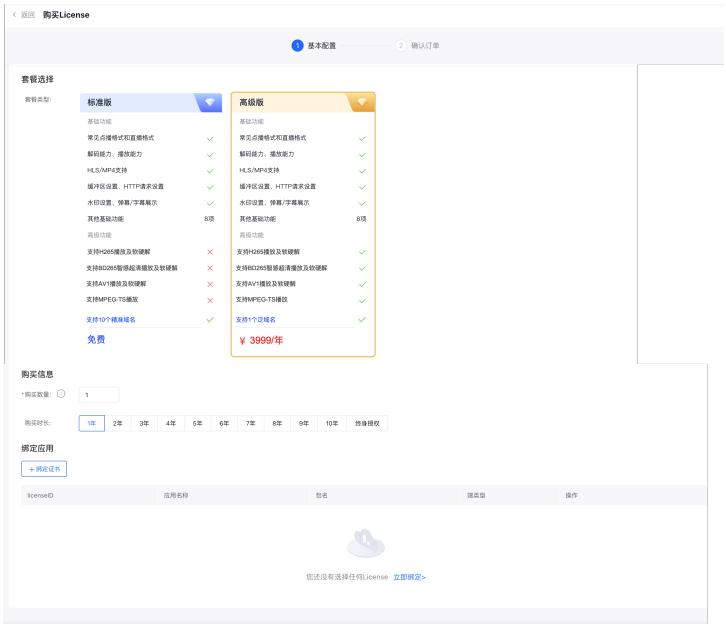
点击License购买或者申请购买

选择套餐类型：标准版和高级版

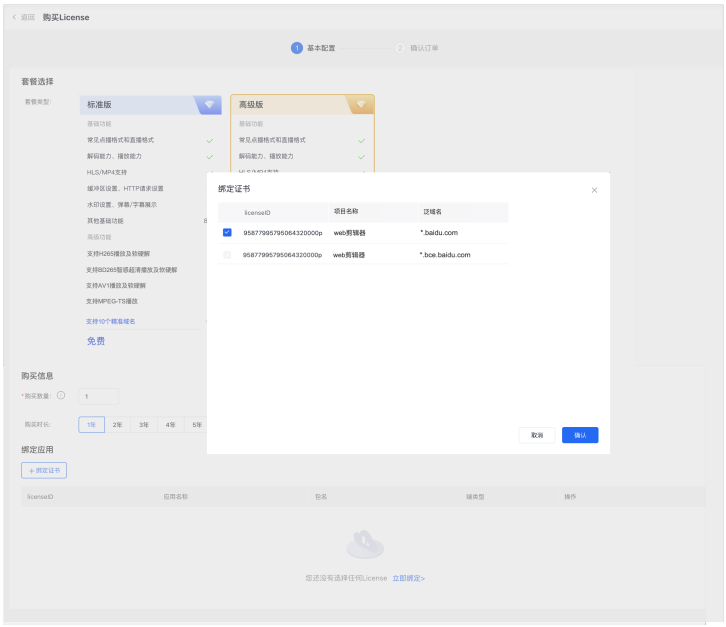
标准版



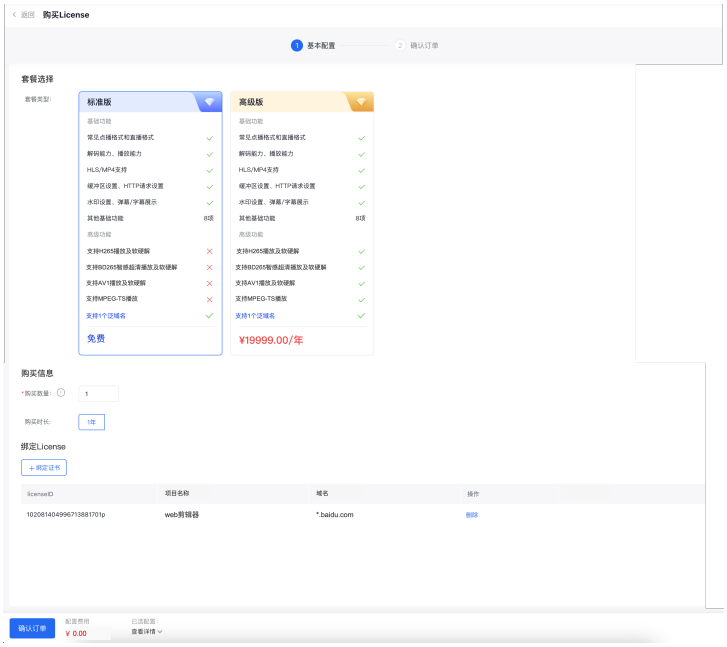
高级版



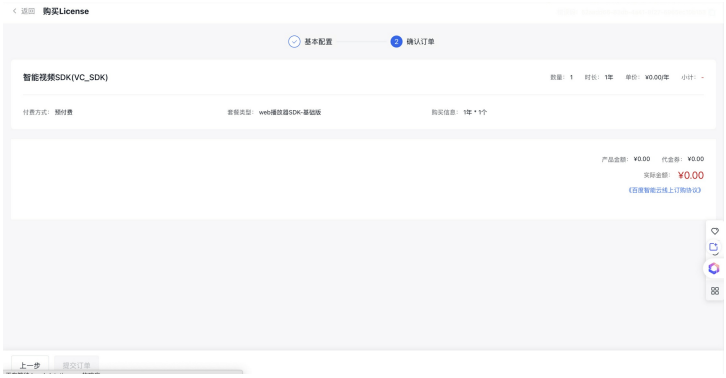
绑定证书



确认订单



提交订单



开通成功

购买成功后可在百度智能云控制台-web播放器查看License列表项**授权状态**变为正式使用

iOS播放器

简介

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 iOS 编程经验。

简介

百度智能云播放器 iOS SDK(以下简称"SDK")是百度智能云推出的 iOS 平台视频播放器软件开发工具包(SDK)，为 iOS 开发者提供简单、便捷的开发接口，帮助开发者在 iPad/iPhone/iPod 和 Apple TV 设备上实现媒体播放功能。SDK 提供简单、便捷的媒体应用开发能力。

- 本地全媒体格式支持 突破 iOS 平台对视频格式的限制，支持目前所有主流的媒体格式(mp4、avi、wmv、flv、mkv、mov、 rmvb 等)。
- 支持广泛的流式视频格式

  支持多种格式文件渐进式和流式播放: HLS、RTMP、HTTP Streaming。
- 性能强大

  CPU/内存占用率低，视频加载速度快。
- 低门槛、高灵活度实现播放功能

  提供了与系统播放器 MPMoviePlayerController 高度相似的调用接口，便于开发者快速开发媒体播放应用，同时提供开发示例。
- 针对流媒体场景进行优化

  提供专门面向流媒体场景的SDK，支持RTMP、HTTP+FLV、HLS协议及H264、HEVC和AAC编码，包体积更小。
- 媒体文件缓存预取

  支持媒体文件播放前预先加载、预先建立连接，起播放更快。支持媒体流边播边缓存，重复播放时节省流量。
- 版权保护

  支持HLS加密视频的离线下载和播放。
- 极低的接入成本

  支持Cocoapods接入方式。

功能列表

功能列表

- 与MPMoviePlayerController接口高度一致
- 版本支持
  - 全媒体版本支持所有常见本地、在线媒体格式
  - 流媒体版本支持RTMP、HTTP-FLV、HLS等点/直播媒体格式
- 播放
  - 支持首屏秒开
  - 支持追帧播放
  - 支持多实例播放
  - 支持单实例多次播放
  - 支持纯音频播放
  - 支持倍速播放
  - 支持循环播放

    - 支持续播
    - 支持后台播放
    - 支持后台自动暂停
    - 支持音频中断自动暂停
    - 支持网速探测
    - 支持网络数据代理
    - 支持多种画幅缩放模式
    - 支持IPV6
    - 支持精准seek
    - 支持seek缩略图显示
    - 支持多维手势交互，包括锁屏、音量、亮度、进度、缩放调节等
    - 支持画中画悬浮小窗播放
    - 支持短视频、Feed流场景播放
    - 支持"听"视频
    - 支持耳机操作
    - 支持多音轨、多字幕切换
- 解码
    - 支持H264硬解
    - 支持HEVC硬解
    - 支持解码方式设置
    - 支持AV1解码
- HLS支持
    - 支持 HLS 离线下载
    - 支持 HLS 多码率无缝切换
    - 支持 HLS 分片请求回调
- MP4支持
    - 支持 MP4 格式预下载/边播边存
    - 支持 MP4 多码率无缝切换
- 缓冲区设置
    - 支持缓冲区大小设置
    - 支持缓冲区时长设置
- HTTP请求设置
    - 支持设置HTTP请求的Header
    - 支持设置HTTP请求的UserAgent
- 水印设置
    - 支持添加水印
    - 支持实时更新水印位置
- 支持ATS
- 支持APM
- 支持DRM版权保护
- 支持播放中截图
- 支持边播放边录制到本地MP4文件
- 支持SEI信息更新回调
- 支持外部渲染
- 弹幕/字幕展示
    - 支持弹幕展示和交互
    - 支持智能防挡弹幕
    - 支持字幕解析和展示
    - 支持外挂字幕
- 支持VR全景视频播放
- 支持投屏
- 支持端上超分

SDK集成

开发环境

- Xcode 9.2
- iOS 9.0 及以上版本

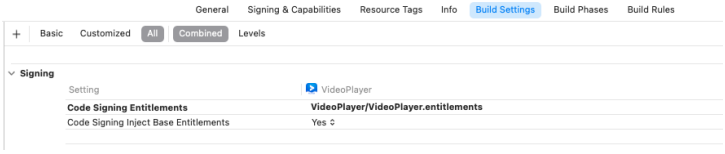
Demo编译指引

进入VideoPlayer目录，执行

```
 pod install
pod update
```

然后打开VideoPlayer.xcworkspace工程，设置包名、TeamID，配置您申请的播放器证书文件和LicenseID，即可开始编译运行。 避免Demo工程路径中的特殊字符或中文，这样可以避免一些XCode的错误。

因为涉及到DLNA投屏功能所需的组播权限，在Demo工程中配置了VideoPlayer.entitlements，若你的包名暂未申请组播权限，可以在XCode的Build Setting中将其删除，避免影响Demo工程编译构建。



Cocoapods快速集成

使用Cocoapods接入方式非常简单，可参考Cocoapods接入方式。

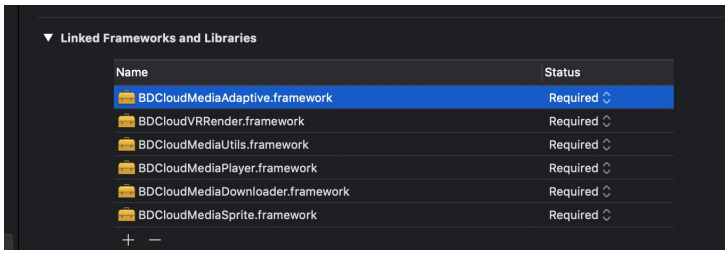
如果使用Cocoapods接入方式，可跳过手动集成。

手动集成

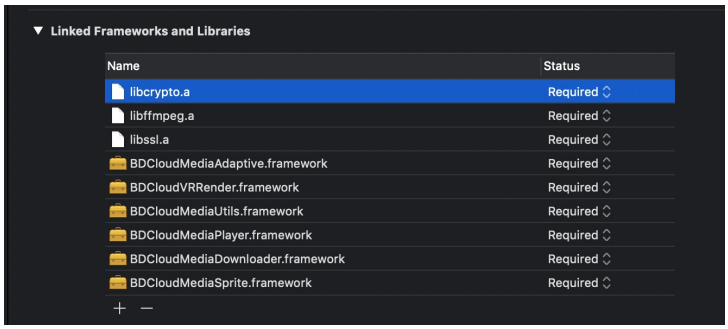
1. 下载最新的播放器 iOS SDK并解压；

```
 Baidu-Cloud-Player-<Type>-<Version>
|------frameworks
|       |-------BDCloudMediaUtils.framework
|       |-------BDCloudMediaPlayer.framework
|       |-------BDCloudMediaDownloader.framework
|       |-------BDCloudMediaSprite.framework
|       |-------BDCloudMediaSource.framework
|       |-------BDCloudMediaAdaptive.framework
|       |-------BaiduAPMAgent.framework
|       |-------BDCloudVRRender.framework       // VR渲染组件，高级版SDK特有
|       |-------BDCloudHdrKit.framework         // HDR渲染组件，高级版SDK特有
|       |-------BaiduRtcPlayerGeneral.framework // 超低延时直播组件，高级版SDK特有
|       |-------ProjectionEngine.framework      // 投屏组件，高级版SDK特有
|       |-------BDCloudVirtualLiveKit.framework // 绿幕抠图组件，高级版SDK特有
|       |-------BDCloudSrKit.framework          // 端上超分组件，高级版SDK特有
|
|------vendor
|       |-------libffmpeg.a
|       |-------libcrypto.a
|       |-------libssl.a
|       |-------libwanosdecoder.a // 全景声解码组件，高级版SDK特有
|       |-------libauthcheck.a    // 全景声鉴权组件，高级版SDK特有
|       |-------libGRF.a          // 渲染组件，高级版SDK特有
|
|------VideoPlayer
        |-------VideoPlayer.xcodeproj
        |-------VideoPlayer
```

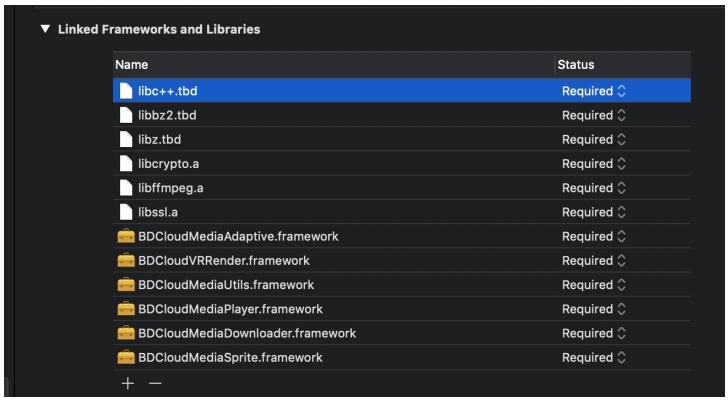
2. 将BDCloudMediaUtils.framework、BDCloudMediaPlayer.framework、BaiduAPMAgent.framework添加到项目中。
    1. 如要使用HLS离线下载功能，将BDCloudMediaDownloader.framework添加到项目中；
    2. 如要使用缩略图显示功能，将BDCloudMediaSprite.framework添加到项目中；
    3. 如要使用网络视频加速功能，将BDCloudMediaSource.framework添加到项目中；
    4. 如果需要使用自适应码率切换功能，将BDCloudMediaAdaptive.framework添加到项目中；
    5. 其他高级版SDK特有组件的使用方式请参考"高级版功能接入"文档



3. 将libcrypto.a、libssl.a、libffmpeg.a添加到Build Phases的Link Binary With Libraries中；

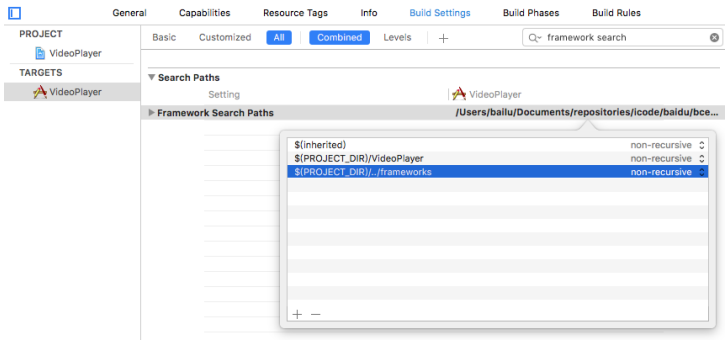


4. 将libz.tbd、libbz2.tbd添加到Build Phases的Link Binary With Libraries中；



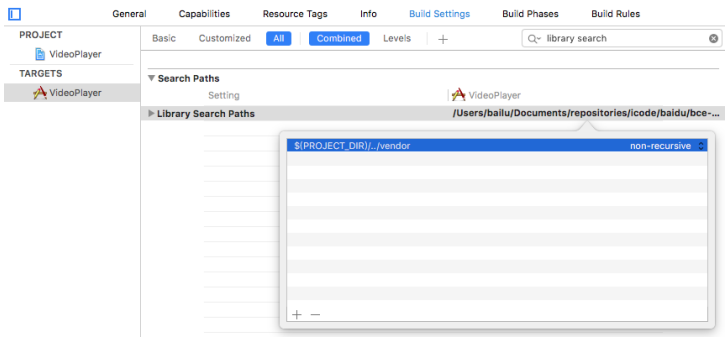
5. 设置合适的Framework Search Paths以保证链接时能找到:

- BDCloudMediaUtils.framework
- BDCloudMediaPlayer.framework
- BDCloudMediaDownloader.framework
- BDCloudMediaSprite.framework
- BDCloudMediaSource.framework
- BDCloudMediaAdaptive.framework



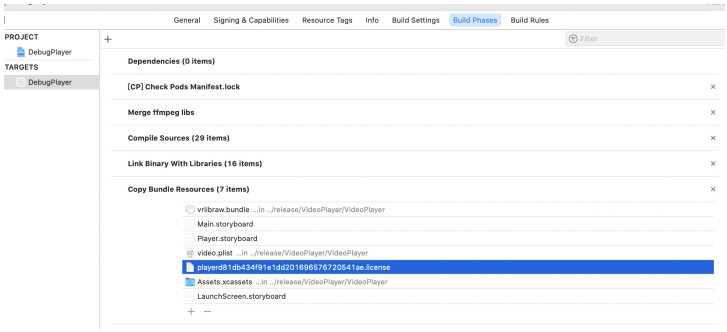
6. 设置合适的Library Search Paths以保证链接时能找到:

- libcrypto.a
- libssl.a
- libffmpeg.a



7. 添加鉴权文件依赖；

- 鉴权文件为：playerxxxxxxxxxxxx.license
- 鉴权文件申请路径：鉴权文件申请



快速开始

设置License ID

在创建播放器实例前，必须设置License ID。否则程序无法正常运行。

用户在使用SDK之前需要获取百度智能云播放器 SDK license， 参考鉴权文件申请 获取license ID。

在您的 AppDelegate 类中实现协议`BDCloudMediaPlayerAuthDelegate`，并设置license ID:

```
[[BDCloudMediaPlayerAuth sharedInstance] authenticateLicenseID:@"8087135619873945600 0p"
                                             completionHandler:^(NSError *error) {
    if (!error) {
        NSLog(@"success");
    }
}];

- (void)authStart {
    // 认证开始。
}

- (void)authEnd:(NSError*)error {
    // 认证完成，error为空表示认证成功。
}
```

播放视频

`BDCloudMediaPlayerController`是SDK的核心类，视频的播放和各种控制都是通过此类来实现。

以下代码演示：

- 创建播放器;
- 设置视频播放地址;
- 播放视频

```
 // 假设self是一个UIViewController的实例。
// 创建播放器并设置视频播放地址。
self.player = [[BDCloudMediaPlayerController alloc] initWithContentString:@"<url>"];

// 将播放器的 view 添加到 self 的 view 中。
[self.view addSubView:self.player.view];

// 为播放器 view 添加布局。
// ...

// 设置视频初始化完成后自动播放。
self.player.shoudAutoPlay = YES;

// 进行视频初始化。
[self.player prepareToPlay];
```

播放器在设置播放视频地址后，需要调用 `prepareToPlay` 方法对视频文件进行初始化工作。

初始化完成后，播放器将发送`BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification`通知，并将`isPreparedToPlay`属性置为 `YES`。 如果`shouldAutoplay`属性为`YES`，则自动调用`play`方法进行播放；如果`shouldAutoplay`属性为`NO`，则等待调用 `play` 方法播放。

暂停、继续播放

暂停正在播放的视频：

```
[self.player pause];
```

继续播放被暂停的视频:

```
[self.player play];
```

改变播放位置

SDK 提供了多种改变播放位置的方式。

初始化时

在调用`prepareToPlay`方法之前，可以设置`initialPlaybackTime`来设置起始播放的秒数。

```
self.player.initialPlaybackTime = 10.0f;
```

播放过程中

有两种方法在播放过程中改变播放位置：

- 设置currentPlaybackTime属性；

```
self.player.currentPlaybackTime = 10.0f;
```

- 调用seek方法。

```
[self.player seek:10.0f];
```

停止播放

调用stop方法停止播放。

```
[self.player stop];
```

停止播放后，播放器会发送BDCloudMediaPlayerPlaybackDidFinishNotification通知。

快速进阶

正确释放播放器

保证App稳定、高效的运行，正确释放播放器非常重要。

目前主流的交互基本是这样的：

1. 用户点击视频列表中的某一个视频，将播放UI**push**到UINavigationController上；
2. 播放UI开始加载视频并播放；
3. 用户点击后退按钮退出播放，播放UI从UINavigationController **pop** 掉。
   这里重点关注第3步，介绍释放播放器的时机、流程：
   - 当用户点击后退按钮时，只调用播放器的stop方法；

     ```
     - (IBAction)onBackButton:(id)sender {
         [self.player stop];
     }
     ```

   - 等待播放器发送BDCloudMediaPlayerPlaybackDidFinishNotification通知后，再将播放UI从UINavigationController **pop** 掉。

     ```
     // 注册通知
     - (void)setupNotifications {
         [[NSNotificationCenter defaultCenter] addObserver:self
                                                  selector:@selector(onPlayerFinish:)
                                                      name:BDCloudMediaPlayerPlaybackDidFinishNotification
                                                    object:nil];
     }

     // 等到Finish通知后再pop
     - (void)onPlayerFinish:(NSNotification*)notification {
         // 如果同时有多个播放器实例的话，最好判断下通知里的object是否跟持有的是同一个。
         if (notification.object != self.player) {
             return;
         }

         // pop UI
         [self.navigationController popViewControllerAnimated:YES];
     }
     ```

   这样的流程可以保证在退出播放UI时，播放器内部本次播放相关资源也已释放完毕，有利于复用播放器实例，发起下一次播放。

进度条相关

视频播放器进度条上包括下列3个信息：

- 视频总时长duration
- 当前播放时间currentPlaybackTime
- 当前缓冲时间playableDuration

在收到BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification通知后，可以获取duration确定进度条的maximumValue(直播视频可能获取到的值为零)。并启动一个每秒触发一次的NSTimer，来实时获取currentPlaybackTime和playableDuration，更新进度条。

> **注意:** iOS系统没有提供带缓冲进度的进度条，需要用户自行实现。

播放下一个视频

正在播放视频时，如果想放弃当前的播放，并立即在当前播放UI播放另一个视频时，可以按照下面的方式来操作：

```
- (IBAction)onPlayNext:(id)sender {
    // 停止当前视频的播放。
    [self.player stop];

    // reset播放器到初始状态。
    [self.player reset];

    // 开始下一个视频的初始化和播放流程。
    self.player.contentString = @"<next url>";
    self.player.shouldAutoplay = YES;
    [self.player prepareToPlay];
}
```

APM接入

APM SDK 提供数据监控、发送、后台报表处理与展示能力。实时性强，能够及时上报用户在点播、直播中的卡顿、网速等信息，方便运营方及时调整策略和调度。

SDK不直接依赖APM SDK，在运行时动态检查App是否接入了APM SDK。

多码率切换

多码率快速切换 当播放的是HLS多码率视频时，播放器支持在播放过程中实时切换码率。具体操作步骤如下：

1. 在收到BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification通知后，调用获取多码率列表的接口；
2. 将多码率列表显示在播放UI上供用户选择码率；
   以上两步代码示例：

```
- (void)onPlayerPrepared:(NSNotification*)notification {
    // 如果同时有多个播放器实例的话，最好判断下通知里的object是否跟持有的是同一个。
    if (notification.object != self.player) {
        return;
    }

    // 获取多码率列表。
    NSArray<BDCloudMediaPlayerBitrateItem*>* items = [self.player getSupportedBitrates];

    // 将码率列表显示在UI上。
    [self showBitrateList:items];
}
```

3. 用户选择码率时，调用setBitrateIndex:方法设置码率索引：

```
- (IBAction)onChangeBitrate:(id)sender {
    NSInteger index = <bitrateIndex>;
    [self.player setBitrateIndex:index];
}
```

多码率无缝切换 播放器不仅支持HLS的多码率快速切换，同时也支持HLS,MP4等主流媒体格式的无缝切换功能。具体操作如下：

Master playlist 的HLS格式多码率无缝切换

1. 设置输入格式

```
- (void)setMediaInputType:(NSInteger)inputType {
    [self.player setMediaItemsInputType:inputType];
}
```

2. 在收到BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification通知后，调用获取多码率列表的接口，显示到UI。

```
- (void)onPlayerPrepared:(NSNotification*)notification {
    // 如果同时有多个播放器实例的话，最好判断下通知里的object是否跟持有的是同一个。
    if (notification.object != self.player) {
        return;
    }

    // 获取多码率列表。
    NSArray* bitrates = [self.player getMediaItems];

    // 获取当前码率索引
    int index = (int)[self.player mediaItemIndex];

    // 将码率列表显示在UI上。
    [self.actions updateBitrateList:bitrates index:index];
}
```

3. 用户根据对应索引切换到指定码率。

```
- (void)changeBitrate:(NSInteger)index {
    [self.player setMediaItemIndex:index];
}
```

4. 当收到BDCloudMediaPlayerMediaChangeStartNotification 通知表示视频开始切换。
5. 当收到BDCloudMediaPlayerMediaChangeEndNotification 通知表示切换结束，同时可以取BDCloudMediaPlayerMediaChangeEndResultUserInfoKey 的值判断切换是否成功。此时播放器缓冲播放完成后即会切换到新的码率显示。
6. 当收到BDCloudMediaPlayerNaturalSizeChangedNotification 通知表示视频分辨率发生变化。

MP4等非嵌套码率的无缝切换

1. 设置输入格式

```
- (void)setMediaInputType:(NSInteger)inputType {
    [self.player setMediaItemsInputType:inputType];
}
```

2. 设置多码率视频链接（注意这里和HLS的区别）

```
- (void)setMediaInputList:(NSArray *)inputList {
    [self.player setMediaItems:inputList];
}
```

3. 在收到BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification通知后，调用获取多码率列表的接口，显示到UI。

```
- (void)onPlayerPrepared:(NSNotification*)notification {
    // 如果同时有多个播放器实例的话，最好判断下通知里的object是否跟持有的是同一个。
    if (notification.object != self.player) {
        return;
    }

    // 获取多码率列表。
    NSArray* bitrates = [self.player getMediaItems];

    // 获取当前码率索引
    int index = (int)[self.player mediaItemIndex];

    // 将码率列表显示在UI上。
    [self.actions updateBitrateList:bitrates index:index];
}
```

4. 用户根据对应索引切换到指定码率。

```
- (void)changeBitrate:(NSInteger)index {
    [self.player setMediaItemIndex:index];
}
```

5. 当收到BDCloudMediaPlayerMediaChangeStartNotification 通知表示视频开始切换。
6. 当收到BDCloudMediaPlayerMediaChangeEndNotification 通知表示切换结束，同时可以取BDCloudMediaPlayerMediaChangeEndResultUserInfoKey 的值判断切换是否成功。此时播放器缓冲播放完成后即会切换到新的码率显示。
7. 当收到BDCloudMediaPlayerNaturalSizeChangedNotification 通知表示视频分辨率发生变化。

播放HLS加密视频

百度智能云MCP服务支持转码成HLS加密视频。

不同的加密方式，播放时的方法略有不同：

- PlayerBinding加密模式 按普通视频播放即可。
- 自定义PlayerId、PlayerKey模式

  播放之前需要先设置`playerId`和`playerKey`。

```
- (void)play {
    // 先设置`playerId`和`playerKey`。
    [self.player setPlayerID:@"<playerId>" key:@"<playerKey>"];

    // 再播放视频。
    self.player.contentString = @"<url>";
    self.player.shouldAutoplay = YES;
    [self.player prepareToPlay];
}
```

- Token模式

  播放之前需要先设置`token`。

```
- (void)play {
    // 先设置`token`。
    [self.player setToken:@"<token>"];

    // 再播放视频。
    self.player.contentString = @"<url>";
    self.player.shouldAutoplay = YES;
    [self.player prepareToPlay];
}
```

设置HTTP请求的Header

如果需要设置常见的一些HTTP Header字段，例如`User-Agent`，可以使用接口`setOptionValue`。

```
NSString* value = @"User-Agent: <your user agent>";
[player setOptionValue:value
                forKey:@"headers"
            ofCategory:BDCloudMediaPlayerOptionCategoryFormat];
```

> **说明：**
>
> 如果需要设置多个HTTP Header字段，需要按照HTTP规范，将多个字段使用`\r\n`连接起来。

下载HLS视频

SDK 除了提供视频播放功能之外，还支持 HLS 视频的下载。类`BDCloudMediaDownload`提供下载相关接口。

普通HLS视频的下载

下面介绍下载的基本流程：

1. 实现`BDCloudMediaDownloadDelegate`来响应下载任务的状态变化回调

```
// 任务开始
- (void)taskStart:(BDCloudMediaDownloadTask*)task {
}

// 任务的进度汇报
- (void)task:(BDCloudMediaDownloadTask*)task progress:(float)progress {
    NSLog(@"任务%@, 下载进度 %.2f", task, progress);
}

// 任务完成，error为nil表示下载成功。
- (void)taskEnd:(BDCloudMediaDownloadTask*)task error:(NSError*)error {
    if (!error) {
        NSLog(@"下载成功!");
        // 获取离线视频的本地路径，拿到之后就可以离线播放。
        NSString* path = [task.item.path stringByAppendingPathComponent task.item.index];
    }
}
```

2. 创建下载器

下载器初始化时，需要传入用户名，用于在本地区分不同登录用户的下载存储(如果App没有账号系统，可以传一个固定值)。

```
self.downloader = [[BDCloudMediaDownload alloc] initWithUser:@"<user>" delegate:self];
```

3. 创建下载任务、开始下载

创建下载任务后，任务会进入到执行队列中，队列的最大并发数是4个。当超出最大并发数时，新创建的任务会处于等待执行的状态。

```
NSError* error;
BDCloudMediaDownloadTask* task = [self.downloader downloadTaskWithURL:@"<url>" title:@"<title>" error:&error];
```

> **注意：**
>
> - 传入的url必须以http或https开头，以.m3u8结尾。
> - 如果传入的url曾经下载成功，并且也没有调用过 removeMediaItem接口对视频数据进行清理，创建下载任务将会失败，error的code会被赋值为BDCloudMediaDownloadErrorCodeAlreadyExists。

加密HLS视频的下载

加密HLS视频的下载与普通的HLS视频下载相比，多出设置认证信息的过程，只需要实现代理方法task:needAuthentication:。

```
- (void)task:(BDCloudMediaDownloadTask*)task needAuthentication:(NSMutableDictionary*)parameters {
    // 当下载的视频是PlayerID， PlayerKey加密方式时，需要在parameters里设置playerId
    [parameters setObject:@"<playerId>" forKey:@"playerId"];

    // 当下载的视频是token加密方式时，需要在parameters里设置token
    [parameters setObject:@"<token>" forKey:@"token"];
}
```

任务的暂停和恢复

下载任务支持暂停和恢复:

- 暂停时，下载到本地的数据不会丢失;
- 恢复时，会从上次下载的位置继续下载。

```
// 暂停任务
[self.downloader suspendTask:task];

// 恢复任务
[self.downloader resumeTask:task];
```

断点续传

当App重新打开要继续上次未完成的下载时，需要先调用resumeUncompletedTasks方法(SDK内部会根据保存的任务状态，自动创建未完成的任务并加入下载队列)返回未完成的下载任务。

```
self.downloader = [[BDCloudMediaDownload alloc] initWithUser:@"<user>" delegate:self];

// 获取上次未下载完成的任务。
self.tasks = [self.downloader resumeUncompletedTasks];
```

下载并显示缩略图

- SDK 除了提供视频播放功能之外，还支持下载并显示缩略图。BDCloudSpriteManager提供缩略图相关接口。

> **注意：**
>
> - 缩略图的规则和获取需要配置MCP转码模板后转码才可使用。

```
// 1. 初始化雪碧图渲染配置并设置必要属性。
BDCloudSpriteRenderConfiguration *renderConfig = [BDCloudSpriteRenderConfiguration defaultConfiguration];
renderConfig.row = <雪碧图中缩略图的行数>;
renderConfig.column = <雪碧图中缩略图的列数> ;
renderConfig.thumbnailDuration = <每个缩略图的时间间隔>;

// 2. 初始化雪碧图下载配置并设置必要属性。
BDCloudSpriteDownloadConfiguration *downloadConfig = [BDCloudSpriteDownloadConfiguration defaultConfiguration];
downloadConfig.imageList = <雪碧图的URL列表>;
downloadConfig.maxThreadNum = <下载开启的做大线程数>;
downloadConfig.path = <下载地址>;

// 3. 初始化雪碧图渲染控制类,并添加显示的画布。
BDCloudSpriteManager *manager = [BDCloudSpriteManager initWithRenderConfig:renderConfig downloadConfig:downloadConfig];
manager.renderView.frame = <画布frame>;
[self.view addSubView:manager.renderView]
[manager startDownload:nil];

// 4. 用户收到下载成功回调后，根据时间戳显示缩略图。
[manager renderTime:<显示时间戳>];
```

网络视频加速

SDK 除了提供视频播放功能之外，还支持MP4格式的网络视频加速。BDCloudMediaSourceManager提供网络视频加速播放。包含以下功能：

- 视频预下载。
- 边播边存。

> **注意：**
>
> - 当前网络视频加速只支持mp4格式。

```
// 1. 预下载视频。
[[BDCloudMediaSourceManager sharedInstance] download:url size:downloadSize complete:^(BOOL succ, NSError *error) {
    NSLog(@"pre download succ = %d, error = %@",error);
}];

// 2. 开启边播边存
[BDCloudMediaSourceManager sharedInstance].autoWriteCache = YES;

// 3. 播放器设置网络代理。
self.player.proxy = [BDCloudMediaSourceManager sharedInstance];

// 4. 预下载完成后，和常规播放流程一样，设置url后开始播放，就会自动通过代理进行播放
[self.player prepareToPlay] ;
```

外挂字幕

通过下面的接口，可以给播放器添加外挂字幕，当前支持的字幕格式有srt/ass/ssa/webvtt。需要注意的是，该接口需要在收到播放器BDCloudMediaPlayerPlaybackIsPreparedToPlayNotification消息后使用。对于同一个播放器示例，同一时刻只允许添加一个外挂字幕，添加新的外挂字幕会自动代替旧的外挂字幕。

```
- (void)addExtSubtitleUrl:(NSString*)url;
```

外挂字幕成功读取后，会通过BDCloudMediaPlayerExtSubtitleOpenNotification消息通知，消息中使用BDCloudMediaPlayerExtSubtitleUrlKey记录相应的外挂字幕URL。

具体的字幕内容也会在BDCloudMediaPlayerTimedTextNotification消息中携带，这一点与内嵌字幕相同。

多音轨、多字幕切换

对于带有多个音轨、字幕轨的片源，可以通过下面的方式实现轨道的无缝切换。

1. 先利用下面的接口获取当前片源所有的轨道信息

```
(NSMutableArray *)getTrackInfo
```

返回的数组中每个元素为BDCloudMediaPlayerTrackInfo，在BDCloudMediaPlayerTrackInfo类中记录了每个轨道的媒体类型和语言信息。

2. 根据轨道信息数组的下标进行切换

利用下面的接口可以执行轨道的选择和反选，传入的参数即对应第一步获取的轨道信息数组下标。

```
(void)selectTrack:(int)trackId
(void)deselectTrack:(int)trackId
```

3. 查询当前选择的轨道

利用下面的接口可以获取当前选中的视频/音频/字幕轨，返回参数也对应第一步获取的轨道信息数组下标。

```
(int)getSelectedTrack:(BDCloudTrackType)trackType
```

4. 轨道变化回调

轨道的添加、切换都会通过BDCloudMediaPlayerTrackChangedNotification消息进行通知，该消息中BDCloudMediaPlayerTrackActionKey字段对应轨道添加或切换事件，BDCloudMediaPlayerTrackTypeKey字段对应轨道类型，BDCloudMediaPlayerTrackIdKey字段则对应轨道序号。

5. 无缝切换与有感切换

播放器默认使能轨道的无缝切换（切换过程中无缓冲状态），如果需要切换为有感切换，可以在prepareToPlay之前设置下面的选项来实现

```
[self.player setOptionIntValue:0
                        forKey:@"enable-smooth-track-change"
                    ofCategory:BDCloudMediaPlayerOptionCategoryPlayer];
```

AV1解码支持

目前仅在播放器SDK全媒体版中，集成了单独的AV1软件解码器。

如果使用播放器SDK流媒体版播放AV1片源，则通过`BDCloudMediaPlayerPlaybackDidFinishNotification`通知错误，错误码为10003，含义为解码器不支持当前格式。相关处理代码可以参考Demo工程中的`PlayerViewModel.m`文件。

常见错误码含义

常见错误码分为播放内核错误、鉴权错误、其他错误三类，具体定义和含义解释可以参考Demo工程`PlayerViewModel.m`中`getReadableErrorMsg`方法的实现。

高级版功能接入

全景声功能接入

接入准备

- 接入全景声功能，需要使用播放器SDK高级版，并申请高级版License。
- 在BDCloudMediaPlayer.framework中包含有音效配置文件WANOS_*.txt，将这些配置文件导入到你的APP中，并设置到Xcode的Copy Bundle Resources选项中。SDK会直接在mainBundle下寻找这些配置文件。
- 在vendor目录下有libauthcheck.a和libwanosdecoder.a两个静态库，将它们导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了全景声（WANOS）音频格式的解码和音效处理能力。其中解码能力无需调用任何接口，由播放内核原生支持。音效处理能力由专门的音效接口提供，既可对全景声（WANOS）格式进行音效处理，也可以对AAC、MP3等常规音频格式进行处理，优化听感。 当前支持的音效列表如下：

音效名称	效果说明
扬声器原声模式	原声，保留多声道听感
扬声器电影模式	使用扬声器虚拟环绕技术，增加声场宽度，使声场以及某些声像不仅仅局限于两个喇叭之间，而能扩展至两个扬声器外侧，提高声音的沉浸感
扬声器音乐模式	音乐相对于电影来说，更需要注重声音的音质，此模式采用最佳的频率响应，不加任何环绕处理，增强了语音的清晰度，使音乐声音更加自然
耳机原声模式	原声，保留多声道听感
耳机电影模式	使用耳机端的虚拟环绕技术，扩展声音的宽度，提高沉浸感，同时在一定程度上减小头中效应
耳机音乐模式	采用最佳的频率响应，不加任何环绕处理，增强了语音的清晰度，使音乐声音更加自然
耳机全景环绕模式	采用动态增强算法，配合科学的滤波处理，提高声音动态感，提升可玩性；让声音包围双耳，在一定程度上较小头中效应

Demo体验

接口说明 在BDCloudMediaPlayer中定义了音效类型枚举

```
typedef NS_ENUM(NSInteger, BDCloudMediaPlayerAudioEffect) {
    // 关闭音效
    BDCloudMediaPlayerAudioEffectClose  = 0,
    // 扬声器原声模式
    BDCloudMediaPlayerAudioEffectSpeakerOriginal,
    // 扬声器音乐模式
    BDCloudMediaPlayerAudioEffectSpeakerMusic,
    // 扬声器电影模式
    BDCloudMediaPlayerAudioEffectSpeakerMovie,
    // 耳机原声模式
    BDCloudMediaPlayerAudioEffectEarphoneOriginal,
    // 耳机音乐模式
    BDCloudMediaPlayerAudioEffectEarphoneMusic,
    // 耳机电影模式
    BDCloudMediaPlayerAudioEffectEarphoneMovie,
    // 耳机全景环绕模式
    BDCloudMediaPlayerAudioEffectEarphoneSurround
};
```

BDCloudMediaPlayerController提供如下的音效设置接口，在播放过程中传入不同的音效枚举类型即可实现音效处理的切换，接口会对SDK有效性和证书有效性做校验，如不符合高级版SDK要求，会返回错误，错误码定义于BDCloudAVAuthErrorCode。 在Demo中也对此接口的使用做了展示，可以参考。

```
- (NSInteger)setAudioEffect:(BDCloudMediaPlayerAudioEffect)audioEffect;
```

HDR功能接入

接入准备

- 接入HDR功能，需要使用播放器SDK高级版，并申请高级版License。
- 在frameworks目录下有BDCloudHdrKit.framewok，在vendor目录下有libGRF.a，将它们导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了HDR视频的解码和渲染能力，能够让HDR视频在高低端机型上都得到正确的色彩呈现。其中解码能力由播放内核原生支持，渲染能力由BDCloudHdrKit组件提供，请确保该组件已集成到你的App中。

SDK当前支持的HDR标准有：HDR10、HLG、HDR Vivid。

Demo体验

快速开始 目前提供两种使用方法

- 简单方法：直接使用BDCloudMediaPlayerController提供的新接口，一行代码开启HDR渲染支持
- 高级自定义方法：使用BDCloudHdrKit提供的底层接口，由接入方自行控制渲染逻辑

建议优先使用简单方法。

简单方法

```
 // 给播放器实例传入播放URL
self.player.contentString = url;
// 使能HDR渲染
[self.player toggleEnableHdr:YES];
// 准备播放
[self.player prepareToPlay];
```

高级自定义方法

1. 初始化和参数设置

```
 BDCloudHdrKitRender *hdrKit
// 创建实例
_hdrKit = [[BDCloudHdrKitRender alloc] init];
// 用licenseId和绘制hdr内容的CAEAGLLayer进行hdrkit的初始化
[_hdrKit initContextwithLicense:_hdrLicenseId withLayer:(__bridge void *)(self.layer)];
// 设置输入视频的HDR标准，默认为HDR10
[_hdrKit setContentType:_hdrContentType];
// 设置输入视频画面宽高
[_hdrKit setInputVideoSize:inputw withHeight:inputh];
// 设置输入视频旋转角度
[_hdrKit setInputVideoRotate:_videoRotate];
// 设置缩放模式
[_hdrKit setScaleType:HDR_SCALE_AspectFit];
// 设置渲染输出画面宽高
int width = [[UIScreen mainScreen] bounds].size.width * _scaleFactor;
int height = [[UIScreen mainScreen] bounds].size.height * _scaleFactor;
[_hdrKit setOutputSize:width withHeight:height];
```

2. 开始渲染

```
 [_hdrKit startRender];
```

3. 传入HDR元数据

```
 // 传入HDR静态元数据
if (_hdrStaticMeta != NULL && _isStaticMetaUpdated) {
    [_hdrKit setHdrStaticMetadata:_hdrStaticMetaType withData:(unsigned char*)[_hdrStaticMeta bytes] withLength:[_hdrStaticMeta length]];
    _isStaticMetaUpdated = FALSE;
}
// 传入HDR Vivid动态元数据
if (_hdrVividMeta != NULL) {
    [_hdrKit setVividMetadata:(unsigned char*)[_hdrVividMeta bytes] withLength:[_hdrVividMeta length]];
}
```

4. 更新视频帧数据

```
 // 输入未经处理的CVPixelBufferRef yuv数据(nv21格式)，SDK将对此yuv数据进行处理后渲染到CAEAGLLayer上
[_hdrKit updateInputBuffer:pixel_buffer];
```

5. 停止渲染和释放

```
 [_hdrKit stopRender];
[_hdrKit clearContext];
```

接口说明

BDCloudMediaPlayerController | 接口名 | 说明 | |----------|----------| | - (int)toggleEnableHdr:(BOOL)isEnable | 设置是否开启HDR渲染，成功开启返回0，失败返回-1 |

BDCloudHdrKitRender | 接口名 | 说明 | |----------|----------| | - (instancetype)initContext:(NSString*)licenseID withLayer:(void*)layer | 初始化
licenseID：通过百度智能云控制台申请的licenseID
layer：绘制hdr内容的CAEAGLLayer | | - (void)setInputVideoSize:(NSNumber*)width withHeight:(NSNumber*)height | 设置输入视频宽高 | | - (void)setInputVideoRotate:(NSNumber*)rotate | 设置输入视频旋转角度 | | - (void)setOutputSize:(NSNumber*)width withHeight:(NSNumber*)height | 设置渲染画布宽高
width：输出宽
height：输出高 | | - (void)setScaleType:(HdrScalingMode)type | 设置渲染缩放类型 | | - (void)setContentType:(HdrContentType)type | 设置HDR内容类型 | | - (void)updateInputBuffer:(CVPixelBufferRef)input | 输入未经处理的yuv数据，SDK将对此yuv数据进行处理后渲染到CAEAGLLayer上
input nv21格式的yuv数据 | | - (void)setHdrStaticMetadata:(int)type withData:(unsigned char*)data withLength:(size_t)data_length | 传入HDR静态元数据
type：SEI_MASTERING_DISPLAY_COLOUR_VOLUME(137)或SEI_CONTENT_LIGHT_LEVEL(144)
data：指向元数据的指针
data_length：元数据长度 | | - (void)setVividMetadata:(unsigned char*)data withLength:(size_t)data_length | 传入HDR Vivid动态元数据
data：指向元数据的指针
data_length：元数据长度 | | - (void)startRender | 开始渲染 | | - (void)stopRender | 停止渲染 | | - (void)clearContext | 销毁实例 |

超低延时直播功能接入

接入准备

- 接入超低延时直播功能，需要使用播放器SDK高级版，并申请高级版License。
- 在frameworks目录下有BaiduRtcPlayerGeneral.framewok，将它导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了超低延时直播流的播放能力，该能力由BaiduRtcPlayerGeneral组件提供，请确保该组件已集成到你的App中。 SDK当前支持的音视频编码格式如下：

- 视频：H.264/HEVC，支持B帧
- 音频：AAC

Demo体验

快速开始

1. 播放器参数配置

```
 BaiduRtcPlayerParameter *param = [BaiduRtcPlayerParameter defaultParameter];
// 设置信令服务地址
param.signalServer = @"http://test-pl-central.bigenemy.cn/brtc/v3/pullstream";
// 设置视频解码类型，默认使用H.264
param.videoCodectype = _boCodec265Type ? CODEC_H264 : CODEC_H265;
// 设置是否开启B帧支持，默认不开启
param.videoBFrame = _boVideoBFrame;
```

2. 创建播放器对象及播放器初始化

```
 // 创建播放器对象，需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台（https://console.bce.baidu.com/bvc/#/bvc/player-license/list）查看。delegate详细参考回调事件
self._bdPlayer = [[BaiduRtcPlayer alloc] initWithParameter:param licenseId:LICENSE_ID delegate:self];
// 外部获取播放器view
[self.mainView addSubview:_bdPlayer.view];
// 开始播放 是否支持异步播放还是同步播放。如果鉴权失败，会抛出异常
[_bdPlayer prepareToPlay:_streamUrl autoPlay:NO];
```

3. 播放控制

```
 // 暂停播放
- (void)pausePlay;
// 恢复播放
- (void)resumePlay;
// 停止播放，释放资源
- (void)stopPlay;
// 重置播放器
- (void)reset;
// 设置音量
- (void)setVoulme:(float)volume;
// 设置显示模式
- (void)setScalingMode:(BaiduRtcPlayerScalingMode)mode;
```

4. 播放回调事件

```
 // prepareToPlay回调事件，如果异步打需要在回调事件后调用startPlay
- (void)onPrepared:(BaiduRtcPlayer *)player;
// 首帧事件
- (void)onFirstVideoFrameRendered:(BaiduRtcPlayer *)player;
// 播放器分辨率改变
- (void)onResolutionChanged:(BaiduRtcPlayer *)player
                       size:(CGSize)size;
// 播放错误
- (void)onPlayerError:(BaiduRtcPlayer *)player
            errorCode:(BaiduRtcPlayerErrorCode)errorCode
                error:(nullable NSError *)error;
// sei回调信息
- (void)onPlayerReceivedSEI:(BaiduRtcPlayer *)player
                        sei:(NSDictionary *)sei;
```

5. 释放播放器

```
- (IBAction)onstopPlay:(id)sender {
    if (_bdPlayer) {
        [_bdPlayer stopPlay];
  _bdPlayer = nil;
    }
}
```

在播放器SDK Demo中对上述流程有详细的展示，可以参考。

接口说明 BaiduRtcPlayerParameter | 参数名 | 含义 | |----------|----------| | netStatus | 网络连接状态
ReachableNot：无网络连接
ReachableWiFi：WIFI网络连接
ReachableWWAN：移动网络连接 | | audioEnable | 使能音频播放，默认开启 | | videoEnable | 使能视频播放，默认开启 | | audioVolume | 音量，范围[0,1]，默认为1 | | signalServer | 信令服务URL | | palyer_url | 媒体流URL，格式为webrtc://domain/app/stream | | checkFrameDuration | 检查断流时间长度用户可以设置，默认10s | | videoCodectype | 视频解码格式，由业务侧确定是否为H265格式，默认为H264格式 | | videoBFrame | 是否开启B帧支持，默认不开启 |

BaiduRtcPlayerErrorCode | 错误码 | 含义 | |----------|----------| | BaiduRtcPlayerErrorInvalidUrl = 10000 | URL格式错误 | | BaiduRtcPlayerErrorIceFailed = 10001 | ICE连接错误 | | BaiduRtcPlayerErrorIceDisconnected = 10002 | ICE断开 | | BaiduRtcPlayerErrorConnection = 10003 | Peer连接创建失败 | | BaiduRtcPlayerErrorGetLocalSdpFailed = 10004 | 本地SDP获取失败 | | BaiduRtcPlayerErrorSetLocalSdpFailed = 10005 | 设置本地SDP失败 | | BaiduRtcPlayerErrorGetRemoteSdpFailed = 10006 | 远端SDP获取失败 | | BaiduRtcPlayerErrorSetRemoteSdpFailed = 10007 | 远端SDP设置失败 | | BaiduRtcPlayerErrorInvalidStatus = 10008 | 播放状态错误 | | BaiduRtcPlayerErrorNullVideoFrame = 10009 | 媒体流中断 一段时间未收媒体流 | | BaiduRtcPlayerErrorLoadLibaraies = 10010 | 库加载失败 |

BaiduRtcPlayerInfoCode | 事件码 | 含义 | |----------|----------| | BaiduRtcPlayerInfoRoomRender = 1000 | 开始远端渲染 | | BaiduRtcPlayerInfoIceConnected = 1001 | ICE连接成功 | | BaiduRtcPlayerInfoPeerConnectionClosed = 1002 | 对端连接关闭 | | BaiduRtcPlayerInfoStatsUpdated = 1003 | 媒体流信息更新 | | BaiduRtcPlayerInfoBufferingStart = 1004 | Buffering start事件 | | BaiduRtcPlayerInfoBufferingEnd = 1005 | Buffering end事件 | | BaiduRtcPlayerInfoIceDisconnected = 1006 | ICE连接断开 | | BaiduRtcPlayerInfoNoStremingDetected = 1007 | 没有检测到媒体流 | | BaiduRtcPlayerInfoPlayTimeStatistic = 1008
BaiduRtcPlayerInfoLocalSDPSetted = 1009
BaiduRtcPlayerInfoRemoteSDPRequested = 1010 | 启播各阶段耗时统计 |

BaiduRtcPlayerDelegate | 接口名 | 说明 | |----------|----------| | - (void)onPrepared:(BaiduRtcPlayer *)player elapse:(NSInteger)interval | 播放器准备就绪回调
interval：prepare耗时 | | - (void)onFirstVideoFrameRendered:(BaiduRtcPlayer *)player elapse:(NSInteger)interval | 首帧渲染回调（在主线程）
interval：首帧耗时 | | - (void)onPeriodRenderStatis:(BaiduRtcPlayer *)player frame:(NSInteger)frameRate | 周期性返回帧率统计 | | - (void)onResolutionChanged:(BaiduRtcPlayer *)player size:(CGSize)size | 分辨率变化回调（在主线程） | | - (void)onPlayerError:(BaiduRtcPlayer *)player errorCode:(BaiduRtcPlayerErrorCode)errorCode error:(nullableNSError *)error | 错误回调（在子线程） | | - (void)onPlayerStreamChanged:(BaiduRtcPlayer *)player hasVideo:(BOOL)hasVideo hasAudio:(BOOL)hasAudio | 通知远端媒体信息（在子线程） | | - (void)onPlayerStatusChanged:(BaiduRtcPlayer *)player status:(BaiduRtcPlayerStatus)status | 播放器状态回调（在子线程） | | - (void)onPlayerReceivedSEI:(BaiduRtcPlayer *)player sei:(NSData *)sei | SEI信息回调（在子线程）
sei：nal_type(1) + sei_type(1) + sei_size(n+1)+sei_payload(n*255+m) | | - (void)onPlayerAudioBufferingStart:(BaiduRtcPlayer *)player | 卡顿事件开始回调（在子线程） | | - (void)onPlayerAudioBufferingEnd:(BaiduRtcPlayer *)player jitter:(NSInteger)interval | 卡顿事件结束回调（在子线程） | | - (void)onPlayerInfo:(BaiduRtcPlayer *)player infoCode:(BaiduRtcPlayerInfoCode)infoCode obj:(NSObject *)obj | 播放器事件回调（在子线程） | | - (void)onPlayerStatisticsInfo:(BaiduRtcPlayer *)player stats:(NSArray *)statistics | RTC引擎状态信息统计
该callback返回当前rtc engine的一些参数和性能信息，如传输fps,码率，网络状况,cpu等信息给应用层 |

BaiduRtcPlayer

接口名	说明
+ (NSString *)version	获取版本号
+ (void)setVerbose:(BOOL)bOnVerbose	是否打开调试信息
- (instancetype)initWithParameter:(BaiduRtcPlayerParameter *)parameter licenseId:(NSString*)id delegate:(id<BaiduRtcPlayerDelegate>)delegate	使用初始化参数、licenseID和事件代理创建播放器对象
- (void)setVideoBFrame:(BOOL)enable	设置是否开启B帧支持
- (void)setCodecType:(CodecType) typeCodec	设置解码格式，默认CODEC_H264，如需开启HEVC支持，则选择CODEC_H265
- (void)prepareToPlay:(NSString*)url autoPlay:(BOOL)autoPlay	传入媒体流地址，准备播放 autoPlay控制是否在准备就绪后自动开始播放 媒体流URL格式为webrtc://domain/app/stream
- (void)startPlay	开始播放
- (void)pausePlay	暂停播放
- (void)resumePlay	从暂停状态恢复播放
- (void)stopPlay	停止播放
- (void)reset	重置播放器参数
- (void)setVoulme:(float)volume	设置音量，范围[0,1]，默认为1
- (void)setScalingMode:(BaiduRtcPlayerScalingMode)mode	设置画面显示模式
- (void)setSingalMode:(BaiduRtcPlayerSingalMode)mode	设置信令模式 BaiduRtcPlayerSingalModeOverHttp：HTTP模式
- (void)addPlayerDelegate:(id)delegate	添加播放器事件代理
- (void)removePlayerDelegate:(id)delegate	移除播放器事件代理
- (void)setSignalServer:(NSString *)signalServer	设置信令服务URL
- (void)setNetStatus:(BRtcNetworkStatus)netStatus	设置当前网络连接状态 ReachableNot：无网络连接 ReachableWiFi：WIFI网络连接 ReachableWWAN: 移动网络连接
- (BOOL)resetAudioDeviceConfigure	重新设置音频参数

投屏功能接入

接入准备

- 接入投屏功能，需要使用播放器SDK高级版，并申请高级版License。
- 在frameworks目录下有ProjectionEngine.framewok，将它导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了DLNA投屏能力，可以将手机端的视频内容推送到大屏端进行播放，并且支持在手机端远程控制大屏端的媒体播放。该能力由ProjectionEngine组件提供，请确保该组件已集成到你的App中。

快速开始 1.初始化ProjectionEngine组件传入LicenseID

```
 // 需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台查看
[[ProjectionEngine shareManager] setLicenseID:LICENSE_ID];
```

2.实现设备发现回调，开始搜索投屏设备，当找到设备时，会利用回调进行通知

```
 // 设备发现回调，可以在这里将发现到的接收端信息保存起来
###### pragma mark DeviceListenerDelegate
- (void)OnAddDevice:(DeviceUpnp *) device {
    NSLog(@"OnAddDevice == %@", device.friendlyName);
    if (![self.dataArray containsObject:device]) {
        [self.dataArray addObject:device];
    }
}

- (void)OnRemoveDevice:(DeviceUpnp *) device {
    if ([self.dataArray containsObject:device]) {
        [self.dataArray removeObject:device];
    }
}
[[ProjectionEngine shareManager] searchDeviceBegin:self];
```

3.选择设备并发起投屏

```
 // 从之前保存的接收端信息中选出一个作为发起投屏的目标
DeviceUpnp *model = self.dataArray[indexPath.row];
// 分配渲染控制器， playId作为渲染控制器标识，后续的媒体播放控制都基于它完成
self.playId = [[ProjectionEngine shareManager] distributeProjectionRender:model delegate:self];
// 设置接收端要播放的媒体url
NSString *urlStr = [self.delegate getCurrentPlayerUrl];
[[ProjectionEngine shareManager] setAVTransportURL:self.playId url:urlStr];
```

4.控制接收端的媒体播放

```
 // 播放
[[ProjectionEngine shareManager] play:self.playId speed:1];
// 暂停
[[ProjectionEngine shareManager] pause:self.playId];
// seek, 单位为秒
[[ProjectionEngine shareManager] seek:self.playId realTime:self.slider.value];
// 停止
[[ProjectionEngine shareManager] stop:self.playId];
```

5.停止搜索设备

```
 [[ProjectionEngine shareManager] searchDeviceEnd];
```

6.结束投屏并释放渲染控制器

```
 [[ProjectionEngine shareManager] stop:self.playId];
[[ProjectionEngine shareManager] releaseProjectionRender:self.playId];
```

在播放器SDK Demo中的PlayerControlVC类对上述流程有详细的展示，可以参考。

关于组播权限申请的说明 投屏设备的发现依赖组播消息的发送。自iOS14.5开始，苹果对APP的组播权限增加了控制，需要开发者单独向苹果申请组播权限。申请方式参考苹果官网文档，通过申请后，在APP的.entitlements配置文件中添加如下内容即可。

```
 <key>com.apple.developer.networking.multicast</key>
<true/>
```

接口说明 ProjectionEngine类 | 接口名 | 描述 | | ---------- | ---------- | | + (instancetype)shareManager | 获取投屏引擎单例 | | - (void) setLicenseID:(NSString*)licenseId | 传入LicenseID | | - (NSString*) getVersion | 获取投屏引擎版本号 | | - (void) searchDeviceBegin:(id) deviceObserverDelegate | 开始搜索设备，鉴权失败的情况下会抛出异常 | | - (void) searchDeviceEnd | 停止搜索设备 | | - (int) distributeProjectionRender:(DeviceUpnp*) devicedelegate:(id)renderListener | 分配渲染控制器，返回值作为渲染控制器标识 | | - (void)releaseProjectionRender:(int)renderIndex | 释放渲染控制器 | | - (void)setAVTransportURL:(int)renderIndexurl:(NSString*)url | 设置接收端播放的URL | | - (void)play:(int)renderIndex speed:(int)realtime | 开始接收端播放，当前speed仅支持传入1 | | - (void)pause:(int)renderIndex | 暂停接收端播放 | | - (void)stop:(int)renderIndex | 停止接收端播放 | | - (void)seek:(int)renderIndex realTime:(float)realtime | 控制接收端进度seek，单位为秒 | | - (int)getDuration:(int)renderIndex | 获取媒体流时长 | | - (int)getCurPos:(int)renderIndex | 获取当前播放位置 |

DeviceUpnp类 | 属性名 | 描述 | | ---------- | -------- | | uuid | 获取接收端设备ID | | friendlyName | 获取接收端设备名 |

事件回调	含义
- (void)OnAddDevice:(DeviceUpnp *) device	发现设备
- (void)OnRemoveDevice:(DeviceUpnp *) device	移除设备

VR功能接入

接入准备

- 接入VR全景视频播放功能，需要使用播放器SDK高级版，并申请高级版License。
- 在frameworks目录下有BDCloudVRRender.framewok，将它导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍

在高级版SDK中，提供了VR全景视频的渲能力，并且可通过陀螺仪进行视角变换。该能力由BDCloudVRRender组件提供，请确保该组件已集成到你的App中。

快速开始

1. 配置VR渲染参数

```
 // 默认参数实例。
self.vrConfig = [BDCloudVRConfiguration defaultConfig];
// 配置VR投影类型。
[self.vrConfig setProjectionMode:<projectionMode>];
// 配置VR渲染模式。
[self.vrConfig setDisplayMode:<displayMode>];
// 配置VR视角交互模式。
[self.vrConfig setInteractiveMode:<interactiveMode>];
// 配置是否支持捏合手势。
[self.vrConfig setPinchEnabled:<pinchEnabled>];
```

2. 配置VR视频输入源，传入播放器渲染视图

```
[self.vrConfig setProviderBDCloudMediaPlayerView:_player.view
                            viaHardwareAccelerate:[_player viaHardwareAccelerate]];
```

3. 配置VR渲染视图的父视图

```
[self.vrConfig setRenderOn:_player.view];
```

4. 初始化VR渲染控制类并开始VR渲染

```
 // 需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台查看
self.vrManager = [BDCloudVRRenderControl renderWithLicenseId:LICENSE_ID Config:self.vrConfig];
[self.vrManager resume];
```

5. 停止渲染

```
[self.vrManager stop]
```

在播放器SDK Demo中的PlayerViewModel类对上述流程有详细的展示，可以参考。

接口说明 BDCloudVRRenderControl | 接口名 | 说明 | | ---------- | ---------- | | + (BDCloudVRRenderControl *)renderWithLicenseId:(NSString*)licenseID Config:(BDCloudVRConfiguration *)config | 根据VR渲染配置初始化渲染器
licenseID：通过百度智能云控制台申请的licenseID
config VR渲染配置 | | - (void)switchInteractiveMode:(BDCVRModeInteractive)interactiveMode | 切换交互方式 | | - (BDCVRModeInteractive)getInteractiveMode | 获取当前交互方式 | | - (void)switchDisplayMode:(BDCVRModeDisplay)displayMode | 切换渲染方式 | | - (BDCVRModeDisplay)getDisplayMode | 获取当前渲染方式 | | - (void)switchProjectionMode:(BDCVRModeProjection)projectionMode | 切换投影类型 | | - (BDCVRModeProjection)getProjectionMode | 获取当前投影类型 | | - (void)pause | 暂停VR渲染 | | - (void)resume | 开始VR渲染 | | - (void)stop | 停止VR渲染 | | - (void) updateOrientation:(UIInterfaceOrientation)orient | 更新陀螺仪方向 |

BDCloudVRConfiguration | 接口名 | 说明 | | ---------- | ---------- | | + (BDCloudVRConfiguration *)defaultConfig | VR渲染默认配置实例化方法 | | - (void)setProviderBDCloudMediaPlayerView:(UIView *)view viaHardwareAccelerate:(BOOL)viaHardwareAccelerate | 配置VR图像输入源为百度智能云播放器渲染视图。
view：百度智能云播放器渲染视图。
viaHardwareAccelerate：百度智能云播放器视频解码方式 | | - (void)setInteractiveMode:(BDCVRModeInteractive)interactiveMode | 配置VR渲染交互模式 | | - (void)setDisplayMode:(BDCVRModeDisplay)displayMode | 配置VR渲染模式 | | - (void)setProjectionMode:(BDCVRModeProjection)projectionMode | 配置VR投影类型 | | - (void)setPinchEnabled:(BOOL)pinch | 配置是否允许捏合手势 | | - (void)setRenderOn:(UIView *)parentView | 配置VR渲染视图的父视图 |

绿幕抠图功能接入

接入准备

- 接入绿幕抠图功能，需要使用播放器SDK高级版，并申请高级版License。
- 在frameworks目录下有BDCloudVirtualLiveKit.framewok，在vendor目录下有libGRF.a，将它们导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了高精度、高性能的绿幕抠图能力，可实现对绿色或其他纯色背景的自动识别和抠像。该能力由BDCloudVirtualLiveKit组件提供，请确保该组件已集成到你的APP中。 SDK当前支持纹理ID输入、纹理ID输出。

Demo体验 **快速开始**

1. 初始化

```
 BDCloudVirtualLiveRender *vlRender
// 初始化，需要传入您申请的高级版证书LicenseID，ID可以在百度云控制台查看
self.vlRender = [[BDCloudVirtualLiveRender alloc] initWithLicenseId:LICENSE_ID];
```

2. 设置输入和输出纹理ID

```
 // 设置输入纹理ID, 该ID对应的是原始绿幕背景视频，SDK处理后的内容将渲染到指定的输出纹理ID上
[self.vlRender prepareWithInputTexId:self.fgInputTexId];
// 设置输出纹理ID，该ID对应的是经过处理后的绿幕背景视频，此时绿幕或其他纯色背景被替换为透明的通道
[self.vlRender setOutputTexId:self.fgOutputTexId];
```

3. 渲染参数设置

```
 // 设置输入画面宽高
[self.vlRender setInputWidth:_viewW AndHeight:_viewH];
// 设置渲染输出宽高
[self.vlRender setOutputWidth:_viewW AndHeight:_viewH];
```

4. 抠像色值设置 默认情况下，SDK可以自动识别背景色值进行抠像。使用方也可以手动指定抠像色值，传入后，将按照指定色值进行抠像

```
 // 设置是否要自动识别背景色值，默认自动
[self.vlRender setAutoCalculateKeyColor:NO];
// 设置抠像色值，范围[0, 255]
float rgba[4] = {0, 255, 0, 0};
[self.vlRender setKeyColor:rgba];
```

5. 开始渲染

```
 // 开始基于输入纹理ID的渲染，调用一次，渲染一帧
[self.vlRender drawFrameFromTexId];
```

6. 释放

```
 // 销毁，如果要重新使用，则需要重新init
[self.vlRender destroy];
```

在播放器SDK Demo中的VirtualLiveViewModel类对上述流程有详细的展示，可以参考。 **接口说明 BDCloudVirtualLiveRender** | 接口名 | 说明 | | ---------- | ---------- | | - (instancetype)initWithLicenseId:(NSString*)licenseID | 初始化 licenseID：通过百度智能云控制台申请的licenseID | | - (void)setInputWidth:(int)w AndHeight:(int)h | 设置输入视频宽高。必须在prepareWithInputTexId之后调用，否则抛出异常 | | - (void)setOutputWidth:(int)w AndHeight:(int)h | 设置渲染输出宽高。必须在prepareWithInputTexId之后调用，否则抛出异常 | | - (void)setAutoCalculateKeyColor:(Boolean)enable | 设置是否要自动识别背景色值，默认自动 | | - (void)setKeyColor:(float[])rgba | 设置抠像色值，传入后，将按照指定色值进行抠像，否则自动识别背景色值并抠像。范围[0, 255] | | - (void)prepareWithInputTexId:(int)texid | 设置输入纹理ID, 该ID对应的是未处理的视频，处理后的内容将渲染到指定的输出纹理ID上。必须在initWithLicenseId之后调用，否则抛出异常 | | - (void)setOutputTexId:(int)texid | 设置输出纹理ID，输出纹理中绿幕部分变为透明。必须在prepareWithInputTexId之后调用，否则抛出异常 | | - (void)drawFrameFromTexId | 开始基于输入纹理ID的渲染，调用一次，渲染一帧。必须在prepareWithInputTexId之后调用，否则抛出异常 | | - (void)destroy | 销毁 |

端上超分功能接入

接入准备

- 接入端上超分功能，需要使用播放器SDK高级版（也可以单独接入端上超分SDK），并申请高级版License。
- 在frameworks目录下有BDCloudSrKit.framewok和opencv2.framework，将它们导入到你的项目中，并设置到Xcode的Link Binary With Libraries选项中

功能介绍 在高级版SDK中，提供了端上超分能力，利用端侧推理能力，实现对低分辨率画面的清晰度提升、噪声和块效应去除，适用于视频播放场景和RTC通话场景。该能力由BDCloudSrKit和opencv2组件提供，请确保相关组件已集成到你的APP中。

SDK能力如下表所示

能力	说明
支持的超分倍数	固定2倍
支持的输入分辨率	不限制输入分辨率
支持的输入-输出格式	CVPixelBuffer -> CVPixelBuffer，支持YUV420P和NV12像素格式
模型策略	SDK可以根据目标分辨率&帧率和设备性能自动选择合适的模型，保证实时处理帧率。如果设备性能太差或输入分辨率太大，无法达到实时帧率，SDK也可以给出预期的处理帧率，接入方可以自由决定是否应用超分

Demo体验

快速开始

1. 初始化

```
 BDCloudSrRender *srKit
// 初始化，需要传入您申请的高级版证书LicenseID，ID可以在百度云控制台查看
srKit = [[BDCloudSrRender alloc] initWithLicenseId:LICENSE_ID];
```

2. 设置目标帧率和模型选择策略

```
 [_srKit setTargetFps:fps AndStategy:SrModelStrategy::STRATEGY_SPEED_FIRST];
```

3. 对输入CVPixelBuffer进行处理，处理结果保存到输出CVPixelBuffer中，调用一次，处理一帧

```
 [_srKit processInput:inBuffer ofFormat:SrInputPixelFormat::FMT_NV12 withOutput:outBuffer];
```

4. 释放

```
 // 销毁，如果要重新使用，则需要重新init
[_srKit destroy];
```

最佳实践 在调用processInput之前，可以依据输入分辨率和帧率获取SDK的预期超分处理帧率，可以根据预期结果，再决定是否应用超分，避免设备性能不足或输入分辨率过高导致的渲染卡顿，示例代码如下

```
 float expectedFps = [_srkit probeExpectedFpsForWidth:frame.width Height:frame.height TargetFps:_srTargetFps];
if (expectedFps > 0 && expectedFps < _srTargetFps) {
    return;
} else {
    [_srKit processInput:inBuffer ofFormat:SrInputPixelFormat::FMT_NV12 withOutput:outBuffer];
}
```

接口说明 BDCloudSrRender | 通知 | 说明 | | ---------- | ---------- | | BDCloudSREngineInitedNotification | 超分内核初始化完成通知 | | BDCloudSRProcessedFrameUpdateNotification | 超分单帧处理完成通知，包含以下key

BDCloudSRProcessedFrameWidthKey：超分后的宽度

BDCloudSRProcessedFrameHeightKey：超分后的高度

BDCloudSRProcessedFrameCostTimeKey：单帧处理耗时，单位毫秒 |

接口名	说明
- (instancetype)initWithLicenseId:(NSString*)licenseID	初始化 licenseID：通过百度智能云控制台申请的licenseID
- (void)destroy	销毁
- (void)setTargetFps:(float)fps AndStategy:(SrModelStrategy)strategy	设置目标帧率和模型选择策略，SDK会依据输入分辨率和目标帧率，自动选择最合适的模型。 可选的模型策略包括 SrModelStrategy::STRATEGY_SPEED_FIRST 速度优先 SrModelStrategy::STRATEGY_QUALITY_FIRST 质量优先
- (BOOL)processInput:(CVPixelBufferRef)inputBuffer ofFormat:(SrInputPixelFormat)fmt withOutput:(CVPixelBufferRef)outBuffer	超分处理 fmt：输入像素格式，可选值包括 SrInputPixelFormat::FMT_NV12，对应kCVPixelFormatType_420YpCbCr8BiPlanarFullRange或kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange SrInputPixelFormat::FMT_YUV420P，对应kCVPixelFormatType_420YpCbCr8Planar或kCVPixelFormatType_420YpCbCr8PlanarFullRange 返回YES代表超分处理成功，返回NO代表超分处理未成功，此时SDK没有向outBuffer写入任何数据。 在调用processInput方法后超分内核才会初始化并抛出BDCloudSREngineInitedNotification消息。
- (float)probeExpectedFpsForWidth:(int)width Height:(int)height TargetFps:(float)fps Strategy:(SrModelStrategy)strategy;	获取模型在指定分辨率、目标帧率、模型策略下的预估实际帧率，接入方可以根据预估结果决定是否开启超分。 如果在当前设备上超分内核从来没有初始化过（收到BDCloudSREngineInitedNotification通知），那么无法获得全局设备算力情况，返回值为0。

接口速查

为了便于用户查询，提供播放器、HLS下载的相关接口列表。P表示属性, F表示方法。

播放器

BDCloudMediaPlayerAuth

负责AccessKey鉴权：

序号	名称	类型	参数	返回值	描述
1	sharedInstance	F	-	单例实例	获取单例
2	delegate	P	-	-	设置鉴权代理，一般是AppDelegate
3	authenticateLicenseID:completionHandler	F	字符串 LicenseID	-	设置LicenseID，发起异步鉴权操作

BDCloudMediaPlayerAuthDelegate

AccessKey鉴权回调事件：

序号	名称	类型	参数	返回值	描述
1	authStart	F	-	-	鉴权开始
2	authEnd	F	NSError 错误	-	鉴权结束，错误为空值表示鉴权成功

BDCloudMediaPlayerBitrateItem

抽象多码率HLS视频的一个码率item：

序号	名称	类型	参数	返回值	描述
1	resolution	P	-	-	分辨率，单位像素
2	bitrate	P	-	-	码率，单位 bps

BDCloudMediaPlayerMeidaItem

抽象无缝切换视频的一个视频item，主要用于视频的无缝切换：

序号	名称	类型	参数	返回值	描述
1	itemFromString	F	NSString 视频信息	-	初始化视频信息
2	resolution	P	-	-	分辨率，单位像素
3	bitrate	P	-	-	码率，单位 bps

BDCloudMediaPlayerController

主要接口

序号	名称	类型	参数	返回值	描述
1	view	P	-	-	负责渲染视频画面，需要添加到UI上并设置布局信息。
2	contentURL	P	-	-	视频播放地址(NSURL形式)
3	contentString	P	-	-	视频播放地址(NSString形式)
4	isPreparedToPlay	P	-	-	是否已准备好播放
5	playbackState	P	-	-	播放器当前播放状态
6	loadState	P	-	-	播放器当前加载状态
7	scalingMode	P	-	-	画面缩放模式
8	initialPlaybackTime	P	-	-	设置或获取视频起始播放位置，单位秒
9	currentPlaybackTime	P	-	-	设置或获取设置视频播放，单位秒
10	naturalSize	P	-	-	获取视频的分辨率
11	videoDAR	P	-	-	自定义视频长宽比
12	duration	P	-	-	获取视频的总时长，单位秒
13	playableDuration	P	-	-	获取视频当前可播放时长，单位秒
14	shouldAutoplay	P	-	-	设置或获取是否允许视频自动播放
15	shouldAutostop	P	-	-	设置或获取是否允许视频自动停止播放
16	playbackRate	P	-	-	设置或获取倍速播放速率 范围[0.0, 4.0]
17	playbackVolume	P	-	-	视频播放音量 范围[0.0, 1.0]
18	downloadSpeed	P	-	-	获取在线视频下载速度，单位Bps
19	clearCanvasWhenReset	P	-	-	设置或获取播放器Reset是否清屏，默认为NO
20	videoBitrate	P	-	-	当前视频编码器输出的比特率，单位 kbps
21	audioBitrate	P	-	-	当前音频编码器输出的比特率，单位 kbps
22	initWithContentURL:	F	NSURL	实例	传入视频播放地址(NSURL形式)初始化播放器
23	initWithContentString:	F	NSString	实例	传入视频播放地址(NSString形式)初始化播放器
24	prepareToPlay	F	-	-	开始异步初始化视频
25	play	F	-	-	播放视频，isPreparedToPlay为YES时，SDK自动调用
26	pause	F	-	-	暂停播放
27	stop	F	-	-	停止播放，停止后需要重新初始化视频
28	reset	F	-	-	重置播放器，重置后可以设置其他的视频URL
29	seek:	F	NSTimeInterval	-	快速定位播放位置
30	isPlaying	F	-	BOOL	是否正在播放视频
31	setPauseInBackground:	F	BOOL	-	App切后台后，是否自动暂停播放
32	enableLooping:	F	BOOL	-	是否循环播放
33	enableAutoIdleTimerDisabled:	F	BOOL	-	设置播放器自动控制屏幕长亮
34	setMaxCacheSizeInBytes:	F	NSUInteger	-	设置缓冲区大小，单位字节
35	setCachePauseTime:	F	NSTimeInterval	-	设置缓冲区最少时长。单位秒
36	setFirstBufferingTime:	F	NSTimeInterval	-	设置起播放时最大缓冲时长。单位秒
37	toggleFrameChasing:	F	BOOL	-	是否开启追帧播放
38	setTimeoutInUs:	F	int	-	设置连接建立和数据下载过程中的超时时长。单位微秒
39	thumbnailImageAtCurrentTime	F	-	UIImage	截图
40	proxy	p	-	-	播放器网络代理
41	setMaxCacheDurationInSeconds:	F	NSTimeInterval	-	设置缓冲区的时长。单位秒
42	playType	p	视频播放类型	-	设置视频播放类型。默认点播。
43	userDefine	p	NSDictionary	-	设置用户自定义信息
44	tcpLoadSizeBlock	p	-	-	设置TCP数据下载回调
45	startRecording:videoCompressionSettings:audioCompressionSettings:recordCallBack:	F	1.NSString（文件录制地址）2.NSDictionary （视频录制配置，参考接口文件）3.NSDictionary （音频录制配置，参考接口文件）4.void(^)(int status)（录制状态回调函数）	-	开启视频录制文件
46	stopRecord:	F	void(^)(int status)（录制状态回调函数）	-	停止视频录制文件
47	setEnableDecodeSubtitle:	F	BOOL	-	设置是否解码字幕Track。 默认不解码。如果打开此功能，字幕会通过 BDCloudMediaPlayerTimedTextNotification 通知给到用户
48	addExtSubtitleUrl:	F	NSString	-	添加外挂字幕，在播放器prepared之后调用
49	selectTrack:	F	int	-	选择轨道，传入参数对应getTrackInfo返回数组的下标
50	deselectTrack:	F	int	-	反选轨道，传入参数对应getTrackInfo返回数组的下标
51	getSelectedTrack:	F	BDCloudTrackType	int	获取当前选中的视频/音频/字幕轨
52	getTrackInfo:	F	-	NSMutableArray	获取所有音视频track的信息
53	isPipRenderMode:	P	-	-	设置或获取是否进入画中画渲染模式
54	toggleEnableHdr:	F	BOOL	-	设置是否开启HDR渲染
55	toggleEnableSr:	F	BOOL	-	设置是否开启超分处理

硬件加速

序号	名称	类型	参数	返回值	描述
1	setVideoDecodeMode:	F	解码模式	-	设置解码模式
2	viaHardwareAccelerate	F	-	BOOL	获取是否当前视频启用了硬件加速

多码率快速切换（HLS）

序号	名称	类型	参数	返回值	描述
1	getSupportedBitrates	F	-	多码率列表	获取多码率列表
2	bitrateIndex	F	-	码率索引	获取当前码率索引
3	setBitrateIndex:	F	码率索引	-	设置当前码率索引

多码率无缝切换（HLS,MP4等主流媒体格式）

序号	名称	类型	参数	返回值	描述
1	setMediaItemsInputType:	F	视频输入格式	-	设置视频输入格式
2	setMediaItems:	F	视频地址列表	-	设置视频地址列表
3	getMediaItems:	F	-	视频列表信息	获取视频列表的视频信息
4	mediaItemIndex	F	-	视频索引	获取当前播放视频索引
5	setMediaItemIndex:	F	视频索引	BOOL	切换视频

类方法

序号	名称	类型	参数	返回值	描述
1	+ setLogReport:	F	BOOL	-	设置是否开启日志
2	+ setLogLevel:	F	日志级别	-	设置日志级别
3	+ setLogPath:	F	NSString	-	设置日志输出路径
4	+ logPath	F	-	NSString	获取日志路径
5	+ clearDiskLog	F	-	-	清除当前路径日志
6	+ getSDKVersion	F	-	SDK版本	获取SDK版本

DRM

序号	名称	类型	参数	返回值	描述
1	setPlayerID:key:	F	playerId playerKey	-	设置自定义的playerId、playerKey
2	setToken:	F	token	-	设置加密视频的临时token

水印

序号	名称	类型	参数	返回值	描述
1	setWatermark:	F	UIImage	-	设置水印图片
2	setWatermarkPosition:	F	CGPoint位置	-	设置水印位置，布局变化时可能需要重新设置

选项

序号	名称	类型	参数	返回值	描述
1	setOptionValue:forKey:ofCategory	F	NSString NSString BDCloudMediaPlayerOptionCategory	-	设置播放器选项，选项值为字符串。
2	setOptionIntValue:forKey:ofCategory	F	int64_t NSString BDCloudMediaPlayerOptionCategory	-	设置播放器选项，选项值为整数值。

HLS下载

BDCloudMediaItem

抽象一个离线HLS视频。

序号	名称	类型	参数	返回值	描述
1	identify:	P	-	-	只读。标识(SDK自动生成)。
2	url:	P	-	-	只读。视频的原始URL。
3	title:	P	-	-	只读。视频的title。
4	path:	P	-	-	只读。视频数据保存路径。
5	index:	P	-	-	只读。可播放文件名。全路径为 path/index
6	size:	P	-	-	只读。数据文件大小。下载完成后有值。
7	status:	P	-	-	只读。状态。Downloading/Ready/Miss。
8	progress:	P	-	-	只读。下载进度。

BDCloudMediaDownloadTask

抽象一个下载任务。

序号	名称	类型	参数	返回值	描述
1	item:	P	-	-	只读。离线HLS视频相关信息。
2	state:	P	-	-	只读。任务的状态。Wait/Running/Suspend/Canceled/Failure/Finish
3	progress:	P	-	-	只读。任务下载进度。值与item.progress一致。

BDCloudMediaDownload

下载管理。

序号	名称	类型	参数	返回值	描述
1	delegate:	P	-	-	设置或获取事件代理
2	initWithUser:	F	用户	-	传入用户名进行初始化，事件代理可通过属性设置
3	initWithUser:delegate:	F	用户 事件代理	-	同时设置用户名、事件代理进行初始化
4	frozen:	F	BOOL是否冻结调度	-	设置是否冻结调度
5	mediaItems	F	-	离线HLS视频信息数组	返回所有下载中和完成的离线HLS视频信息
6	removeMediaItem:	F	离线HLS视频信息	-	删除一个离线HLS视频
7	downloadTaskWithURL:title:error:	F	视频URL 视频title 错误	下载任务	创建一个下载任务
8	suspendTask:	F	任务	-	挂起任务
9	resumeTask:	F	任务	-	恢复任务
10	cancelTask:	F	任务	-	取消任务(会删除已下载数据)
11	resumeUncompletedTasks	F	-	任务列表	查询并返回上次未完成的任务列表
12	stopAllTasks	F	-	-	停止所有任务
13	clean	F	-	-	清理已下载的所有数据和目录。调用后此实例将不再可用。

BDCloudMediaDownloadDelegate

下载回调事件

序号	名称	类型	参数	返回值	描述
1	taskStart:	F	任务	-	任务开始
2	task:needAuthentication:	F	任务 认证信息存储字典	-	设置任务鉴权信息 可设置playerId或token字段
3	task:progress:	F	任务 进度	-	任务进度汇报
4	task:speed:	F	任务 网速	-	任务网速汇报
5	taskEnd:error:	F	任务 错误	-	任务完成，error为空时表示下载成功

缩略图显示

BDCloudSpriteRenderConfiguration

缩略图渲染配置

序号	名称	类型	参数	返回值	描述	
1	row	P	-	-	必选。每张雪碧图中缩略图的行数。	
2	column	P	-	-	必选。每张雪碧图中缩略图的列数。	
3	startTime	P	-	-	可选。开始时间，默认0	
4	thumbnailDuration	P	-	-	必选。每个缩率图之间的时间间隔	
5	imageSize	P	-	-	可选。雪碧图分辨率，默认CGSizeZero	
6	margin	P	-	-	可选。雪碧图边距，默认0	
7	padding	P	-	-	可选。缩略图间距，默认0	
8	scalingMode	P	-	-	可选。渲染模式，默认BDCloudSpriteScalingModeNone	
9	defaultConfiguration	F	-	实例	初始化方法，并设置默认参数	

BDCloudSpriteDownloadConfiguration

缩略图下载配置

序号	名称	类型	参数	返回值	描述
1	imageList	P	-	-	必选。多张雪碧图的地址，且有序
2	maxThreadNum	P	-	-	可选。最大下载线程数，默认1
3	path	P	-	-	可选。磁盘保存路径，默认临时文件夹，覆盖
4	defaultConfiguration	F	-	实例	初始化方法

BDCloudSpriteManager

缩略图显示管理类

序号	名称	类型	参数	返回值	描述
1	renderView	P	-	-	显示缩略图画布
2	status	P	-	-	下载状态
3	initWithRenderConfig:downloadConfig	F	渲染配置 下载配置	实例	初始化方法
6	startDownload:	F	BOOL NSError	-	开始下载并回调
9	stopDownload	F	-	-	结束下载
10	clearCache	F	-	-	清除磁盘
11	renderTime:	F	NSTimeInterval	-	显示某时刻缩率图

网络视频代理

BDCloudMediaPlayerProtocol

播放器网络代理

序号	名称	类型	参数	返回值	描述
1	onEnableIODelegate:url	F	播放器示例 URL	0：支持 -1：不支持	允许网络代理
2	onOpenPlayer:url:	F	播放器示例 URL	0：成功 -1：失败	打开视频
3	onReadplayer:url:buffer:size:	F	播放器示例 URL 视频数据 数据大小	int：本次读取的真实数据大小	视频数据读取
4	onSeekplayer:url:offset:whence:	F	播放器示例 URL 偏移量 偏移类型	0：成功 -1：失败	视频定位
5	onCloseplayer:url:	F	播放器示例 URL	0：成功 -1：失败	关闭视频
6	onTotalSizePlayer:url:	F	播放器示例 URL	int_64：视频总大小	视频总大小获取

网络视频加速

BDCloudMediaSourceManager

播放器网络视频加速

序号	名称	类型	参数	返回值	描述
1	countLimit	p	-	-	预下载通道数。默认50。
2	sizeLimit	p	-	-	最大存储空间。默认500M，单位byte。
3	configuration	p	-	-	下载配置。默认default。
4	readTimeout	p	-	-	播放过程中网络数据下载超时时间。默认500ms。
5	autoWriteCache	p	-	-	开启边播边存。默认YES。
6	enableWhenPreDwonloadFail	p	-	-	预下载失败是否继续走网络库。默认NO。
7	setVideoDuration:duration:	F	视频网络连接 视频时长	-	设置当前视频的总时长，设置后可加快起播速度。
8	setVideoCacheTime:time	F	视频网络连接 缓冲时长	-	数据缓冲时间。默认10S。
9	download:size:complete	F	视频网络连接 预下载大小 成功回调	-	预下载视频一小段，可提升起播速度。
10	downloadCancel:	F	视频网络连接	-	取消预下载。
11	downloadInCache:	F	视频网络连接	-	根据ULR获取视频是否预下载成功。
12	downloadCacheDelete:	F	视频网络连接	-	删除磁盘中下载的资源。
13	downloadCacheDeleteAll:	F	-	-	删除磁盘中全部下载资源。

VR视频播放

BDCloudVRConfiguration VR视频播放配置信息

序号	名称	类型	参数	返回值	描述
1	defaultConfig	F	-	BDCloudVRConfiguration实例	默认配置信息实例。
2	setProviderAVPlayerItem:	F	AVPlayerItem	-	配置VR渲染图像输入源为AVPlayerItem。
3	setProviderImage	F	-	-	配置VR图像输入源为Ulimage图像。
4	setProviderBDCloudMediaPlayerView:viaHardwareAccelerate:	F	UIView:百度云播放器渲染视图 BOOL:是否硬件解码	-	配置VR图像输入源为百度云播放器渲染视图。
5	setInteractiveMode:	F	交互模式	-	配置VR渲染交互模式。
6	setDisplayMode:	F	渲染模式	-	配置VR渲染渲染模式。
7	setProjectionMode:	F	资源类型	-	配置VR渲染资源类型。
8	setPinchEnabled:	F	BOOL	-	配置是否允许捏合手势。
9	setRenderOn:	F	UIView	-	配置VR渲染视图的父视图。

BDCloudVRRenderControl VR视频播放控制

序号	名称	类型	参数	返回值	描述
1	renderWithConfig:	F	VR渲染配置实例	BDCloudVRConfiguration实例	VR渲染控制实例
2	switchImage:	F	UIImage	-	VR渲染控制实例
3	switchInteractiveMode:	F	交互模式	-	切换交互方式
4	getInteractiveMode	F	-	交互方式	获取当前交互方式
5	switchDisplayMode:	F	渲染模式	-	切换渲染模式
6	getDisplayMode	F	-	渲染模式	获取当前渲染模式
7	switchProjectionMode:	F	资源类型	-	切换资源类型
8	getProjectionMode	F	-	资源类型	获取当前资源类型
9	pause	F	-	-	暂停VR渲染。
10	resume	F	-	-	开始VR渲染。

动态码率自适应

BDCloudMediaAdaptiveItem 多码率视频信息

序号	名称	类型	参数	返回值	描述
1	init:bitrate	F	视频地址 视频码率	BDCloudMediaAdaptiveItem	多码率视频信息实例
2	contentString	P	-	-	视频地址
3	bitrate	P	-	-	视频码率
4	mediaIndex	P	-	-	播放器索引

BDCloudMediaAdaptiveControl 动态码率自适应切换

序号	名称	类型	参数	返回值	描述
1	defaultControl	F	-	BDCloudMediaAdaptiveControl	自适应切换实例
1	setItems:	F	NSArray	-	设置多码率视频信息
1	setMaxCacheDuration:	F	NSTimeInterval	-	设置最大缓冲时长
1	setMaxCacheDuration:	F	NSTimeInterval	-	设置最大缓冲时长
1	setCacheDuration:	F	NSTimeInterval	-	设置当前缓冲时长
1	setCurrentPlayIndex:	F	NSInterger	-	设置当前播放索引
1	addLoadSize:loadSpeed	F	下载数据大小 下载速度	-	更新当前下载速度
1	getPredictBandwidth	F	-	int64_t	获取预测宽带
1	setMinLoadSizeInAddLoadSpeed	F	int64_t	-	设置预测宽带的最小下载量
1	setMinLoadDurationInPredictMediaItem	F	int64_t	-	设置预测切换的最小缓冲时长

版本更新记录

版本	功能描述
v3.9.0	- 新增端上超分能力 - 优化AV1播放性能 - 优化弱网播放体验 - 新增Feed流场景展示 - 其他bugfix，性能优化
v3.8.0	- 新增AV1编码格式支持 - 其他bugfix，性能优化
v3.7.0	- BDCloudMediaPlayer.framework 1. 新增外挂字幕支持 2. 新增多音轨、多字幕切换支持 - 其他bugfix，性能优化
v3.6.0	- 新增绿幕抠图能力 - BDCloudMediaPlayer.framework 1. 新增外部渲染接口 - 其他bugfix，性能优化
	- 新增投屏能力

v3.5.0	- 新增智能防挡弹幕和弹幕交互能力 - 新增手势交互控制场景展示、画中画能力展示、Feed流场景展示、『听』视频场景展示 - 播放器内核升级，新增HLS分片请求回调 - 其他bugfix，性能优化
v3.0.0	-提供高级版SDK，支持全景声音效、HDR解码和渲染、超低延时直播 -修复已知问题，优化稳定性和性能
v2.3.7	-播放器内核升级 -Demo添加弹幕、字幕、倍速、续播功能展示 -BDCloudMediaPlayer.framework 1. 新增SEI信息解析和回调 2. 优化4K片源播放体验 3. 修复已知问题，优化稳定性和性能
v2.3.6	-BDCloudMediaPlayer.framework 1. 新增开始录制和结束录制接口
v2.3.5	-BDCloudMediaPlayer.framework 1. 更新了播放器的鉴权方式。 2. 修复反向seek偶现失效问题。 3. 优化了SDK稳定性。
v2.3.4	-BDCloudMediaPlayer.framework 1. 修复弱网下精准seek不准问题。 2. 修复播放器状态不准问题。 3. 修复已知crash。
v2.3.3	-BDCloudMediaPlayer.framework 1. 网络TCP数据请求回调。 -BDCloudMediaAdaptive.framework 1. 新增framework，提供自适应码率切换功能。
v2.3.2	-BDCloudMediaPlayer.framework 1. 新增播放类型接口，支持直播重连。 2. 新增用户自定义信息接口。 3. 新增写入日志地址设置、获取接口。 4. 新增磁盘日志清除接口。 5. 更新多码率设置视频列表接口。 -BDCloudMediaSource.framework 1. 更新预下载失败是否继续走网络库接口的默认配置。 2. 更新网络库下载数据超时接口。 3. 移除网络库网络错误信息，使用播放器错误信息代替。 -BDCloudVRRender.framework 1. 新增framework，提供提供VR视频、图片渲染功能。
v2.3.1	-更新多码率视频初始化接口。(移除contentString属性；初始化参数更新为指定格式的字符串) -更新网络库获取失败信息接口。(参数更新为视频URL) -修复视频渲染偶现crash。 -修复音频初始化偶现crash。
v2.3.0	-新增HLS,MP4等主流媒体格式多码率无缝切换功能。 -新增缓冲区时长设置功能。 -更新网络代理接口，新增视频URL参数。 -修复设置视频开始播放时刻时播放器异常结束问题。
v2.2.9	-修复HLS视频精准seek异常问题。 -修复HLS视频播放进度读取异常问题。 -修复视频DAR信息读取异常问题。
v2.2.8	-新增网络视频加速库。 - 新增边播边存功能。 - 新增预下载功能。
v2.2.7	- 新增缩略图下载显示。 - 新增网络代理。 - 部分视频获取总时长异常bugFix。
v2.2.6	- 新增HLS下载任务网速汇报。 - 视频播放完成通知回调异常bugFix。 - 更新Demo失效播放链接。
v2.2.5	- 已知bugFix
v2.2.4	- 适配iPhoneXS 、iPhoneXS Max。 - 其他已知bugFix。
v2.2.3	- 新增videoBitrate属性，支持视频码率获取。 - 新增audioBitrate属性，支持音频码率获取。 - 新增enableAutoIdleTimerDisabled:方法，支持播放器自动控制屏幕长亮。 - 偶发接收不到播放结束通知bugFix。 - 偶发后台创建播放器黑屏bugFix。 - 其他已知bugFix
v2.2.2	- 新增videoDAR属性，支持自定义视屏长宽比。 - 纯音频文件seek无效bugFix - 其他已知bugFix
v2.2.1	- 新增shouldAutostop属性 - 增加直播时的metadata回调(可应用于直播答题场景) - 直播场景开启追帧播放偶现绿屏bugFix
v2.1.1	- 新增请求超时控制接口 - 支持在url中指定播放区间（使用range参数）

	- 第一帧视频黑屏bugFix - 其他bugFix
v2.1.0	- 升级ffmpeg内核。 - 支持循环播放。 - 支持H265格式。 - 增加精确seek功能。 - 增加弱网下起播重试机制。 - 增加清屏开关。 - 其他bugFix。
v2.0.1	- 修复播放过程中旋转时闪屏问题。 - 解决正常结束播放时，无法收到正确消息类型的问题。 - 持续回调buffer更新bugFix。 - 解决32位机器兼容性问题。 - 解决下载模块兼容性问题。 - 解决播放结束时，偶现收到Error提示的问题。
v2.0.0	升级2.0 全新版本，更新说明参考iOS播放器 2.0 上线公告。

Andriod播放器

简介

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 Android 编程经验。

简介

百度智能云播放器 Android SDK(以下简称"SDK"）是百度智能云推出的 Android 平台视频播放器软件开发工具包 (SDK)，为 Android 开发者提供简单、便捷的开发接口，帮助开发者在 Android 移动设备上实现媒体播放功能。SDK 提供简单、便捷的媒体应用开发能力。

- 本地全媒体格式支持

  突破 Android 平台对视频格式的限制，支持目前所有主流的媒体格式(mp4、avi、wmv、flv、mkv、mov、 rmvb 等)。

- 支持广泛的流式视频格式

  支持多种格式文件渐进式和流式播放: HLS、RTMP、HTTP Streaming。

- 性能强大

  CPU/内存占用率低，视频加载速度快。

- 低门槛、高灵活度实现播放功能

  提供了与系统播放器 MediaPlayer 类高度相似的调用接口，便于开发者快速开发媒体播放应用；提供了与系统播放控件 VideoView 高度相似的接口(该部分以源码形式提供，详见 demo 中的 BDCloudVideoView)；同时提供开发示例。

- 针对流媒体场景进行优化

  提供专门面向流媒体场景的SDK，支持 RTMP、HTTP+FLV、HLS 协议及 H264、HEVC、AV1 和 AAC 编码，包体积更小。

- 媒体文件缓存预取

  支持媒体文件播放前预先加载、预先建立连接，起播放更快。支持媒体流边播边缓存，重复播放时节省流量。

- 版权保护

  支持百度智能云 PlayerBinding 与 Token 加密方式；支持 HLS 加密视频的离线下载和播放。

- CPU架构支持完整

  完整支持 armeabi-v7a,arm64-v8a

功能列表

- 接口
  - 与`MediaPlayer`接口高度相似
  - 提供`BDCloudVideoView`控件，与`VideoView`接口高度一致（开源）
- 版本支持
  - 全媒体版本支持所有本地、在线媒体格式
  - 流媒体版本支持 RTMP、HTTP-FLV、HLS 等点/直播场景
- 播放
  - 支持首屏秒开
  - 支持追帧播放
  - 支持 IPV6
  - 支持多实例播放
  - 支持单实例多次播放
  - 支持纯音频播放
  - 支持续播
  - 支持后台播放
  - 支持倍速播放
  - 支持循环播放
  - 支持网速探测
  - 支持多种画幅缩放模式

    - 支持精准seek
    - 支持seek缩略图预览
    - 支持多维手势交互，包括锁屏、音量、亮度、进度、缩放调节等
    - 支持画中画悬浮小窗播放
    - 支持短视频、Feed流场景播放
    - 支持"听"视频
    - 支持耳机操作
    - 支持多音轨、多字幕切换
- 解码
    - 支持硬件解码
    - 支持解码方式设置
    - 支持AV1解码
- HLS支持
    - 支持 HLS 离线下载
    - 支持 HLS 多码率无缝切换
    - 支持 HLS 分片请求回调
- MP4支持
    - 支持 MP4 格式预下载/边播边存
    - 支持 MP4 多码率无缝切换
- 缓冲区设置
    - 支持缓冲区大小设置
    - 支持缓冲区时长设置
- HTTP 请求设置
    - 支持设置 HTTP 请求的 Header
    - 支持设置 HTTP 请求的 UserAgent
- 支持 APM
- 支持 RTMP 或 HTTP-FLV 直播实时 metadata 信息更新回调
- 支持 DRM 版权保护
- 支持播放中截图
- 支持边播放边录制到本地MP4文件
- 支持SEI信息更新回调
- 弹幕/字幕展示
    - 支持弹幕展示与交互
    - 支持智能防挡弹幕
    - 支持字幕解析和展示
    - 支持外挂字幕
- 支持全景VR视频播放
- 支持投屏
- 支持端上超分

SDK集成

开发与运行环境

- Android Studio
- 支持 Android 4.4 及以上系统版本; 支持 armv7a/arm64。

下载最新的SDK并解压

下载最新的播放器 Android SDK，解压后文件目录如下：

```
├── Baidu-Cloud-Player-Android-<Version>-<Revision>
│   ├── bdplayer-sample
│   ├── BDCloudVideoView_sourcecode
│   ├── libs
│   │   ├── HDRKit.aar            // HDR渲染组件，高级版SDK特有
│   │   ├── RTCPlayerWrapper.aar  // 超低延时直播组件，高级版SDK特有
│   │   ├── VRKit.aar             // VR渲染组件，高级版SDK特有
│   │   ├── ProjectionWrapper.aar // 投屏组件，高级版SDK特有
│   │   ├── VirtualLiveKit.aar    // 绿幕抠图组件，高级版SDK特有
│   │   ├── SRKit.aar             // 端上超分组件，高级版SDK特有
│   │   ├── VideoProcessor.aar    // 视频后处理组件，高级版SDK特有
│   │   ├── DanmakuWrapper.aar    // 弹幕组件
│   │   ├── VideoCache.aar
│   │   ├── PlayerLicense.aar
│   │   └── baiduPlayerSDk.aar
```

其中：

- bdplayer-sample为demo示例；
- BDCloudVideoView_sourcecode为BDCloudVideoView控件的java源代码；
- libs为aar包。

申请license

申请播放器SDK license：您需要登录百度智能云控制台申请获取播放器SDK license。

配置工程

您可以选择使用maven配置，也可以通过手动集成将aar包加入到工程中。

maven配置 在根级gradle中添加mavenCentral仓库，如下所示

```
 buildscript {
    repositories {
        mavenCentral()
    }
}
allprojects {
    repositories {
        mavenCentral()
    }
}
```

在模块gradle中添加具体SDK的依赖，如下所示

```
 defaultConfig {
    packagingOptions {
        pickFirst 'lib/armeabi-v7a/*.so'
        pickFirst 'lib/arm64-v8a/*.so'
    }
}

dependencies {
    // 按需在以下四个版本的baiduPlayerSDK中选择一个即可
    // 流媒体标准版
    implementation "com.baidubce.mediasdk:baiduPlayerSDK:3.9.0"
    // 全媒体标准版
    // implementation "com.baidubce.mediasdk:baiduPlayerSDK-full:3.9.0"
    // 流媒体高级版
    // implementation "com.baidubce.mediasdk:baiduPlayerSDK-advance:3.9.0"
    // 全媒体高级版
    // implementation "com.baidubce.mediasdk:baiduPlayerSDK-full-advance:3.9.0"

    implementation "com.baidubce.mediasdk:playerlicense:3.9.0"
    implementation "com.baidubce.mediasdk:videocache:3.9.0"
    implementation "com.baidubce.mediasdk:danmaku-wrapper:1.0.0"
    // 以下组件为高级版特有
    implementation "com.baidubce.mediasdk:videoProcessor:1.0.5"
    implementation "com.baidubce.mediasdk:hdrkit:1.0.5"
    implementation "com.baidubce.mediasdk:srkit:1.0.5"
    implementation "com.baidubce.mediasdk:virtualLiveKit:1.0.5"
    implementation "com.baidubce.mediasdk:rtcplayer-wrapper:1.0.24"
    implementation "com.baidubce.mediasdk:projection-wrapper:1.0.5"
    implementation "com.baidubce.mediasdk:vrkit:1.0.0"
}
```

手动配置aar包

将baiduPlayerSDK.aar等aar包复制到您工程的app/libs目录下，并在gradle文件的dependencies模块注明aar包路径，如下所示：

```
 defaultConfig {
    packagingOptions {
        pickFirst 'lib/armeabi-v7a/*.so'
        pickFirst 'lib/arm64-v8a/*.so'
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.aar'])
}
```

配置证书 通过百度智能云控制台下载证书，复制到 app/src/main/assets目录下。

配置BDCloudVideoView控件

SDK默认提供BDCloudMediaPlayer类，如果您想使用BDCloudVideoView控件，需复制解压包中BDCloudVideoView_sourcecode目录下的代码到app/src/main/java中。

配置完成

如果选择手动集成aar方式，则配置完成后，目录如下所示:

```
 ├──── bdplayer-sample
|   ├──── app
|   |   ├──── libs
|   |   |   └──── baiduPlayerSDK.aar // aar包位置
|   |   |   └──── VideoCache.aar
|   |   |   └──── PlayerLicense.aar
|   |   ├──── src
|   |   |   ├──── main
|   |   |   |   ├──── assets
|   |   |   |   |   └──── playerxxxxx.license // 证书位置
|   |   |   |   ├──── java
|   |   |   |   |   └──── com.baidu.cloud.videoplayer.widget // BDCloudVideoView位置
```

声明SDK需要的权限

将以下权限加入到您的AndroidManifest.xml中，

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

防混淆设置

将以下语句加入到您的proguard混淆配置文件中，

```
-keep class com.baidu.cloud.**{ *;}
```

快速开始

设置AppId(licenseId)

用户在使用SDK之前需要去百度智能云控制台申请并下载.license文件放到自己工程assets目录下，并将licenseId设置给播放器。

调用BDCloudMediaPlayer的静态方法setAppId来设置appid(licenseId):

```
BDCloudMediaPlayer.setAppId(appid);
```

初始化与设置监听

初始化播放器BDCloudMediaPlayer

```
BDCloudMediaPlayer mMediaPlayer = new BDCloudMediaPlayer(this.getApplicationContext());
```

设置监听回调

```
 // 播放器已经解析出播放源格式时回调
mMediaPlayer.setOnPreparedListener(mPreparedListener);
// 视频宽高变化时回调, 首次解析出播放源的宽高时也会回调
mMediaPlayer.setOnVideoSizeChangedListener(mSizeChangedListener);
// 播放完成时回调
mMediaPlayer.setOnCompletionListener(mCompletionListener);
// 播放出错时回调
mMediaPlayer.setOnErrorListener(mErrorListener);
// 播放器信息回调，如缓冲开始、缓冲结束
mMediaPlayer.setOnInfoListener(mInfoListener);
// 总体加载进度回调，返回为已加载进度占视频总时长的百分比
mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
// seek快速调节播放位置，完成后回调
mMediaPlayer.setOnSeekCompleteListener(mSeekCompleteListener);
// RTMP 或 http-flv 直播时实时 metadata 信息回调
mMediaPlayer.setOnMetadataListener(mOnMetadataListener);
```

直播的播放源在播放过程中，根据播放协议的不同，主播断流时可能走onCompletion回调也可能走onError回调。程序中，需要您检查主播是否完成直播，若未完成直播，自写监听器定期调用 start() 尝试继续拉流播放。

设置播放源与显示目标

设置播放源并让播放器解析准备播放

```
 // 若想设置headers，需使用setDataSource(String url, Map<String, String> headers) 方法
mMediaPlayer.setDataSource(strVideoUrl);
// 播放器仅支持异步准备，在onPrepared回调后方可调用start()启动播放
mMediaPlayer.prepareAsync();
```

设置播放的显示目标

支持设置 setDisplay(SurfaceHolder holder)/setSurface(Surface surface)两种设置方法。设置时机一般在onSurfaceCreated/onSurfaceChanged回调函数中，拿到有效的SurfaceHolder时进行设置。

```
 public void surfaceCreated(SurfaceHolder holder) {
   mMediaPlayer.setDisplay(holder);
}
```

播放控制

开始播放

开始播放或者继续播放均使用start接口。调用该接口时，需保证播放器已经回调onPrepared。

```
mMediaPlayer.start();
```

暂停播放

```
mMediaPlayer.pause();
```

停止播放

```
mMediaPlayer.stop();
```

从stop到再次播放可以走如下几种调用：

- 不更换播放源：stop() --> prepareAsync() --> start() when prepared
- 更换播放源：stop() --> reset() --> setDataSource(url) --> prepareAsync() --> start() when prepared

释放播放器

```
mMediaPlayer.release();
```

释放后，重新播放需创建新的player。

属性获取

获取音视频时长

获取音视频时长，单位为毫秒

```
long durationInMilliSeconds = mMediaPlayer.getDuration();
```

获取当前播放位置

获取当前播放位置，单位为毫秒

```
long positionInMilliSeconds = mMediaPlayer.getCurrentPosition();
```

获取视频宽高

```
 int videoWidth = mMediaPlayer.getVideoWidth();
int videoHeight = mMediaPlayer.getVideoHeight();
```

BDCloudVideoView控件

`BDCloudVideoView`封装了`BDCloudMediaPlayer`，让您可以直接使用控件的方式来播放视频。该类直接提供了源代码，如果使用，需要将源码复制到您的项目中。大多数情况下，您仅需使用而不需要修改该控件，以防升级时与新版BDCloudVideoView不兼容。

BDCloudVideoView的接口与系统类VideoView接口非常相似，主要作用如下：

1. 内置视频渲染控件(TextureView/SurfaceView均可选)，并可以指定视频渲染的模式(裁剪、填充、铺满)，避免您直接操作mediaplayer去渲染视图的麻烦；

2. 简化了MediaPlayer的控制接口。使用 `new BDCloudVideoView --> setVideoPath --> start`可以快速播放;

3. 封装了与BDCloudMediaPlayer几乎一样的参数设置接口。使用该控件，您不需要再跟`BDCloudMediaPlayer`交互；

BDCloudVideoView播放控制

设置新的播放源

该接口会创建一个全新的BDCloudMediaPlayer内部对象，来播放新的源。

> **注意：**
> 调用stopPlayback()之前，不可多次调用该接口，以防对象泄漏。

```
bdCloudVideoView.setVideoPath(info.getUrl());
```

启动或恢复播放

```
bdCloudVideoView.start();
```

暂停播放

```
bdCloudVideoView.pause(); // 后续可继续调用start()来恢复播放
```

停止播放

该函数会释放当前的BDCloudMediaPlayer内部对象

```
bdCloudVideoView.stopPlayback();
```

释放全部资源

该函数会释放当前的BDCloudMediaPlayer内部对象和渲染目标等对象。当不再使用bdCloudVideoView对象时务必调用。

```
bdCloudVideoView.release();
```

设置播放相关参数

BDCloudVideoView封装了BDCloudMediaPlayer的几乎所有参数设置接口，并以相同名字命名。

> **注意：**
> 下列接口，请在bdCloudVideoView.setVideoPath之前调用，因setVideoPath中初始化BDCloudMediaPlayer时，需要用到这些参数的设置。

举例：

```
bdCloudVideoView.setLogEnabled(boolean enabled); // 是否显示日志
bdCloudVideoView.setDecodeMode(int decodeMode); // 设置解码模式
bdCloudVideoView.setInitPlayPosition(long milliSeconds); // 设置初始播放位置
bdCloudVideoView.setUseApmDetect(boolean useApmDetect); // 是否开启APM探测；若开启，需额外嵌入APM SDK
bdCloudVideoView.setMaxProbeTime(int maxProbeTimeInMs); // 设置最大探测时长
bdCloudVideoView.setMaxProbeSize(int maxProbeSizeInBytes); // 设置最大探测的数据大小
bdCloudVideoView.setMaxCacheSizeInBytes(int sizeInBytes); // 设置最大缓存数据大小
bdCloudVideoView.setLooping(boolean isLoop); // 是否循环播放
bdCloudVideoView.setBufferTimeInMs(int milliSeconds); // 设置"加载中"触发时，缓存多长时间的数据才结束
```

另外，BDCloudVideoView还可以设置显示模式:

```
// VIDEO_SCALING_MODE_SCALE_TO_FIT 填充，遵守宽高比，有黑边
// VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING 裁剪，遵守宽高比，无黑边，但视频边缘可能被裁剪
// VIDEO_SCALING_MODE_SCALE_TO_MATCH_PARENT 铺满，不遵守宽高比，直接铺满显示区域
bdVideoView.setVideoScalingMode(BDCloudVideoView.VIDEO_SCALING_MODE_SCALE_TO_FIT);
```

指定使用TextureView或SurfaceView

- TextureView在Android 4.0引入，Android 4.1真正可用(因为4.1加入了setSurfaceTexture接口，可以避免黑屏)。该控件支持动画、旋转、截图等功能。如果您的播放器在ListView或ScrollView中，TextureView是首选，否则滚动的时候可能会出现边缘闪动。但缺点是性能表现不如SurfaceView。
- SurfaceView在Android 1.0即引入，缺点是不支持旋转等动画，不宜放在有滚动条的viewgroup中。但优点是，有更好的性能，如果长时间播放视频，SurfaceView是首选。

BDCloudVideoView默认优先使用TextureView(即：4.1及以上版本使用TextureView, 老版本使用SurfaceView)。
如果您想设置不管系统版本，始终使用SurfaceView，以达到最好的性能，节省电量消耗，可将以下构造函数的第二个参数设置为false：

```
bdCloudVideoView = new BDCloudVideoView(Context context, boolean useTextureViewFirst);
```

自定义"加载中"的界面提示

默认的加载中提示为 『一个加载圈』+『正在缓冲...』的提示。

如果您想定制『加载中』的界面和提示信息，需要调用如下代码：

```
bdCloudVideoView.showCacheInfo(false);
```

然后，就可以在App层面自己写相关的控件，并自己控制该控件的显示与隐藏。

工程中使用BDCloudVideoView实例

初始化设置播放源并启动

```
BDCloudVideoView.setAK(ak);
bdVideoView = new BDCloudVideoView(this);
/**
 * 注册listener
 */
bdVideoView.setOnPreparedListener(this);
bdVideoView.setOnCompletionListener(this);
bdVideoView.setOnErrorListener(this);
bdVideoView.setOnInfoListener(this);
bdVideoView.setOnBufferingUpdateListener(this);
bdVideoView.setOnPlayerStateListener(this);

bdVideoView.setVideoScalingMode(BDCloudVideoView.VIDEO_SCALING_MODE_SCALE_TO_FIT);

RelativeLayout.LayoutParams rllp = new RelativeLayout.LayoutParams(-1, -1);
rllp.addRule(RelativeLayout.CENTER_IN_PARENT);
mViewHolder.addView(bdVideoView, rllp);

// 参见demo中的AdvancedMediaController
mediaController.setMediaPlayerControl(bdVideoView);

bdVideoView.setVideoPath(info.getUrl());
// BDCloudVideoView无prepare相关接口，允许setVideoPath后立即调用start。
bdVideoView.start();
```

停止并释放BDCloudVideoView所有资源

> **注意：**不使用该VideoView时务必调用本接口。

```
bdVideoView.stopPlayback();
bdVideoView.release();
```

同一界面播放下一个新的视频源

```
bdVideoView.stopPlayback(); // 释放上一个视频源
bdVideoView.reSetRender(); // 清除上一个播放源的最后遗留的一帧
bdVideoView.setVideoPath(info.getUrl());
bdVideoView.start();
```

同一界面播放同样视频的不同分辨率链接

这种情况适用于您不同清晰度的视频是不同的播放链接时。如果不同清晰度视频在同一个链接内(即Master m3u8文件)，参考进阶里面的"多码率切换"。

```
bdVideoView.stopPlayback(); // 释放上一个视频源
bdVideoView.setInitPlayPosition(1 * 1000); // 指定初始播放位置，单位为毫秒
bdVideoView.setVideoPath(info.getUrl());
bdVideoView.start();
```

详细代码参见demo中的AdvancedPlayActivity.java或SimplePlayActivity.java

快速进阶

播放控制条

简单播放控制条可参考demo中的SimpleMediaController类。该类由播放按钮、播放进度条等组成。 涉及到的接口有：

接口	功能
long getDuration()	获取视频时长，单位为毫秒
long getCurrentPosition	获取当前播放位置，单位为毫秒
void seekTo(long milliSecond)	切换到某处播放
start()	开始播放
pause()	暂停播放
OnBufferingUpdateListener	监听，回调时返回已缓存时长占视频播放时长的百分比，根据该值更新二级进度条（缓存进度）
boolean isPlaying()	是否正在播放

高级设置接口

BDCloudMediaPlayer提供多个高级设置接口：

接口	功能
setBufferSizeInBytes(int size)	- 设置缓冲过程中，起播数据字节长度; - 即『加载中』缓冲字节数达到这个size后，回调BUFFERING_END; - 默认为 1*1024*1024，即1M，上限为4M；
setBufferTimeInMs(int time)	- 设置缓冲过程中，起播数据时长，单位为毫秒; - 即『加载中』缓冲数据可播放时长达到这个time后，回调BUFFERING_END; - 与上面的设置函数相关，以先达到者为准；默认为1*1000，即1秒上限为4秒。
setDecodeMode(int mode)	设置解码模式，可以设置为 DECODE_AUTO(优先硬解，其次软解) 或者 DECODE_SW(软解)，默认为DECODE_AUTO;
setLogEnabled(boolean enable)	- 设置开启播放器Logcat输出，默认为false； - 发布应用时，需保证该值为false；
setMaxCacheSizeInBytes(int size)	- 设置最大缓存区大小； - 默认为 5*1024*1024，即 5M，上限为5M。 - 下载数据领先于播放数据的大小。
setMaxProbeSize(int maxProbeSize)	- 设置最大视频探测probe大小； - 默认为 1*1024*1024，即1M，上限为4M；
setMaxProbeTime(int maxProbeTime)	- 设置最大视频探测probe时长，单位为毫秒； - 默认为1*1000，即1秒，上限为4秒。
setVolume(float leftVolume, float rightVolume)	设置左右声道的音量
setInitPlayPosition(long positionInMilliSeconds)	设置初始播放位置
setTimeoutInUs(int timeout)	设置建立连接和数据下载过程中超时时长，单位为微秒
setAudioDataCallback(OnReceiveAudioDataCallback callback)	设置正在播放的音频回调

多码率有感切换

当播放的是HLS多码率视频时，播放器支持在播放过程中实时切换码率。

获取多码率视频(Master M3U8)的index数目

```
String[] getVariantInfo()
```

- 该variantinfo直接从Master m3u8文件中取值，根据(#EXT-X-STREAM-INF)格式的不同，取到的值可能为：1920x1080,3541000也可能为,232370(不规范的视频)。逗号前是分辨率，逗号后是码率；
- 数组的大小即为多码率的数目；

选择多码率

播放过程中，选择多码率：

```
mMediaPlayer.selectResolutionByIndex(index);
```

该函数会使得player内部的状态变化stop --> prepareAsync --> prepared。

BDCloudMediaPlayer还开放了selectVariantByIndex接口。该接口不帮忙维护mediaplayer状态。如果想使用该接口，需要保证在prepareAsync之前调用。

多码率无缝切换

播放器不仅支持HLS的多码率快速切换，同时也支持MP4等主流媒体格式的无缝切换功能。具体操作如下：

Master playlist 的HLS格式多码率无缝切换

1. 设置输入格式: mMediaPlayer.setMediaInputType(mode)。
2. 在PlayerState.STATE_PREPARED状态后，调用mMediaPlayer.getMediaItems()获取多码率信息列表，更新UI。
3. 用户根据对应索引切换到指定码率: mMediaPlayer.selectMediaByIndex(index)。
4. 当收到IMediaPlayer.MEDIA_INFO_MEDIA_CHANGE_START通知表示视频开始切换。
5. 当收到IMediaPlayer.MEDIA_INFO_MEDIA_CHANGE_END通知表示切换结束，同时可以取extra值判断切换是否成功。此时播放器缓冲播放完成后即会切换到新的码率显示。
6. 当收到onVideoSizeChanged(...) 回调表示视频分辨率发生变化。

MP4等非嵌套码率的无缝切换

1. 设置输入格式: mMediaPlayer.setMediaInputType(mode)。
2. 设置多码率视频链接（注意这里和HLS的区别）：mMediaPlayer.setMediaItems(mediaItems)。
3. 在PlayerState.STATE_PREPARED状态后，调用 mVideoView.getMutilMediasDsc()获取多码率信息列表，更新UI。
4. 用户根据对应索引切换到指定码率 mMediaPlayer.selectMediaByIndex(index)。
5. 当收到IMediaPlayer.MEDIA_INFO_MEDIA_CHANGE_START通知表示视频开始切换。
6. 当收到IMediaPlayer.MEDIA_INFO_MEDIA_CHANGE_END通知表示切换结束，同时可以取extra值判断切换是否成功。此时播放器缓冲播放完成后即会切换到新的码率显示。
7. 当收到onVideoSizeChanged(...)回调表示视频分辨率发生变化。

播放HLS加密视频

百度智能云MCP服务和VideoWorks服务支持转码成HLS加密视频。

不同的加密方式，播放时的方法略有不同：

- PlayerBinding加密模式

  按普通视频播放即可，必须使用百度播放器(Web、 Android、 iOS)。

- Token模式(推荐)

  Token模式是PlayerBinding模式的升级版，根据约定的token进行更加安全地播放。
  token需要您的服务器与百度智能云服务器合作来生成，您的App从您的服务器拿到token后，设置给播放器即可。 prepare之前需要先设置token:

```
mMediaPlayer.setDecryptTokenForHLS(mDrmToken);
```

HLS视频下载

播放器支持对M3U8视频进行下载和管理。

下载管理器

下载管理器以单例的形式提供：

```
globalVideoDownloadManager = VideoDownloadManager.getInstance(context, userName);
```

其中userName仅用于做下载记录的区分，不要求为实际用户名，可以是实际用户名的加密串或签名串。

下载步骤

1. 调用globalVideoDownloadManager = VideoDownloadManager.getInstance(context, userName)获得下载管理单例。
2. 调用globalVideoDownloadManager.startOrResumeDownloader(url, observer)启动下载任务。

   函数原型：

```
public void startOrResumeDownloader(String url, DownloadObserver observer)
public void startOrResumeDownloaderWithToken(String url, String token, DownloadObserver observer) // 下载加密视频
```

此后也可调用暂停（pauseDownloader）、恢复（startOrResumeDownloader）、删除（deleteDownloader）、批量停止（stopAll）等接口对下载任务进行管理。

其中，在启动下载时，若想实时监听下载进度及状态，可继承DownloadObserver类并实现update方法，将该类的实例传给startOrResumeDownloader即可实时监听。不想实时监听，startOrResumeDownloader方法的第二个参数可传null。

```
public void update(DownloadableVideoItem downloader)
```

3. 实时获取下载任务的信息

   任务下载过程中，回调update(DownloadableVideoItem downloader)，实时获取下载状态。

   downloader对象可获得下述方法：

方法	描述
getUrl()	获取单个下载的url
getLocalAbsolutePath()	获取下载文件的本地绝对路径
getProgress()	获取下载进度
getStatus()	获取下载状态（枚举类DownloadStatus）
getErrorCode()	下载状态为Error的时候，通过该接口获取错误码
getFailReason()	获取下载失败或中止的简单描述

> **注意：** 错误码如DownloadableVideoItem.ERROR_CODE_INVALID_URL等

其中，发生错误时的错误码如下表：

错误码	描述
ERROR_CODE_NO_ERROR	无错误
ERROR_CODE_INVALID_URL	地址无效
ERROR_CODE_NETWORK_FAILED	网络问题
ERROR_CODE_SDCARD_UNMOUNTED	本地存储问题
ERROR_CODE_M3U8_\INVALID_FORMAT	m3u8格式问题
ERROR_CODE_M3U8_SAVE_FAILED	m3u8存储问题
ERROR_CODE_M3U8_DRM_INVALID	drm保护相关key或者token无效
ERROR_CODE_TS_SAVE_FAILED	ts文件保存失败

4. 通过URL查找下载任务

   通过 `downloadableVideoItem = globalVideoDownloadManager.getDownloadableVideoItemByUrl(url)` 获得下载任务item，以便访问item中的信息。

   通过 `globalVideoDownloadManager.getAllDownloadableVideoItems()`获得所有下载单元，可以从中选择出未下载完成的任务，启动继续下载(`startOrResumeDownloader(oneItem.getUrl(), null)`)，即可实现断点续传。

5. 下载成功后，`DownloadableVideoItem#getLocalAbsolutePath` 方法拿到本地视频文件全路径，传给播放器进行本地播放即可。

预下载/边播边缓存

代理缓存器支持MP4视频的预下载和边播边存

设置视频预下载/边播边缓存步骤

```
 // 获取缓存代理管理器
mProxyCacheManager = ProxyCacheManager.getInstance();
// 获取播放代理链接
String proxyPath = mProxyCacheManager.getProxyUrl(mAppContext, path);
// 设置缓存进度监听器
mProxyCacheManager.setCacheAvailableListener(cacheListener);
// 将代理链接设置给播放器
setVideoURI(proxyPath);
// 反注册缓存监听器，释放资源
mProxyCacheManager.release();
```

详细代码参见demo中的`BDCloudVideoView.java`。

边播边录制

```
 // 创建录制控制器
mRecordController = new RecordController();
// 初始化录制控制器
mRecordController.init(mNewSurfaceTexture,RecordConstants.VIDEO_ENCODE_FRAME_RATE,RecordConstants.DEFAULT_BIT_RATE_GTE_API18)
// 设置渲染回调
mVideoRender.setRenderList(mRecordController.getRenderCallbackList());
// 设置录制尺寸
mRecordController.setRecordSize(mWidth, mHeight);
// 开始录制
mRecordController.startRecord(filePath);
// 设置录制监听
mRecordController.setRecordListener(listener);
// 停止录制
mRecordController.stopRecord();
```

详细代码参见demo中`BDCloudView.java`，`TextureRenderView.java`。

外挂字幕

通过下面的接口可以添加外挂字幕，当前支持的字幕格式包括srt\ssa\ass\webvtt。需要注意的是，该接口需要在收到播放器onPrepared回调后调用，并且对于同一个播放器，同一时刻仅能添加一个外挂字幕轨道，添加新的外挂字幕会自动代替旧的外挂字幕。

```
 // extSubUrl对应外挂字幕地址
public void addExtSubtitleUrl(String extSubUrl);
```

在播放内核中，当外挂字幕成功读取后，会通过下面的回调进行通知

```
private final IMediaPlayer.OnExtSubtitleOpenListener mExtSubtitleOpenListener =
        new IMediaPlayer.OnExtSubtitleOpenListener() {
            @Override
            public void onExtSubtitleOpen(IMediaPlayer mp, String url) {
                // 外挂字幕打开
            }
        };
```

具体的字幕内容也会在`IMediaPlayer.OnTimedTextListener`中回调，这一点与内嵌字幕相同。

多音轨、多字幕切换

对于带有多个音轨、字幕轨的片源，可以通过下面的方式实现轨道的无缝切换。

1. 先利用下面的接口获取当前片源所有的轨道信息

```
public @Nullable BDCloudTrackInfo[] getTrackInfo();
```

在`BDCloudTrackInfo`类中记录了每个轨道的媒体类型和语言信息。

2. 根据轨道信息数组的下标进行切换

利用下面的接口可以执行轨道的选择和反选，传入的参数即对应第一步获取的轨道信息数组下标。

```
 public void selectTrack(int track);
public void deselectTrack(int track);
```

3. 查询当前选择的轨道

利用下面的接口可以获取当前选中的视频/音频/字幕轨，返回参数也对应第一步获取的轨道信息数组下标。

```
public int getSelectedTrack(@ITrackInfo.TrackType int trackType)
```

4. 轨道变化回调

轨道的添加、切换都会通过下面的回调进行通知，其中action对应轨道添加或切换事件，type对应轨道类型，trackid则对应轨道序号。

```
interface OnTrackChangedListener {
    void onTrackChanged(IMediaPlayer mp,
                        @ITrackInfo.TrackAction int action,
                        @ITrackInfo.TrackType int type,
                        int trackid);
 }
```

5. 无缝切换与有感切换

播放器默认使能轨道的无缝切换（切换过程中无缓冲状态），如果需要切换为有感切换，可以在`prepareAsync`之前设置下面的选项来实现

```
setOption(OPT_CATEGORY_PLAYER, "enable-smooth-track-change", 0);
```

AV1解码支持

在播放器SDK流媒体版中，依赖系统硬解和系统软解能力实现对AV1编码格式的支持，如果系统不具备AV1解码能力，则通过`OnErrorListener`接口回调`IMediaPlayer.MEDIA_ERROR_VIDEO_DECODER_NOT_SUPPORTED`错误。

在播放器SDK全媒体版中，集成了单独的AV1软件解码器，此时的解码器选择顺序为：优先选择系统硬件解码，否则使用AV1软件解码。

常见错误码含义

常见错误码分为播放内核错误、鉴权错误、其他错误三类，具体定义和含义解释可以参考Demo工程`CommonUtils.java`中`getReadableErrorMsg`方法的实现。

高级版功能接入

全景声功能接入

接入准备

接入全景声功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍

在高级版SDK中，提供了全景声（WANOS）音频格式的解码和音效处理能力。其中解码能力无需调用任何接口，由播放内核原生支持。音效处理能力由专门的音效接口提供，既可对全景声（WANOS）格式进行音效处理，也可以对AAC、MP3等常规音频格式进行处理，优化听感。 当前支持的音效列表如下：

音效名称	效果说明
扬声器原声模式	原声，保留多声道听感
扬声器电影模式	使用扬声器虚拟环绕技术，增加声场宽度，使声场以及某些声像不仅仅局限于两个喇叭之间，而能扩展至两个扬声器外侧，提高声音的沉浸感
扬声器音乐模式	音乐相对于电影来说，更需要注重声音的音质，此模式采用最佳的频率响应，不加任何环绕处理，增强了语音的清晰度，使音乐声音更加自然
耳机原声模式	原声，保留多声道听感
耳机电影模式	使用耳机端的虚拟环绕技术，扩展声音的宽度，提高沉浸感，同时在一定程度上减小头中效应
耳机音乐模式	采用最佳的频率响应，不加任何环绕处理，增强了语音的清晰度，使音乐声音更加自然
耳机全景环绕模式	采用动态增强算法，配合科学的滤波处理，提高声音动态感，提升可玩性；让声音包围双耳，在一定程度上较小头中效应

Demo体验

接口说明

在BDCloudMediaPlayer中定义了音效类型枚举

```
public enum AUDIO_EFFECT {
  // 关闭音效
  CLOSE,
  // 扬声器原声模式
  SPEAKER_ORIGINAL,
  // 扬声器音乐模式
  SPEAKER_MUSIC,
  // 扬声器电影模式
  SPEAKER_MOVIE,
  // 耳机原声模式
  EARPHONE_ORIGINAL,
  // 耳机音乐模式
  EARPHONE_MUSIC,
  // 耳机电影模式
  EARPHONE_MOVIE,
  // 耳机全景环绕模式
  EARPHONE_SURROUND
}
```

BDCloudMediaPlayer提供如下的音效设置接口，在播放过程中传入不同的音效枚举类型即可实现音效处理的切换，接口会对SDK有效性和证书有效性做校验，如不符合高级版SDK要求，会抛出异常。

在Demo中也对此接口的使用做了展示，可以参考。

```
public void setAudioEffect(AUDIO_EFFECT audioEffect) throws FeatureException
```

HDR功能接入

接入准备 接入HDR功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了HDR视频的解码和渲染能力，能够让HDR视频在高低端机型上都得到正确的色彩呈现。其中解码能力由播放内核原生支持，渲染能力由HDRKit组件和VideoProcessor组件提供，请确保该组件已集成到你的APP中。

SDK当前支持的HDR标准有：HDR10、HLG、HDR Vivid。

Demo体验

快速开始

1. 初始化设备HDR能力信息和HDRKit组件

```
 // 查询设备HDR能力信息
HdrInfo.initHDRDeviceInfo(mAppContext);
mHdrKit = new HdrKit();
// 需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台(https://console.bce.baidu.com/bvc/#/bvc/player-license/list)查看
mHdrKit.init(mAppContext, "your-license-id");
```

2. 设置输出surface和输入surface，典型场景是在onSurfaceCreated回调中进行设置

```
 // 设置目标surface，HDR处理后的内容将渲染到此surface上
mHdrKit.prepareWithTargetSurface(mSurfaceholder.getSurface());
// 获取输入surface, 将原始画面渲染到此surface上
mMediaPlayer.setSurface(mHdrKit.getInputSurface());
```

3. 渲染参数设置

```
 // 设置输入画面宽高
mHdrKit.setInputSize(mSurfaceWidth, mSurfaceHeight);
// 设置渲染输出宽高
mHdrKit.setOutputSize(mSurfaceWidth, mSurfaceHeight);
// 设置输入视频的HDR标准，默认为HDR10
mHdrKit.setInputVideoContentType(HdrKit.ContentType.VIVID);
// 设置是否自动调节屏幕亮度，默认开启。
mHdrKit.setEnableBrightAdjust(true);
```

4. HDR元数据设置

```
 播放器提供了HDR10/HLG静态元数据和HDR Vivid动态元数据的回调
// 静态元数据回调
public boolean onHdrStaticMetadata(IMediaPlayer mp, int metadataType, byte[] metadata)
// 动态元数据回调
public boolean onVividMetadata(IMediaPlayer mp, byte[] metadata)
在这两个回调中，可以相应地将元数据设置给HDRKit组件
// 设置静态元数据
mHdrKit.setHdrStaticMetadata(metadataType, metadata);
// 设置动态元数据
mHdrKit.setVividMetadata(metadata);
```

5. 开始渲染，典型场景是随播放器的start调用

```
 // 开始渲染到目标surface
mHdrKit.startRenderFromSurface();
```

6. 释放，典型场景是随播放器的release调用

```
 // 销毁，如果要重新使用，则需要重新init
mHdrKit.release();
```

7. 高级用法

HDRKit组件除了可以用surface作为输入输出之外，还可以用纹理id作为输入输出，使用纹理ID时需使用如下的接口

```
 // 设置输入纹理id, 该id对应的是未处理的视频，HDR处理后的内容将渲染到指定的输出纹理id或屏幕上
mHdrKit.prepareWithInputTexId(texid);
// 设置输出纹理id，如果不调用，则直接渲染到屏幕
mHdrKit.setOutputTexId(outputTexId)
// 开始基于输入纹理id的渲染，调用一次，渲染一帧。典型场景是在GLSurfaceView的onDrawFrame中调用
mHdrKit.drawFrameFromTexId()
```

在播放器SDK Demo中的BDCloudVideoView类对上述流程有详细的展示，可以参考。

接口说明

HDRKit类

枚举成员	描述
ContentType.HDR10	HDR10标准视频
ContentType.HLG	HLG标准视频
ContentType.VIVID	HDR Vivid标准视频
ContentType.SDR	SDR视频

接口名	描述
void init(Context context, String appID)	初始化，鉴权失败时会抛出异常
void release()	销毁。如果要重新使用，则需要重新调用init
int getState()	获取HDRKit当前状态
void setInputVideoContentType(ContentType contentType)	输入视频的hdr类型，默认为HDR10
void setInputSize(int inputWidth, int inputHeight)	输入画面宽高。 必须在prepareWithTargetSurface或prepareWithInputTexId之后调用，否则抛出异常。
void setOutputSize(int outputWidth, int outputHeight)	渲染输出宽高。 必须在prepareWithTargetSurface或prepareWithInputTexId之后调用，否则抛出异常。
void setHdrStaticMetadata(int metadataType, byte[] metadata)	传入HDR静态元数据。 metadataType 为SEI_MASTERING_DISPLAY_COLOUR_VOLUME(137)或SEI_CONTENT_LIGHT_LEVEL(144)
void setVividMetadata(byte[] metadata)	传入HDR Vivid动态元数据
void setEnableBrightAdjust(boolean enable)	是否开启自动亮度调节
boolean getEnableBrightAdjust()	查询是否开启了自动亮度调节
void prepareWithTargetSurface(Surface surface)	设置目标surface，HDR处理后的内容将渲染到此surface上。必须在init之后调用，否则抛出异常。
Surface getInputSurface()	获取输入surface, 将原始画面渲染到此surface上。 必须在prepareWithTargetSurface之后调用，否则抛出异常。
void startRenderFromSurface()	开始渲染到目标surface。 必须在prepareWithTargetSurface之后调用，否则抛出异常。
void prepareWithInputTexId(int texid)	设置输入纹理id, 该id对应的是未处理的视频，HDR处理后的内容将渲染到指定的输出纹理id或屏幕上。 必须在init之后调用，否则抛出异常。
void setOutputTexId(int outputTexId)	设置输出纹理id，如果不调用，则直接渲染到屏幕。 必须在prepareWithInputTexId之后调用，否则抛出异常。
void drawFrameFromTexId()	开始基于输入纹理id的渲染，调用一次，渲染一帧。 必须在prepareWithInputTexId之后调用，否则抛出异常。

超低延时直播功能接入

接入准备 接入超低延时直播功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了超低延时直播流的播放能力，该能力由RTCPlayerWrapper组件提供，请确保该组件已集成到你的App中，SDK接口设计接近Android原生MediaPlayer接口，方便理解与接入。

SDK当前支持的音视频编码格式如下：

- 视频：H.264/HEVC，支持B帧
- 音频：AAC

Demo体验 前往SDK简介与下载页面扫码安装DEMO

快速开始

1. 创建播放渲染视窗

```
 // 在布局文件中添加渲染控件
<com.baidu.rtc.RTCVideoView
        android:id="@+id/brtc_video_view"
    android:layout_width="match_parent"
        android:layout_height="match_parent" />

// 寻回渲染控件对象
mVideoView = (RTCVideoView) findViewById(R.id.brtc_video_view);
```

2. 设置默认加载SO架构并启动SO后下载

```
 // SO提供armeabi/armeabi-v7a/arm64-v8a三种架构，设置默认加载的架构
RTCLoadManager.getInstance(this).setDefaultCpuType("arm64-v8a");
// 注册SO后下载进度监听器
RTCLoadManager.getInstance(this).registerCallback(mLoadListener);
// 开始SO后下载，下载完成后自动加载
RTCLoadManager.getInstance(this).loadLibraries(BRTCPlayerImpl.getDefaultSoDownloadUrl(), "arm64-v8a", null);
```

3. 创建播放器对象及播放器初始化

```
 // 创建播放器对象，需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台(https://console.bce.baidu.com/bvc/#/bvc/player-license/list)查看
mBRTCPlayer = new BRTCPlayerImpl(this, "your-license-id");
// 创建播放参数集，初始化播放器
mPlayerParameters = new BRTCPlayerParameters();
// 使能SO后下载
mPlayerParameters.enableSoLaterLoad(true);
// 设置加载的SO架构
mPlayerParameters.setCpuType("arm64-v8a");
// 设置播放器信令服务地址
mPlayerParameters.setPullUrl(mSignalUrl);
// 初始化播放器 传入播放参数及事件监听器
mBRTCPlayer.initPlayer(mPlayerParameters, this);
```

4. 播放设置

```
 // 设置播放渲染视图
mBRTCPlayer.setSurfaceView(mVideoView);
// 设置媒体流地址
mBRTCPlayer.setStreamUrl(mStreamUrl);
```

5. 播放控制

```
 // 准备播放，获取播放依赖资源
mBRTCPlayer.prepareAsync();
// 开始播放
mBRTCPlayer.startPlay();
// 暂停播放
mBRTCPlayer.pausePlay();
// 恢复播放
mBRTCPlayer.resumePlay();
// 停止播放
mBRTCPlayer.stopPlay();
```

6. 释放播放器

```
 // 释放播放器
mBRTCPlayer.releasePlayer();
```

7. 混淆规则

```
 -keepclasseswithmembernames,allowshrinking,allowoptimization class * {
    native <methods>;
}

-keep interface * {
    public <fields>;
    public <methods>;
}

-keep class com.baidu.rtc.** {*;}
-keep class com.webrtc.** {*;}
```

在播放器SDK Demo中对上述流程有详细的展示，可以参考

接口说明 BRTCPlayerParameters类 | 接口名 | 描述 | |----------|----------| | void setEnableDebug(boolean enable) | 设置是否开启debug日志 | | void setPullUrl(String pullUrl) | 设置信令服务地址，格式为http[:s]//domain/brtc/v3/pullstream | | void setCpuType(String cpuType) | 指定下载的SO库CPU架构 | | void enableSoLaterLoad(boolean enableSoLaterLoad) | 是否开启SO后下载 | | void setAutoPlay(boolean autoPlay) | 设置是否自动启播，默认false | | void setVideoDecodeFormat(int videoDecodeFormat) | 设置视频解码格式，由业务侧确定是否为H265格式，默认为H264格式 | | void setEnableVideoBFrame(boolean bFrame) | 是否开启B帧支持，由业务侧确定是否开启，默认关闭 |

BRTCPlayerEvents类 | 错误码 | 含义 | |----------|----------| | BRTC_PLAYER_ERROR_INVALID_URL = 10000 | URL 格式错误 | | BRTC_PLAYER_ERROR_ICE_CHANNEL = 10001 | ICE 连接错误 | | BRTC_PLAYER_ERROR_RESERVED = 10002 | 替换预留错误码 | | BRTC_PLAYER_ERROR_CONNECTION = 10003 | Peer连接创建失败 | | BRTC_PLAYER_ERROR_LOCAL_SDP_REQUEST = 10004 | 媒体描述请求失败 | | BRTC_PLAYER_ERROR_LOCAL_SDP_SET = 10005 | 媒体描述设置失败 | | BRTC_PLAYER_ERROR_REMOTE_SDP_REQUEST = 10006 | 远端媒体描述请求失败 | | BRTC_PLAYER_ERROR_REMOTE_SDP_SET = 10007 | 远端媒体描述设置失败 | | BRTC_PLAYER_ERROR_INVALID_STATE = 10008 | 播放状态错误 | | BRTC_PLAYER_ERROR_STREAMING_INTERRUPT = 10009 | 媒体流中断，一段时间未收媒体流，SDK内部检测到断流错误后会立即停止播放 | | BRTC_PLAYER_ERROR_LOAD_LIBRARIES = 10010 | SO库文件后下载失败 | | BRTC_PLAYER_ERROR_INVALID_LICENSE = 100011 | license错误 | | BRTC_PLAYER_ERROR_LICENSE_FEATURE_INVALID = 10012 | 当前license不包含低延时播放feature | | BRTC_PLAYER_ERROR_DECODER_OPEN_FAILED = 10013 | 解码器打开失败 |

事件码	含义
BRTC_PLAYER_EVENT_REMOTE_RENDER = 1000	开始远端渲染
BRTC_PLAYER_EVENT_ICE_CONNECTED = 1001	ICE连接成功
BRTC_PLAYER_EVENT_PEER_CONNECTION_CLOSED = 1002	对端连接关闭
BRTC_PLAYER_EVENT_STATS_UPDATED = 1003	媒体流信息更新
BRTC_PLAYER_EVENT_BUFFERING_START = 1004	Buffering start事件
BRTC_PLAYER_EVENT_BUFFERING_END = 1005	Buffering end事件
BRTC_PLAYER_EVENT_ICE_DISCONNECTED = 1006	ICE连接断开
BRTC_PLAYER_EVENT_NO_STREAMING_DETECTED = 1007	没有检测到媒体流
BRTC_PLAYER_EVENT_PLAY_TIME_STATISTIC = 1008 BRTC_PLAYER_EVENT_LOCAL_SDP_SET = 1009 BRTC_PLAYER_EVENT_REMOTE_SDP_ACQUIRED = 1010 BRTC_PLAYER_EVENT_LIBS_DOWNLOAD_COMPLETED = 1011 BRTC_PLAYER_EVENT_LIBS_LOADED_SUCCESS = 1012 BRTC_PLAYER_EVENT_RENDERVIEW_VISIBLE = 1013	启播各阶段耗时统计
BRTC_PLAYER_EVENT_DECODER_OPENED = 1014	解码器成功打开

事件回调	含义
void onPrepared()	播放器准备就绪
void onFirstFrameRendered()	首帧渲染事件
void onResolutionChanged(int w, int h)	分辨率更新回调
void onError(int errCode, String msg)	错误错误回调，错误码定义见前文
void onInfoUpdated(int event, Object msg)	事件更新回调，事件码定义见前文
void onPlayerStateChanged(BRTCPlayer.PlayerState currentState)	播放状态更新
void onSEIRecv(ByteBuffer data)	接收到SEI 消息，回调在解码线程,不应在该回调里实现复杂业务

> SEI 数据格式:[nal_type(1)|sei_type(1)|sei_size(n+1)|sei_payload(n*255+m)|trailing_bits(1)]
> sei_size: 需判断第3个字节起连续0xFF的个数n，加第1个不为0xFF的字节(如: 0x21)，该字段占字节数为 n + 1; 其值为:0xFF*n+0x21;
> sei_payload: SEI字段的值；如:[0x06 0x05 0x18 0x54 0x80 0x83 0x97 0xF0 0x23 0x47 0x4B 0xB7 0xF7 0x4F 0x32 0xB5 0x4E 0x06 0xAC 0x27 0x11 0xFE 0x69 0x79 0x01 0x00 0x00 0x80]

RTCVideoView类 | 枚举类型 | 描述 | |----------|--------| | ScalingType.SCALE_ASPECT_FIT | 缩放模式：适应 | | ScalingType.SCALE_ASPECT_FILL | 缩放模式：拉伸 |

RTCLoadManager类 | 接口名 | 描述 | |----------|----------| | RTCLoadManager getInstance(Context context) | 获取单例 | | void setDefaultCpuType(String cpuType) | 指定下载的SO库CPU架构 | | void registerCallback(LoadListener callback) | 注册SO后下载进度监听器 | | void unregisterCallback(LoadListener callback) | 反注册SO后下载进度监听器 | | void loadLibraries(String soLaterLoadUrl, String cpuType, LoadListener loadListener) | 从soLaterLoadUrl下载cpuType架构的SO，并将进度回调给loadListener。
默认SO下载地址可通过BRTCPlayerImpl.getDefaultSoDownloadUrl()获取 |

BRTCPlayer接口 | 接口名 | 描述 | |----------|----------| | long initPlayer(BRTCPlayerParameters playParameters, BRTCPlayerEvents events) | 初始化播放器。
playParameters 播放参数集
events 播放事件监听 | | void setEventObserver(BRTCPlayerEvents events) | 设置播放事件监听 | | void setSurfaceView(RTCVideoView surfaceView) | 设置播放渲染视窗,当前只支持RTCVideoView | | void

setScalingType(RTCVideoView.ScalingType scaleType) | 设置播放视窗缩放模式 | | void prepareAsync() | 准备播放 | | void setPlayWhenReady(boolean autoPlay) | 设置是否自动启播，默认false | | void startPlay() | 开始播放 | | void pausePlay() | 暂停播放,不停止拉流，仅暂停本地媒体流渲染 | | void resumePlay() | 恢复播放,从暂停状态恢复播放 | | void stopPlay() | 停止播放，停止媒体拉流及渲染，再次播放需调用startPlay() | | boolean hasVideo() | 媒体流是否包含视频流 | | boolean hasAudio() | 媒体流是否包含音频流 | | void releasePlayer() | 释放播放器 | | void setVolume(double volume) | 设备播放音量 | | void setStreamUri(String streamPath) | 设置播放媒体流地址，媒体流URL格式为webrtc://domain/app/stream | | PlayerState getPlayerState() | 获取播放状态 |

投屏功能接入

接入准备 接入投屏功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了DLNA投屏能力，可以将手机端的视频内容推送到大屏端进行播放，并且支持在手机端远程控制大屏端的媒体播放，该能力由ProjectionWrapper组件提供，请确保该组件已集成到你的App中。

快速开始 1.初始化ProjectionEngine组件，传入LicenseID和设备发现回调

```
 // 需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台查看
ProjectionEngine.setLicenseID(BaseApp.APP_ID);
// 设备发现回调，可以在这里将发现到的接收端信息保存起来
private ArrayList<DeviceUpnp> mCastReceivers = new ArrayList<>();
private DeviceUpnp.Devicelistener upnpListenerImp = new  DeviceUpnp.Devicelistener() {

    @Override
    public void OnAddDevice(DeviceUpnp device) {
        Log.d(TAG, "onAddDevice " + device.getUuid());
        mCastReceivers.add(device);
    }

    @Override
    public void OnRemoveDevice(DeviceUpnp device) {

    }
};
ProjectionEngine.getInstance().setUpnpDeviceServerListener(getContext(), upnpListenerImp);
```

2.开始搜索投屏设备，当找到设备时，会利用上面的回调进行通知

```
 ProjectionEngine.getInstance().searchDeviceBegin();
```

3.选择设备并发起投屏

```
 // 创建渲染控制器回调，当投屏开始后，会通知接收端的状态变化
private ProjectionRenderListenerImp projectionRenderListenerImp = new ProjectionRenderListenerImp();
class ProjectionRenderListenerImp  implements ProjectionRender.ProjectionRenderListener {
    @Override
    public void onSetAVTransportURI(int renderId, boolean result) {
        Log.i(TAG, "ProjectionRender onSetAVTransportURI " + renderId + ", result: " + result);
    }

    @Override
    public void onPaused(int renderId, boolean result) {
        Log.i(TAG, "ProjectionRender onPaused " + renderId + ", result " + result);
    }

    public ProjectionRenderListenerImp() {
    }

    @Override
    public void onDuration(int renderId, final Integer duration) {
        Log.i(TAG, "ProjectionRender onDuration " + renderId + ", duration " + duration);
    }

    @Override
    public void onCurrent(int renderId, final Integer current) {
        Log.i(TAG, "ProjectionRender onCurrentPosition " + renderId + ", position " + current);
    }

    @Override
    public void onPlayed(int renderId, boolean result) {
        Log.i(TAG, "ProjectionRender onPlayed " + renderId + ", result " + result);
    }

    @Override
    public void onStoped(int renderId, boolean result) {
        Log.i(TAG, "ProjectionRender onStoped " + renderId + ", result " + result);
    }
}
// 从之前保存的接收端信息中选出一个作为发起投屏的目标
DeviceUpnp device = mCastReceivers.get(position);
private int mCastReceiverIdx = -1;
if (null != device) {
    // 分配渲染控制器， mCastReceiverIdx作为渲染控制器标识，后续的媒体播放控制都基于它完成
    mCastReceiverIdx = ProjectionEngine.getInstance().distributeProjectionRender(device, projectionRenderListenerImp);
    // 设置接收端要播放的媒体url
    ProjectionEngine.getInstance().setAVTransportURL(mCastReceiverIdx, url);
    // 开始播放
    ProjectionEngine.getInstance().play(mCastReceiverIdx, 1);
}
```

4.控制接收端的媒体播放

```
// 播放
ProjectionEngine.getInstance().play(mCastReceiverIdx, 1);
// 暂停
ProjectionEngine.getInstance().pause(mCastReceiverIdx);
// seek, 单位为秒
ProjectionEngine.getInstance().seek(mCastReceiverIdx, 30);
// 停止
ProjectionEngine.getInstance().stop(mCastReceiverIdx);
```

5.停止搜索设备

```
ProjectionEngine.getInstance().searchDeviceEnd();
```

6.结束投屏并释放渲染控制器

```
ProjectionEngine.getInstance().stop(mCastReceiverIdx);
ProjectionEngine.getInstance().releaseProjectionRender(mCastReceiverIdx);
```

在播放器SDK Demo中的AdvancedMediaController类对上述流程有详细的展示，可以参考。

接口说明

ProjectionEngine类

接口名	描述
ProjectionEngine getInstance()	获取投屏引擎单例
void setLicenseID(String id)	传入LicenseID
String getVersion()	获取投屏引擎版本号
void setUpnpDeviceServerListener(Context context, DeviceUpnp.Devicelistener upnpListener)	设置设备发现回调
void searchDeviceBegin()	开始搜索设备，鉴权失败的情况下会抛出异常
void searchDeviceEnd()	停止搜索设备
int distributeProjectionRender(DeviceUpnp device, ProjectionRender.ProjectionRenderListener renderListener)	分配渲染控制器，返回值作为渲染控制器标识
void releaseProjectionRender(Integer renderIndex)	释放渲染控制器
void setAVTransportURL(Integer renderIndex, String url)	设置接收端播放的URL
void play(Integer renderIndex, Integer speed)	开始接收端播放，当前speed仅支持传入1
void pause(Integer renderIndex)	暂停接收端播放
void stop(Integer renderIndex)	停止接收端播放
void seek(Integer renderIndex, Integer realTime)	控制接收端进度seek，单位为秒
Integer getDuration(Integer renderIndex)	获取媒体流时长
Integer getCurPos(Integer renderIndex)	获取当前播放位置

DeviceUpnp类

接口名	描述
String getUuid()	获取接收端设备ID
String getFriendlName()	获取接收端设备名

事件回调	含义
void OnAddDevice(DeviceUpnp device)	发现设备
void OnRemoveDevice(DeviceUpnp device)	移除设备

VR功能接入

接入准备 接入VR全景播放功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了VR全景视频的渲能力，并且可通过陀螺仪进行视角变换。该能力由VRKit组件提供，请确保该组件已集成到你的APP中。

快速开始

1.初始化VRKit组件

```
mVrKit = new VRKit();
// 需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台查看
mVrKit.init(mAppContext, "your-license-id");
```

2.设置目标View，并启动渲染

```
// 设置目标surface，VR渲染的内容将绘制到这个view上
VrRenderView view = new VrRenderView(context);
mVrKit.prepareWithTargetSurface(view);
mVrKit.start();
```

3.获取输入surface，将播放器输出的原始画面渲染到此surface上，

```
mediaPlayer.setSurface(mVrKit.getInputSurface());
```

4.在播放器准备就绪时和视频宽高发生变化时通知VRKit，典型场景是在播放器对应的回调中使用

```
 public void onPrepared(IMediaPlayer mp) {
    mVrKit.onPlayerPrepared();
}
public void onVideoSizeChanged(IMediaPlayer mp, int width, int height, int sarNum, int sarDen) {
    mVideoWidth = mp.getVideoWidth();
    mVideoHeight = mp.getVideoHeight();
    mVrKit.onVideoSizeChanged(mVideoWidth, mVideoHeight);
}
```

5.释放，典型场景是随播放器的release调用

```
 // 销毁，如果要重新使用，则需要重新init
mVrKit.release();
```

在播放器SDK Demo中的BDCloudVideoView类对上述流程有详细的展示，可以参考。

接口说明 VRKit类 | 接口名 | 描述 | | ---------- | ---------- | | void init(Context context, String appID) | 初始化，鉴权失败时会抛出异常 | | void release() | 销毁。如果要重新使用，则需要重新调用init | | void prepareWithTargetSurface(Surface surface) | 设置目标GLTextureView，VR渲染结果将绘制到此textureView上。
必须在init之后调用，否则抛出异常。 | | Surface getInputSurface() | 获取输入surface, 将原始画面渲染到此surface上。
必须在prepareWithTargetSurface之后调用，否则抛出异常。 | | void onPlayerPrepared() | 通知VRKit播放器已经准备就绪。
必须在prepareWithTargetSurface之后调用，否则抛出异常。 | | void onVideoSizeChanged(int w, int h) | 通知VRKit视频宽高发生了变化。
必须在prepareWithTargetSurface之后调用，否则抛出异常。 | | void start() | 开始渲染。
必须在prepareWithTargetSurface之后调用，否则抛出异常。 | | void pause() | 暂停渲染。
必须在prepareWithTargetSurface之后调用，否则抛出异常。 |

绿幕抠图功能接入

接入准备 接入绿幕抠图功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了高精度、高性能的绿幕抠图能力，可实现对绿色或其他纯色背景的自动识别和抠像。该能力由VirtualLiveKit组件和VideoProcessor组件提供，请确保该组件已集成到你的APP中。

SDK当前支持纹理ID输入、纹理ID输出。

Demo体验

快速开始

1. 初始化VirtualLiveKit组件

```
 VirtualLiveKit mVirtualLiveKit = new VirtualLiveKit();
// 需要传入您申请的高级版证书LicenseID，ID可以在百度云控制台查看
mVirtualLiveKit.init(mAppContext, "your-license-id");
```

2. 设置输入和输出纹理ID，典型场景是在GLSurfaceView的onSurfaceCreated或onSurfaceChanged回调中进行设置

```
 // 设置输入纹理ID, 该ID对应的是原始绿幕背景视频，SDK处理后的内容将渲染到指定的输出纹理ID上
mVirtualLiveKit.prepareWithInputTexId(mFgInputTex);
// 设置输出纹理ID，该ID对应的是经过处理后的绿幕背景视频，此时绿幕或其他纯色背景被替换为透明的通道
mVirtualLiveKit.setOutputTexId(mFgOutputTex)
```

3. 渲染参数设置

```
 // 设置输入画面宽高
mVirtualLiveKit.setInputSize(mInputWidth, mInputHeight);
// 设置渲染输出宽高
mVirtualLiveKit.setOutputSize(mOutputWidth, mOutputHeight);
```

4. 抠像色值设置 默认情况下，SDK可以自动识别背景色值进行抠像。使用方也可以手动指定抠像色值，传入后，将按照指定色值进行抠像

```
// 设置是否要自动识别背景色值，默认自动
mVirtualLiveKit.setAutoCalculateKeyColor(false);
// 设置抠像色值，范围[0, 255]
float[] keyColor = {0, 255, 0, 0};
mVirtualLiveKit.setKeyColor(keyColor);
```

5. 开始渲染，典型场景是在GLSurfaceView的onDrawFrame中调用

```
// 开始基于输入纹理ID的渲染，调用一次，渲染一帧
mVirtualLiveKit.drawFrameFromTexId()
```

6. 释放

```
// 销毁，如果要重新使用，则需要重新init
mVirtualLiveKit.release();
```

在播放器SDK Demo中的VirtualLiveRenderView类对上述流程有详细的展示，可以参考。

接口说明 VirtualLiveKit类 | 接口名 | 描述 | | ---------- | ---------- | | void init(Context context, String appID) | 初始化，鉴权失败时会抛出异常 | | void release() | 销毁。如果要重新使用，则需要重新调用init | | int getState() | 获取VirtualLiveKit当前状态 | | void setInputSize(int inputWidth, int inputHeight) | 输入画面宽高。必须在prepareWithInputTexId之后调用，否则抛出异常。 | | void setOutputSize(int outputWidth, int outputHeight) | 渲染输出宽高。必须在prepareWithInputTexId之后调用，否则抛出异常。 | | void setAutoCalculateKeyColor(boolean auto) | 设置是否要自动识别背景色值，默认自动 | | void setKeyColor(float[] rgba) | 设置抠像色值，传入后，将按照指定色值进行抠像，否则自动识别背景色值并抠像。范围[0, 255] | | void prepareWithInputTexId(int texid) | 设置输入纹理ID, 该ID对应的是未处理的视频，SDK处理后的内容将渲染到指定的输出纹理ID上。必须在init之后调用，否则抛出异常。 | | void setOutputTexId(int outputTexId) | 设置输出纹理ID。必须在prepareWithInputTexId之后调用，否则抛出异常。 | | void drawFrameFromTexId() | 开始基于输入纹理ID的渲染，调用一次，渲染一帧。必须在prepareWithInputTexId之后调用，否则抛出异常。 |

端上超分功能接入

接入准备 接入端上超分功能，需要使用播放器SDK高级版（也可以单独接入端上超分SDK），并申请高级版License。

功能介绍 在高级版SDK中，提供了端上超分能力，利用端侧推理能力，实现对低分辨率画面的清晰度提升、噪声和块效应去除，适用于视频播放场景和RTC通话场景。

该能力由SRKit组件和VideoProcessor组件提供，请确保这两个组件已集成到你的APP中。

SDK能力如下表所示

能力	说明
支持的超分倍数	固定2倍
支持的输入分辨率	无限制
支持的输入-输出格式	纹理ID -> 纹理ID 或 Surface -> Surface
模型策略	SDK可以根据目标分辨率&帧率和设备性能自动选择合适的模型，保证实时处理帧率。如果设备性能太差或输入分辨率太大，无法达到实时帧率，SDK也可以给出预期的处理帧率，接入方可以自由决定是否应用超分

Demo体验

快速开始

1. 初始化SRKit组件

```
SRKit mSrKit = new SRKit();
// 需要传入您申请的高级版证书LicenseID，ID可以在百度云控制台查看
mSrKit.init(mAppContext,"your-license-id", new IVideoProcessor.OnVideoProcessorInitListener() {
            @Override
            public void onVideoProcessorInited() {
                // 超分内核初始化完成的回调
            }
});
```

如果您的Android TargetApi为31及以上，还需要在AndroidManifest.xml中添加如下的配置。

```
<uses-native-library android:name="libOpenCL.so" android:required="false" />
```

2. 设置输入和输出

```
SDK支持以纹理作为输入输出，典型场景是配合GLSurfaceView使用，此时使用方式如下
// 设置输入纹理ID, 该ID对应的是原始视频，超分处理后的内容将渲染到指定的输出纹理ID上
mSrKit.prepareWithInputTexId(mInputTex);
// 设置输出纹理ID，该ID对应的是经过超分处理后的视频
mSrKit.setOutputTexId(mOutputTex)
```

SDK也支持以Surface作为输入输出，典型场景是配合TextureView或SurfaceView使用，可以在onSurfaceCreated回调中进行设置，此时使用方式如下

```
// 设置目标surface，超分处理后的内容将渲染到此surface上
mSrKit.prepareWithTargetSurface(mSurfaceholder.getSurface());
// 获取输入surface, 将原始画面渲染到此surface上
mMediaPlayer.setSurface(mSrKit.getInputSurface());
注意：SDK仅支持同类型的输入输出，不支持纹理输入、Surface输出这类情况。
```

3. 超分参数设置

```
// 设置输入视频宽高
mSrKit.setInputSize(mVideoWidth, mVideoHeight);
// 设置渲染输出宽高
mSrKit.setOutputSize(mOutputWidth, mOutputHeight);
// 设置目标帧率和模型选择策略，SDK会依据输入分辨率和目标帧率，自动选择最合适的模型。模型选择策略可以为速度优先或质量优先。
mSrKit.setTargetFps(mTargetFps, ModelManager.STRATEGY_SPEED_FIRST);
```

4. 开始渲染 当以纹理作为输入输出时，通过下面的方式开始渲染，典型场景是在GLSurfaceView的onDrawFrame中调用

```
// 开始基于输入纹理ID的渲染，调用一次，渲染一帧
mSrKit.drawFrameFromTexId()
```

当以Surface作为输入输出，通过下面的方式开始渲染，典型场景是随播放器的start调用

```
// 开始渲染到目标surface
mSrKit.startRenderFromSurface();
```

5. 释放

```
// 销毁，如果要重新使用，则需要重新init
mSrKit.release();
```

在播放器SDK Demo中的GLSurfaceRenderView类和BDCloudVideoView类中对上述流程有详细的展示，可以参考。

最佳实践 在调用prepareWithInputTexId或prepareWithTargetSurface之前，可以依据输入分辨率和帧率获取SDK的预期超分处理帧率，可以根据预期结果，再决定是否应用超分，避免设备性能不足或输入分辨率过高导致的渲染卡顿，示例代码如下

```
float expectFps = mSrKit.probeExpectedFpsForResolution(mVideoWidth, mVideoHeight, mTargetFps, ModelManager.STRATEGY_SPEED_FIRST);
if (expectFps > 0 && expectFps < mTargetFps) {
    Log.w(TAG, "disable sr automatically since expect fps " + expectFps + " is too low");
    return;
} else {
    // 继续进行正常的输入输出设置、参数设置流程
}
```

接口说明 SRKit类 | 接口名 | 描述 | | ---------- | ---------- | | void init(Context context, String appID, OnVideoProcessorInitListener onSrCoreInitListener) | 初始化，鉴权失败时会抛出异常 onSrCoreInitListener：超分内核初始化完成的回调。因为超分内核会根据输入分辨率动态初始化，所以该回调会在调用startRenderFromSurface()或drawFrameFromTexId()之后才抛出 | | void release() | 销毁。如果要重新使用，则需要重新调用init | | int getState() | 获取SRKit当前状态。返回值为IVideoProcessor.STATE_RELEASED / IVideoProcessor.STATE_INIT / IVideoProcessor.STATE_PREPARED / IVideoProcessor.STATE_STARTED | | void prepareWithInputTexId(int texid) | 设置输入纹理ID, 该ID对应的是未处理的视频，超分处理后的内容将渲染到指定的输出纹理ID上。必须在init之后调用，否则抛出异常。 | | void setOutputTexId(int outputTexId) | 设置输出纹理ID。必须在prepareWithInputTexId之后调用，否则抛出异常。 | | void prepareWithTargetSurface(Surface surface) | 设置目标surface，超分后的内容将渲染到此surface上。必须在init之后调用，否则抛出异常. | | Surface getInputSurface() | 获取输入surface, 将原始画面渲染到此surface上。必须在prepareWithTargetSurface(Surface)之后调用，否则抛出异常。 | | void setInputSize(int inputWidth, int inputHeight) | 输入视频宽高。必须在prepareWithInputTexId或prepareWithTargetSurface之后调用，否则抛出异常。 在调用startRenderFromSurface()或drawFrameFromTexId()开始渲染后，不可再扩大inputSize，如果需要扩大，则必须调用release，重新进行初始化。 | | void setOutputSize(int outputWidth, int outputHeight) | 渲染输出宽高。必须在prepareWithInputTexId或prepareWithTargetSurface之后调用，否则抛出异常。 | | void setTargetFps(float fps, @ModelManager.MODEL_STRATEGY int strategy) | 设置目标帧率和模型选择策略，SDK会依据输入分辨率和目标帧率，自动选择最合适的模型。/br<>可选的模型策略包括： ModelManager.STRATEGY_SPEED_FIRST 速度优先 ModelManager#STRATEGY_QUALITY_FIRST 质量优先 | | float probeExpectedFpsForResolution(int width, int height, float targetFps, @ModelManager.MODEL_STRATEGY int strategy) | 获取模型在指定分辨率、目标帧率、模型策略下的预估实际帧率，接入方可以根据预估结果决定是否开启超分。 如果在当前设备上超分内核从来没有初始化过（收到OnVideoProcessorInitListener回调），那么无法获得全局设备算力情况，返回值为0。 | | void enableProcess(boolean enable) | 是否开启超分处理，默认开启，必须在prepareWithInputTexId或prepareWithTargetSurface之后调用，否则抛出异常。 即使不开启超分处理，依然会将画面渲染到指定的纹理或Surface上，只不过渲染结果没有清晰度提升的效果。 |

接口速查

播放器相关

BDCloudMediaPlayer类

接口名	描述
static void setAppId(String appid)	设置AppId(即LicenseId)
static void setCuid(String cuid)	设置自定义cuid,该信息将透传给APM模块上报
static String getSdkVersion()	获取sdk版本，形式为 xx.xx.xx
void setDataSource(Context context, Uri uri)	设置播放源
void setDataSource(Context context, Uri uri, Map<String,String> headers)	设置播放源，可设置请求头信息
void setDataSource(String path)	设置播放源 (file-path or http/rtsp URL)
void setDataSource(String path, Map<String,String> headers)	设置播放源 (file-path or http/rtsp URL)，可设置网络请求头部信息。
void setDecryptTokenForHLS(String token)	DRM加密-token加密，通过该接口设置token
String getDataSource()	获取播放路径
void setDisplay(SurfaceHolder sh)	设置 SurfaceHolder 用于显示视频
void setSurface(android.view.Surface surface)	设置Surface来显示视频，与接口setDisplay(SurfaceHolder)功能类似，但不支持setScreenOnWhilePlaying(boolean) 接口的设置。
void setScreenOnWhilePlaying(boolean screenOn)	设置播放时屏幕保持，仅在设置过SurfaceHolder时有效。
void prepareAsync()	异步准备，播放器仅支持异步准备
void start()	启动播放 要求播放源已经准备好
void pause()	暂停播放
void seekTo(long msec)	快速切换到某个时间点进行播放
void stop()	停止播放
void reset()	重置，将状态重置为IDLE 重置后需重新设置播放源
void release()	释放播放器
long getCurrentPosition()	获取当前播放位置，单位为毫秒
long getDuration()	获取音视频时长，单位为毫秒
void setDecodeMode(int mode)	设置软硬解模式，默认为auto模式(自动检测，优先硬解)
int getDecodeMode()	获取之前设置的解码模式
int getVideoHeight()	获取视频高度
int getVideoWidth()	获取视频宽度
void setLooping(boolean looping)	设置是否循环播放
boolean isLooping()	是否循环播放
boolean isPlaying()	是否正在播放
void setBufferSizeInBytes(int size)	设置缓冲过程中，起播数据字节长度
void setBufferTimeInMs(int time)	设置缓冲过程中，起播数据时长
void setTimeoutInUs(int timeout)	设置建立连接和数据下载过程中的超时时长，单位为微秒，默认值为15秒
void setMaxCacheSizeInBytes(int size)	设置 最大缓冲区长度
void setMaxCacheDuration(int duration)	设置 最大缓冲区时长，单位毫秒
void setMaxProbeSize(int maxProbeSize)	设置probe(音视频格式探测)最大数据大小
void setMaxProbeTimeInMs(int maxProbeTime)	设置probe（音视频格式探测）最大时长，单位毫秒
void setInitPlayPosition(long positionInMilliSeconds)	设置初始播放位置, 需在prepareAsync之前调用
void setLogEnabled(boolean enable)	是否显示debug级别日志
long getDownloadSpeed()	获取网络下载速度
void setVolume(float leftVolume, float rightVolume)	设置左右声道的音量
void setWakeMode(android.content.Context context, int mode)	设置保持唤醒模式
BDCloudTrackInfo[] getTrackInfo()	获取所有音视频track的信息
int getSelectedTrack(int trackType)	获取当前选中的视频/音频/字幕轨
void selectTrack(int track)	选择轨道，track 对应getTrackInfo()返回数组的下标
void deselectTrack(int track)	反选轨道，track 对应getTrackInfo()返回数组的下标
void toggleFrameChasing(boolean isEnable)	设置是否开启追帧播放功能（目前只支持 rtmp 和 http-flv 直播，其他场景请勿开启追帧）
void setAudioDataCallback(OnReceiveAudioDataCallback callback)	设置音频回调 （录制功能使用）
int getCurrentVariantIndex()	有感码率切换场景下，获得当前多码率子码流的index
boolean selectResolutionByIndex(int index)	切换多分辨率，以重新启播的方式实现，不保证无缝，需要首先调用getVariantInfo拿到数组
void selectVariantByIndex(int index)	在起播时设置多码率index，在播放过程中的切换请使用selectResolutionByIndex
String[] getVariantInfo()	有感切换（SOURCE_SWITCH_NORMAL_MODE）场景下，获得多码率视频的各子码流信息
void addExtSubtitleUrl(String extSubUrl)	添加外挂字幕，在播放器prepared之后调用

多码率无缝切换（HLS,MP4等主流媒体格式） | 接口名 | 描述 | | ---------- | ---------- | | void setMediaInputType(int input) | 设置多码率切换模式，可选模式包括
- SOURCE_SWITCH_NORMAL_MODE：普通模式，非无缝切换
- SOURCE_SWITCH_SMOOTH_MODE：MP4无缝切换模式
- SOURCE_SWITCH_SMOOTH_HLS_MODE：HLS无缝切换模式 | | void setMediaItems(String[] mediaItems) | MP4多码率切换场景下，传入各子码流信息 | | String[] getMediaItems() | MP4和HLS无缝切换场景下，获得多码率视频的各子码流信息 | | int getCurrentMediaIndex() | MP4和HLS无缝切换场景下，获得当前的子码流index | | boolean setMediaItemIndex(int index) | 执行无缝码流切换 |

IMediaPlayer接口

接口名	描述
IMediaPlayer.OnBufferingUpdateListener	interface，总缓冲进度监听器
IMediaPlayer.OnCompletionListener	interface，播放完成监听器
IMediaPlayer.OnErrorListener	interface，播放失败监听器
IMediaPlayer.OnInfoListener	interface，信息通知监听器
IMediaPlayer.OnPreparedListener	interface，准备完成监听器
IMediaPlayer.OnSeekCompleteListener	interface，Seek结束监听器
IMediaPlayer.OnVideoSizeChangedListener	interface，视频大小改变监听器
IMediaPlayer.OnMetadataListener	interface，直播视频元信息回调
IMediaPlayer.OnSEIListener	interface，SEI信息回调
IMediaPlayer.OnHdrStaticMetadataListener	interface，HDR静态元数据回调
IMediaPlayer.OnVividMetadataListener	interface，HDR Vivid动态元数据回调
IMediaPlayer.OnTimedTextListener	interface，字幕内容回调
IMediaPlayer.OnHlsSegmentRequestListener	interface，HLS分片请求回调
IMediaPlayer.OnTrackChangedListener	interface，音轨、字幕轨切换回调
IMediaPlayer.OnExtSubtitleOpenListener	interface，外挂字幕打开回调

静态变量名	描述
MEDIA_INFO_UNKNOWN	1，未知通知
MEDIA_INFO_VIDEO_RENDERING_START	3，视频开始渲染
MEDIA_INFO_BUFFERING_START	701，开始缓冲
MEDIA_INFO_BUFFERING_END	702，停止缓冲
MEDIA_INFO_VIDEO_ROTATION_CHANGED	10001，视频方向通知
MEDIA_INFO_AUDIO_RENDERING_START	10002，音频开始播放
MEDIA_INFO_FRAMECHASING_START	10003，直播追帧开始
MEDIA_INFO_FRAMECHASING_END	10004，直播追帧结束
MEDIA_INFO_MEDIA_CHANGE_START	10005，多码率切换开始
MEDIA_INFO_MEDIA_CHANGE_END	10006，多码率切换结束

IMediaDataSource接口（外部媒体数据透传） | 方法 | 描述 | | ---------- | ---------- | | int readAt(long position, byte[] buffer, int offset, int size) | 读取媒体数据 | | long getSize() | 获取媒体文件大小 | | void close() | 关闭外部数据源 |

下载器相关

VideoDownloadManager类

方法	描述
void changeMaxDownloadingItems(int maxItems)	更改最大并行下载数目，默认为5，范围为(1, 10]。
void deleteDownloader(String url)	删除下载
HashMap<String,DownloadableVideoItem> getAllDownloadableVideoItems()	获得所有下载项目(该hashMap为所有下载项的镜像列表)
DownloadableVideoItem getDownloadableVideoItemByUrl(String url)	获得单个下载项目信息
static VideoDownloadManager getInstance(Context context, String userName)	获取下载器单例
String getUserName()	获取设置的用户名
void pauseDownloader(String url)	停止下载
void startOrResumeDownloader(String url, DownloadObserver observer)	开始下载或者恢复下载
void startOrResumeDownloaderWithToken(String url, String token, DownloadObserver observer)	开始下载或者恢复下载
void stopAll()	暂停所有下载

DownloadableVideoItem类

方法	描述
int getErrorCode()	获取错误码 当DownloadStatus的状态为ERROR时，通过该接口获取错误码
String getFailReason()	获取错误描述 推荐使用getErrorCode接口
String getLocalAbsolutePath()	获取本地文件的全路径名
float getProgress()	获得当前下载进度
DownloadableVideoItem.DownloadStatus getStatus()	获取下载状态
String getUrl()	获取url

错误码	描述
ERROR_CODE_NO_ERROR	0，无错误
ERROR_CODE_INVALID_URL	1，地址无效
ERROR_CODE_NETWORK_FAILED	2， 网络问题
ERROR_CODE_SDCARD_UNMOUNTED	3，本地存储问题
ERROR_CODE_M3U8_INVALID_FORMAT	4，m3u8格式问题
ERROR_CODE_M3U8_SAVE_FAILED	5，m3u8存储问题
ERROR_CODE_M3U8_DRM_INVALID	6，drm保护相关key或者token无效
ERROR_CODE_TS_SAVE_FAILED	7，ts文件保存失败

DownloadableVideoItem.DownloadStatus类

枚举成员	描述
DownloadStatus.NONE	初始状态，短暂
DownloadStatus.PENDING	达到最大并发下载数，新加入的下载会pending
DownloadStatus.DOWNLOADING	正在下载
DownloadStatus.PAUSED	下载暂停
DownloadStatus.ERROR	下载出错
DownloadStatus.COMPLETED	下载完成
DownloadStatus.DELETED	删除状态。短暂：删除后会通过该状态回调observer的update，随后记录删除。

网络视频代理缓存

ProxyCacheManager类

方法	描述
static synchronized ProxyCacheManager getInstance()	获取缓存代理管理器单例
void setCacheDir(File cacheDir)	设置视频缓存路径，若路径无效则使用默认路径
void setProxy(HttpProxyCacheServer proxy)	设置网络代理服务
void setCacheAvailableListener(CacheListener listener)	设置视频进度监听器
String getProxyUrl(Context context,String url)	获取代理播放链接，播放器基于此链接进行播放
void cleanVideoCacheDir(Context context)	清空缓存
void release()	反注册缓存监听，释放资源

播放器录制

RecordController类 | 方法 | 描述 | | ---------- | ---------- | | void init(SurfaceTexture surfaceTexture, int framerate, int videoBitrate) | 初始化录制 | | void setRecordListener(IRecordListener listener) | 设置录制监听 | | void setRecordSize(int width, int height) | 设置录制尺寸 | | void startRecord(String filePath) | 开始录制 | | void onUpdateAudioData(byte[] bytes, int byteSize) | 更新正在播放的音频数据（通过调用BDCloudMediaPlayer#setAudioDataCallback 回调后设置） | | boolean isRecording() | 是否正在录制 | | void stopRecord() | 结束录制 | | void onDestroy() | 销毁 |

版本更新记录

版本	功能描述
v3.9.0	- 新增端上超分能力 - 优化AV1播放性能 - 优化弱网播放体验 - 适配API级别31 - 新增Feed流场景展示 - 新增进度条打点展示 - 修复已知问题，优化稳定性和性能
v3.8.0	- 新增AV1编码格式支持 - 修复已知问题，优化稳定性和性能
v3.7.0	- 新增外挂字幕支持 - 新增多音轨、多字幕切换支持 - 投屏能力优化 - 修复已知问题，优化稳定性和性能
v3.6.0	- 新增绿幕抠像能力 - 修复已知问题，优化稳定性和性能
v3.5.0	- 新增VR视频播放能力 - 新增投屏能力 - 新增智能防挡弹幕和弹幕交互能力 - 新增手势交互控制 - 新增画中画能力展示 - 新增Feed流场景展示 - 新增『听』视频场景展示 - 新增HLS分片请求回调 - 播放器内核升级 - 其他bugfix，性能优化
v3.0.0	- 提供高级版SDK，支持全景声音效、HDR解码和渲染、超低延时直播 - 修复已知问题，优化稳定性和性能
v2.3.7	- 播放器内核升级 - 新增SEI信息解析和回调 - Demo添加弹幕、字幕、续播、倍速功能展示 - 其他bugfix，性能优化

v2.3.6	- 新增播放器本地录制功能 - 新增OpenGL，视频渲染自定义Render
v2.3.5	- 更新了播放器的鉴权方式。 - 优化初始位置播放准度。 - 新增应用Assert资源播放。
v2.3.2	- 修复弱网环境下精准seek不准问题 - 其它bugFix
v2.3.1	- 更新播放器性能指标统计 - 其它bugFix
V2.3.0	- 新增HLS,MP4等主流媒体格式多码率无缝切换功能。 - 新增缓冲区时长设置功能。 - 新增cuid设置接口。
V2.2.8	- 修复HLS视频精准seek异常问题 - 修复HLS视频播放进度读取异常问题 - 修复视频DAR信息读取异常问题
V2.2.7	- 新增Seek预览功能 - 新增HLS、MP4协议边播边缓存功能
v2.2.5	- 新增HLS下载任务网速汇报。 - 更新Demo失效播放链接。
v2.2.4	- 播放器性能统计apm sdk更新 - 其他bugFix
v2.2.3	- 缓冲进度条不能达到100%bugFix - 其他bugFix
v2.2.2	- 播放音频时seek不起作用bugFix - 其他bugFix
v2.2.1	- 修复了 http-flv 和 rtmp 直播追帧时 I 帧丢失导致的绿屏问题（目前只支持 rtmp 和 http-flv 直播，其他场景请勿开启追帧） - 添加了 http-flv 和 rtmp 直播的 metadata 回调监听
v2.2.0	- 新增 apm 播放器性能统计 - 调整起播流程，优化起播速度 - 修复倍速播放声音不正常问题 - 其他bugFix
v2.1.1	- 新增请求超时控制接口 - 支持在url中指定播放区间（使用range参数） - Android 4.3以下设备初始化失败bugFix - 其他bugFix
v2.1.0	- 增加精确seek功能 - 支持所有Android版本变速播放 - 升级ffmpeg内核 - 增加弱网下起播重试机制 - 支持H265格式 - 改进缓冲机制，加速起播 - 支持pcdn协议，请提交工单开通 - 其他bugFix
v2.0.1	- 解决播放器结束时偶现Error消息的问题。 - 更新Demo，解决在Flyme机型上切换前后台，可能静帧的问题。
v2.0.0	升级2.0 全新版本，更新说明参考Android播放器 2.0 上线公告。

Unity播放器

简介

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 Unity 开发经验。

简介

百度智能云播放器 Unity SDK(以下简称“SDK”）是百度智能云推出的适用于 Unity 框架的视频播放器软件开发工具包 (SDK)，为 Unity 开发者提供简单、便捷的开发接口，帮助开发者在 Unity 框架上实现媒体播放功能。SDK 支持 Android 和 iOS 平台，提供简单、便捷的媒体应用开发能力。

- 针对流媒体场景进行优化，支持 RTMP、HTTP+FLV/MP4、HLS 协议及 H264/HEVC 和 AAC/MP3/WANOS 编码格式，适用于点/直播场景。
- 性能强大，CPU/内存占用率低，视频加载速度快，支持高性能4K视频硬件解码和OpenGL/Metal渲染。
- 低门槛、高灵活度实现媒体播放功能，同时提供2D视频和360°全景视频开发示例。
- 版权保护支持百度智能云 PlayerBinding 与 Token 加密方式，同时支持标准 HLS AES 加密视频的播放。
- 支持6DoF空间音效，可打造完全的3D视听体验。

功能列表

- 播放
  - 支持首屏秒开
  - 支持多实例播放
  - 支持单实例多次播放
  - 支持纯音频播放
  - 支持续播
  - 支持精准seek

    - 支持循环播放
    - 支持倍速播放
- 解码
    - 支持硬件解码
- 渲染
    - 支持OpenGL
    - 支持Metal
- HTTP 请求设置
    - 支持设置 HTTP 请求的 Header
    - 支持设置 HTTP 请求的 UserAgent
- 支持 DRM 版权保护
- 支持 SEI 信息更新回调
- 支持 360° 视频播放
- 支持 6DoF 空间音效

SDK集成

开发与运行环境

- Unity Hub
- Unity 2019.4.24 或以上版本
- 支持 Android 4.4 及以上系统版本
- 支持 iOS 9.0 及以上版本

SDK目录结构

下载最新的播放器 Unity SDK，解压后可得到unitypackage，目录结构如下：

```
 ├─── Baidu-Cloud-Unity-Player-<version>.unitypackage
│   ├─── Plugins // 播放内核底层库
│   ├─── Scenes // Demo展示
│   │   ├─── BasicScene.unity // 常规2D视频播放场景展示
│   │   ├─── SkyboxScene.unity // 360°视频播放场景展示
│   │   ├─── DebugUI.cs // Demo UI
│   │   ├─── Player.cs // 对常见的播放功能进行了封装，适用于不需要太多自定义功能的开发者直接使用。开发者也可以参考它进行自定义的扩展。
│   ├─── StreamingAssets // 测试证书目录
│   ├─── UnityXplayer // SDK接口目录
│   │   ├─── Scripts // SDK C# 代码
│   │   ├─── Shaders // SDK shader 代码
```

申请license

申请播放器SDK license：您需要登录百度智能云控制台申请获取高级版播放器SDK license。

配置工程

配置证书 通过百度智能云控制台下载证书，复制到Unity项目的Assets/StreamingAssets目录下。

注意：证书文件名不可修改；证书、应用包名、LicenseID一一对应。 **Android项目配置** 除了SDK包自带的AAR之外，还需要在gradle中添加以下依赖项目：

```
 // retrofit
implementation 'com.squareup.retrofit2:retrofit:2.1.0'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
implementation 'com.squareup.retrofit2:adapter-rxjava:2.1.0'

// okhttp
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
```

Android Target 30适配 如果您的应用设置了Target SDK 30及以上，则需要在AndroidManifest.xml中添加以下内容：

```
 <application android:allowNativeHeapPointerTagging="false">
 ...
</application>
```

iOS项目配置 除了SDK包自带的 framework 和 .a 静态库之外，还需要额外添加libz.tbd 依赖。

快速开始

设置LicenseID（AppId）并初始化播放器

用户在使用SDK之前需要去百度智能云控制台申请并下载.license文件放到自己工程下，并将LicenseID设置给播放器。

LicenseID、包名、证书文件一一对应，所以可能需要按平台设置对应的ID。

```
using BDCloud.MetaMediaSDK;

##### if UNITY_ANDROID
    // android
    private static string appId = "your-android-license-id";
##### elif UNITY_IOS
    // ios
    private static string appId = "your-ios-license-id";
##### endif
private UnityXplayer xplayer;
xplayer = new UnityXplayer(appId);
```

创建Shader和Material

针对Android和iOS平台，需要选择不同的Shader。

```
private Shader playerShader = null;
private Material playerMaterial = null;

##### if UNITY_ANDROID
    playerShader = Shader.Find("Xplayer/AndroidOESShader");
##### else
    playerShader = Shader.Find("Xplayer/YUVShader");
##### endif

playerMaterial = new Material(playerShader);

// 将material设置给你想要显示视频画面的GameObject
private GameObject cube0 = GameObject.Find("Cube0");
if (cube0 != null) {
    cube0.GetComponent<Renderer>().material = playerMaterial;
}
```

设置监听回调

```
// SEI信息回调
class MyUnityXplayerSEIListener : UnityXplayerSEIListener
{
    public override int OnSEI(byte[] buf)
    {
        Debug.Log("OnSEI");
        seiBuf = buf;
        return 0;
    }
}

// 错误码回调，具体错误码含义参考接口与错误码速查页面
private static int errorCode = 0;
class MyUnityXplayerErrorListener : UnityXplayerErrorListener
{
    private UnityXplayer player = null;
    public MyUnityXplayerErrorListener(UnityXplayer player)
    {
        this.player = player;
    }

    public override int OnError(int _errorCode)
    {
        Debug.Log("OnError: " + _errorCode);
        errorCode = _errorCode;

        // 在错误回调中可以自定义处理逻辑，如重新播放
        this.player.Stop();
        this.player.OpenUrl(this.player.GetUrl());

        return 0;
    }
}

// 信息码回调，具体信息码含义参考接口与错误码速查页面
class MyUnityXplayerInfoListener : UnityXplayerInfoListener
{
    public override int OnInfo(int infoCode)
    {
        Debug.Log("OnInfo: " + infoCode);
        return 0;
    }
}

private UnityXplayerSEIListener seiListener = new MyUnityXplayerSEIListener();
private UnityXplayerErrorListener errorListener = new MyUnityXplayerInfoListener();
private UnityXplayerInfoListener infoListener = new MyUnityXplayerErrorListener(xplayer);
```

创建Android平台的显示Surface

对于Android平台，需要提前在渲染线程创建Surface用于绘制视频内容

```
##### if UNITY_ANDROID
    GL.IssuePluginEvent(RenderThreadHandlePtr, CREATE_SURFACE_INTERNAL);
##### endif
```

传入URL并开始播放

```
xplayer.OpenUrl(url);
```

在Update事件函数中渲染上屏并且响应播放监听回调

因为Android平台和iOS平台的渲染机制不同，所以在Update事件函数中也要根据平台做不同的处理。

```
 void Update()
{
// 在iOS平台上，渲染基于Y纹理和UV纹理
##### if UNITY_IOS || UNITY_STANDALONE_OSX
    GL.IssuePluginEvent(UnityXplayerCPP.UnityRenderEvent(xplayer.GetPlayer(), METAL_LOCK));
    if (texY == null) {
        IntPtr ytex = (IntPtr)xplayer.GetYTex(yTexWidth, yTexHeight);
        if (ytex != IntPtr.Zero && ytex != null) {
            texY = Texture2D.CreateExternalTexture (Marshal.ReadInt32(yTexWidth), Marshal.ReadInt32(yTexHeight), TextureFormat.R8, false, false, ytex);
        }
    }

    if (texUV == null) {
        IntPtr uvtex = (IntPtr)xplayer.GetUVTex(uvTexWidth, uvTexHeight);
        if (uvtex != IntPtr.Zero && uvtex != null) {
            texUV = Texture2D.CreateExternalTexture (Marshal.ReadInt32(uvTexWidth), Marshal.ReadInt32(uvTexHeight), TextureFormat.RG16, false, false, uvtex);
        }
    }

    if (texY != null && texUV != null) {
        IntPtr ytex = (IntPtr)xplayer.GetYTex(yTexWidth, yTexHeight);
        IntPtr uvtex = (IntPtr)xplayer.GetUVTex(uvTexWidth, uvTexHeight);
        if (ytex != IntPtr.Zero && ytex != null && uvtex != IntPtr.Zero && uvtex != null) {

            texY.UpdateExternalTexture (ytex);
            texUV.UpdateExternalTexture (uvtex);

            foreach (Renderer renderer in rendererList)
            {
                if (renderer != null) {
                    renderer.sharedMaterial.SetTexture("_YTex", texY);
                    renderer.sharedMaterial.SetTexture("_UVTex", texUV);
                    renderer.sharedMaterial.SetInt("_Type", 3);
                }
            }

            int width = Marshal.ReadInt32(yTexWidth);
            int height = Marshal.ReadInt32(yTexHeight);

            if (renderTexture != null && playerMaterial != null)
            {
                playerMaterial.SetTexture("_YTex", texY);
                playerMaterial.SetTexture("_UVTex", texUV);
                playerMaterial.SetInt("_Type", 3);

                // 判断尺寸发生变化
                if (renderTexture.width != width || renderTexture.height != height)
                {
                    renderTexture.Release();
                    renderTexture.width = width;
                    renderTexture.height = height;
                }

                Graphics.Blit(null, renderTexture, playerMaterial);
            }

        }
    }
##### elif UNITY_ANDROID
    // 在Android平台上渲染基于OES纹理
    GL.IssuePluginEvent(RenderThreadHandlePtr, UPDATE_TEX);
    if (texOES == null) {
        int oestex = xplayer.GetOesTex();
        if (oestex > 0) {
            yTexWidth = xplayer.GetVideoWidth();
            yTexHeight = xplayer.GetVideoHeight();
            texOES = Texture2D.CreateExternalTexture (yTexWidth, yTexHeight,
                                                      TextureFormat.RGBA32, false, false, new System.IntPtr(oestex));
        }
    }

    if (texOES != null) {
        int oestex = xplayer.GetOesTex();
        if (oestex > 0) {
            yTexWidth = xplayer.GetVideoWidth();
            yTexHeight = xplayer.GetVideoHeight();
            if (texOES.width != yTexWidth || texOES.height != yTexHeight) {
                texOES = Texture2D.CreateExternalTexture (yTexWidth, yTexHeight,
                                                          TextureFormat.RGBA32, false, false, new System.IntPtr(oestex));
            }

            texOES.UpdateExternalTexture (new System.IntPtr(oestex));

            foreach (Renderer renderer in rendererList)
            {
                if (renderer != null) {
                    renderer.sharedMaterial.SetTexture("_MainTex", texOES);
                }
            }

            if (renderTexture != null && playerMaterial != null)
            {
                playerMaterial.SetTexture("_MainTex", texOES);

                // 判断尺寸发生变化
```

```
                if (renderTexture.width != yTexWidth || renderTexture.height != yTexHeight)
                {
                    renderTexture.Release();
                    renderTexture.width = yTexWidth;
                    renderTexture.height = yTexHeight;
                }

                Graphics.Blit(null, renderTexture, playerMaterial);
            }

        }
    }
##### endif

    // 处理SEI回调
    byte[] seiBuf = xplayer.GetSEI();
    if (seiBuf != null) {
        if(seiListener != null){
            seiListener.OnSEI(seiBuf);
        }
    }

    // 处理错误回调
    int errorCode = xplayer.GetErrorCode();
    if (errorCode != 0)
    {
        if (errorListener != null)
        {
            errorListener.OnError(errorCode);
        }
    }

    // 处理事件回调
    int infoCode = xplayer.GetInfo();
    if (infoCode != 0)
    {
        if (infoListener != null)
        {
            infoListener.OnInfo(infoCode);
        }
    }
}
```

播放控制

暂停播放

```
xplayer.Pause();
```

恢复播放

```
xplayer.Resume();
```

进度SEEK

```
xplayer.SeekTo(time);
```

倍速播放

```
xplayer.SetSpeed(speed);
```

循环播放

```
xplayer.SetLooping(looping);
```

音量设置

```
xplayer.SetVolume(volume);
```

停止播放并释放资源

```
xplayer.Stop()
```

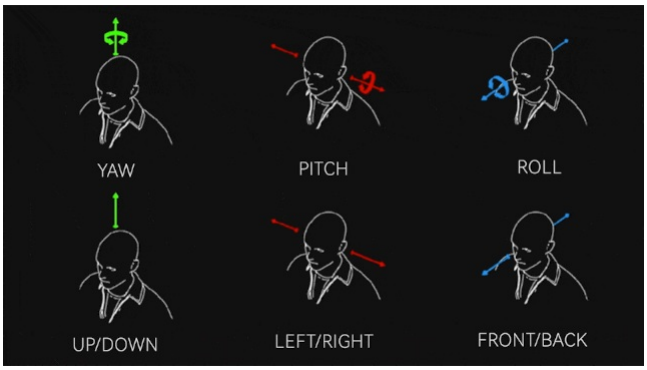
以上流程在Demo工程的`DebugUI.cs`和`Player.cs`中进行了详细的展示，可以参考。

`Player.cs`对常见的播放功能进行了封装，适用于不需要太多自定义功能的开发者直接使用。

快速进阶

空间音频功能接入

SDK支持6DoF空间音频能力，6DoF中的x，y，z参数对应位移信息，yaw，pitch，roll对应旋转信息，如下图所示



使用空间音频能力时，按照以下的流程调用接口

```
// 首先开启音效开关
xplayer.EnableWanos(true);
// 判断6DoF参数范围，其中x左负右正，y后负前正，z下负上正，单位米，无限制范围，模拟真实场景
// yaw：0 ~ 360，头向左转，角度增加；向右转，角度减少
if (yaw < 0.0f || yaw > 360.0f) {
    return;
}
// pitch: -90 ~ 90，抬头为正，低头为负
if (pitch < -90.0f || pitch > 90.0f) {
    return;
}
// roll: -90 ~ 90, 右倾为正，左倾为负
if (roll < -90.0f || roll > 90.0f) {
    return;
}
// 将6DoF数据传给播放器
xplayer.Set6DofData(x, y, z, yaw, pitch, roll);
```

除了设置完整的6DoF信息外，在部分场景下还可以选择更简单的接口，仅传入yaw、pitch参数，此时其他参数自动设置为0。

```
xplayer.SetYawAndPitch(yaw, pitch);
```

播放参数配置

SDK提供UnityXplayerParam类，用于实现定制化的播放参数，包括纯音频播放、从指定时间起播、百度智能云HLS DRM Token设置、传入自定义HTTP header。 使用方式如下

```
// 播放参数需要在OpenURL时传入
UnityXplayerParam playerParam = new UnityXplayerParam();
if (disableVideo) {
    // 关闭视频，仅播放音频
    playerParam.disableVideo = true;
} else {
    playerParam.disableVideo = false;
}
// 设置百度智能云hls drm token
playerParam.decryptTokenForHLS = "87fd2d8aaeefff3023e3d7a6204aa63abf159d2f2968ddac75bfaafec4d4c447_74c1802568074d7bbb02174ca9b81d53_1679973039";
// 设置user agent
playerParam.httpHeaderMap = new Dictionary<string, string>();
playerParam.httpHeaderMap.Add("User-Agent", "something");
// 从指定位置起播，以毫秒为单位
playerParam.initPosInMilliSeconds = 10000;
xplayer.OpenUrl(url, playerParam);
```

接口速查

UnityXplayer类

接口名	描述
public UnityXplayer(String appId)	构造函数
public IntPtr GetPlayer()	返回播放器实例
public int OpenUrl(string url)	开始播放URL 正常起播返回0，鉴权失败时会返回错误码，含义如下 20003：鉴权服务器无返回 20004：找不到证书 20005：证书校验错误 20006：证书过期 20008：证书参数错误 20009：其他系统错误
public int Pause()	暂停播放
public int Resume()	恢复播放
public int IsPlaying()	是否在播放状态中
public int GetDuration()	返回媒体时长，单位毫秒
public int GetCurrentPosition()	返回当面播放位置，单位毫秒
public int SeekTo(int time)	seek到指定位置，单位毫秒
public int Stop()	停止播放，释放资源，重新播放需创建新的实例
public byte[] GetSEI()	获取码流中的SEI信息
public int GetErrorCode()	获取错误码
public int GetInfo()	获取事件码
public int GetStatus()	获取播放器状态码
public int SetVolume(float volume)	设置音量，范围[0, 1]
public int SetSpeed(float speed)	设置播放速度，建议范围[0.25, 2]
public int IsLooping()	是否在循环播放状态中
public int SetLooping(int looping)	设置循环播放
public void EnableWanos(bool enable)	开启/关闭全景声空间音效
public int SetYawAndPitch(float yaw, float pitch)	设置空间音效的yaw和pitch参数，其他参数自动为0
public int Set6DofData(float x, float y, float z, float yaw, float pitch, float roll)	设置空间音效6DoF参数
public int GetVideoWidth()	获取视频宽度
public int GetVideoHeight()	获取视频高度
public IntPtr GetYTex(IntPtr width, IntPtr height)	获取解码后的Y纹理指针和宽高 iOS平台特有
public IntPtr GetUVTex(IntPtr width, IntPtr height)	获取解码后的UV纹理指针和宽高 iOS平台特有
public int CreateSurface()	创建用于显示视频画面的Surface Android平台特有
public int GetOesTex()	获取OES纹理ID Android平台特有
public int DestroySurface()	销毁Surface Android平台特有

常见错误码含义

错误码	含义
-1	操作无权限
-2	本地没有此文件
-5	I/O错误
Android：-110 iOS：-60	网络连接超时 常见原因： - 客户端弱网下产生连接超时 - 服务端返还数据慢导致超时 - http劫持
Android：-111 iOS：-61	连接被拒绝，服务端行为，例如鉴权
Android：-101 iOS：-51	网络不可达，典型原因：断网或者网络切换后，使用原IP继续链接
-541478725	EOF
-808465656	HTTP_BAD_REQUEST
-858797304	网络请求失败（http 403）
-875574520	网络请求失败（http 404）
-1482175992	网络请求失败（http code > 500）
-1094995529	解码信息异常
-1414092869	视频帧读取超时

常见事件码含义

事件码	含义
200	播放器准备就绪
300	播放完成
301	播放停止
400	视频分辨率变化
402	视频渲染开始
403	音频渲染开始
500	缓冲开始
501	缓冲结束
502	缓冲进度回调
600	seek完成
700	播放状态更新回调

状态码含义

状态码	含义
0	初始状态
1	已初始化
2	准备中
3	准备就绪
4	播放已开始
5	播放已暂停
6	播放已完成
7	播放已停止
8	播放错误
9	播放器已释放

版本更新记录

版本	功能描述
3.0.0.1	初版本发布

HarmonyOS NEXT

简介

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 HarmonyOS NEXT编程经验。

简介

百度智能云播放器 HarmonyOS NEXT SDK(以下简称“SDK”）是百度智能云推出的 HarmonyOS 平台视频播放器软件开发工具包 (SDK)，为 HarmonyOS NEXT开发者提供简单、便捷的开发接口，帮助开发者在 HarmonyOS 移动设备上实现媒体播放功能。SDK 提供简单、便捷的媒体应用开发能力。

- 支持广泛的音视频格式

  支持常见的RTMP、HLS（M3U8）、FLV直播场景的视频格式。

  支持常见的HLS（M3U8）、MP4、FLV、TS点播场景的视频格式。

  支持MP3、aac点播场景的音频格式。

- 播放协议

  支持播放 H.264 编码协议的视频流。

  支持播放 H.265 编码协议的视频流。

- 性能强大

  CPU/内存占用率低，视频加载速度快。

- 版权保护

  支持百度智能云 Token 加密方式 播放 HLS 加密视频。

功能列表

SDK集成

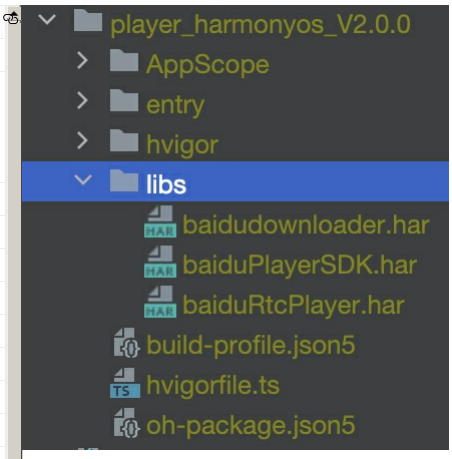
本文为您介绍如何将 HarmonyOS NEXT 播放器 SDK 集成至您的项目中。

SDK&Demo下载

获取SDK&Demo

下载最新的播放器 HarmonyOS NEXT 播放器Demo及SDK，解压后文件目录如下：

分类	功能	说明
播放协议与格式	直播播放	支持常见的RTMP、HLS（M3U8）、FLV直播场景的视频格式
	点播播放	* 支持常见的HLS（M3U8）、MP4、FLV、TS点播场景的视频格式 * 支持MP3、aac点播场景的音频格式
	H.264编码协议	支持播放H.264编码协议的视频流
	H.265编码协议	支持播放H.265编码协议的视频流
	URL播放	支持在线视频、本地视频以URL的方式播放
播放控制	基础控制	支持开始、结束、暂停、seek、循环播放等播放控制功能
	清晰度切换	支持用户流畅无卡顿地切换视频的多路清晰度流。
	Seek	支持拖动到指定位置/快进、后退
	精确Seek	支持精确到帧级别拖动到指定位置
	小窗口播放	支持切换小窗口播放
	全屏、退出全屏	支持全屏、退出全屏操作
	截图	支持截图保存到本地相册
	智能防挡弹幕	支持智能防挡弹幕
	录制	支持录制音视频保存到本地相册
	外挂字幕	支持设置srt\ssa\ass\webvtt格式外挂字幕
视频效果	自定义播放器尺寸	支持自定义设置播放器的宽高
	显示模式	支持填充、铺满、裁剪显示模式
	亮度调节	支持系统的亮度调节
	画中画（小窗）	画中画以小窗形式播放。



说明：

- baiduPlayerSDK.har为HarmonyOS NEXT 播放器 SDK，Demo已集成sdk。
- baidudownloader.har为HarmonyOS NEXT 本地缓存下载 SDK，包括hls、mp4、flv等常见的音视频下载格式，Demo已集成sdk，可参考entry/src/ets/pages/CachePage.ets，不使用此功能可不引入此SDK。
- baiduRtcPlayer.har为HarmonyOS NEXT 超低延迟webrtc直播播放sdk，独立的播放组件，Demo已集成sdk，可参考entry/src/ets/controller/LowLatencyLivePlayer.ets，不使用此功能可不引入此SDK。
- entry为Demo示例代码，编译本Demo时，需要基于自己的华为账号更新项目的签名后，方可运行。
- 该Demo仅供集成SDK时参考，我们计划在后续发布版本中提供更多的演示内容。如果在集成过程中遇到任何问题，请随时与我们联系。

集成准备

环境要求

类别	说明
开发工具	DevEco Studio（推荐使用最新版本）
系统版本	鸿蒙HarmonyOS Next Next.0.0.26及其之后的稳定版本
兼容的最低 SDK 版本	"compatibleSdkVersion": "5.0.0(12)"
手机设备	推荐Huawei Mate 60系列手机，例如Huawei Mate 60 Pro（ALN-AL80）

SDK集成

HarmonyOS NEXT 点播 SDK 可以通过har包本地配置的方式集成，也支持ohpm install方式引入。

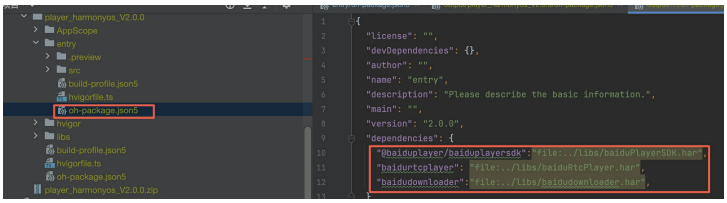
添加依赖 ohpm install 1、安装

```
ohpm install  @baiduplayer/baiduplayersdk
```

2、快速引入

```
import { BDCloudMediaPlayer, OnPreparedListener,InterruptEvent, InterruptHintType } from  '@baiduplayer/baiduplayersdk'
```

本地集成 1、将 har 文件放置于下图所示的路径中。



2、在 entry 下的 oh-package.json5 中添加依赖：

```
{
 "name": "entry",
 "version": "1.0.0",
 "description": "Please describe the basic information.",
 "main": "",
 "author": "",
 "license": "",
 "dependencies": {
   "baiduplayersdk":"file:../libs/baiduPlayerSDK.har",
   "baidurtcplayer": "file:../libs/baiduRtcPlayer.har", // 超低延时直播独立SDK，高级版SDK特有，不使用此功能可不引入
   "baidudownloader":"file:../libs/baidudownloader.har", // 下载独立sdk，可单独引入，不使用此功能可不引入
 }
}
```

3、声明权限 在 enrty 下的 module.json5 文件中声明权限：

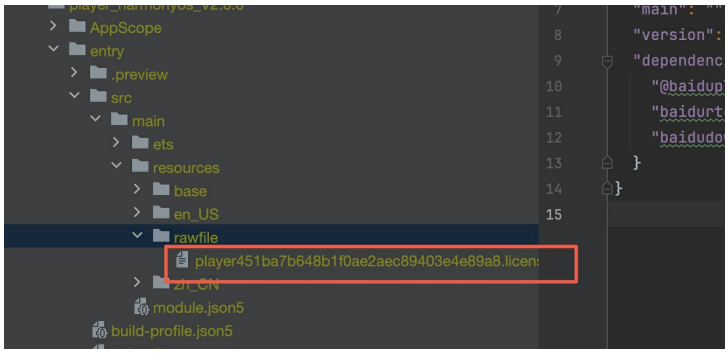
```
  "requestPermissions": [
 {
  // 网络权限，建议添加
  "name": "ohos.permission.INTERNET"
 },
 {
  // 获取 WIFI 信息，如获取 mac，建议添加
  "name": "ohos.permission.GET_WIFI_INFO"
 },
 {
  // 获取网络信息，建议添加
  "name": "ohos.permission.GET_NETWORK_INFO"
 },
 {
  // 资产持久化，建议添加
  "name": "ohos.permission.STORE_PERSISTENT_DATA"
 },
 { // 获取WRITE_MEDIA权限，截图功能使用
  "name": "ohos.permission.WRITE_MEDIA",
  "reason": "$string:app_name",
  "usedScene": {
   "abilities": [
    "FormAbility"
   ],
   "when":"always"
  }
 },
 { // 获取READ_MEDIA权限，截图功能使用
  "name": "ohos.permission.READ_MEDIA",
  "reason": "$string:app_name",
  "usedScene": {
   "abilities": [
    "FormAbility"
   ],
   "when":"always"
  }
 }
]
```

配置证书

1. 申请播放器SDK license： 您需要登录百度智能云控制台申请获取播放器SDK license。

2. 配置证书

下载申请的证书，复制到 entry/src/main/resources/rawfile目录下。



3. 证书鉴权

将.license文件放到自己工程assets目录下后，在使用播放器前需将licenseId设置给播放器。调用BDCloudMediaPlayer的静态方法setAppId来设置appid(licenseId)

```
await BDCloudMediaPlayer.setAppId(licenseId);
```

快速开始

1、导入BDCloudMediaPlayer组件和相关接口

```
  import { BDCloudMediaPlayer, InterruptEvent, InterruptHintType } from  '@baiduplayer/baiduplayersdk'
 import { OnPreparedListener } from '@baiduplayer/baiduplayersdk';
 import { OnVideoSizeChangedListener } from '@baiduplayer/baiduplayersdk';
 import { OnCompletionListener } from '@baiduplayer/baiduplayersdk';
 import { OnBufferingUpdateListener } from '@baiduplayer/baiduplayersdk';
 import { OnErrorListener, OnTimedTextListener } from '@baiduplayer/baiduplayersdk';
 import { OnInfoListener } from '@baiduplayer/baiduplayersdk';
 import { OnSeekCompleteListener } from '@baiduplayer/baiduplayersdk';
```

2、设置AppId(licenseId)

用户在使用SDK之前需要去百度智能云控制台申请并下载.license文件放到自己工程entry/src/main/resources/rawfile目录下，并将licenseId设置给播放器。调用BDCloudMediaPlayer的静态方法setAppId来设置appid(licenseId):

```
 const licenseId = "XXXXp"; // licenseId对应百度智能云控制台申请 License 后的licenseID
await BDCloudMediaPlayer.setAppId(licenseId);
```

3、绑定播放器与鸿蒙XComponent组件，实现视频画面渲染。

```
/**
 * 接入鸿蒙播放器SDK
 */

import { BDCloudMediaPlayer, OnPreparedListener ,InterruptEvent, InterruptHintType } from  '@baiduplayer/baiduplayersdk'

@Preview
@Component
export struct MyPlayerComponent {

  private videoUrl: string = 'http播放地址';
  private xComponentController = new XComponentController();
  private mBDCloudMediaPlayer : BDCloudMediaPlayer | null = null;
  private xComponentId = "xid" + Math.random();
  build() {
    XComponent({
      id: this.xComponentId, // unique XC id
      type:'surface',
      libraryname: 'bdplayer_napi',  //第三方库名称，固定为'bdplayer_napi'
      controller: this.xComponentController
    })
    .onLoad(async (context) => {
      // 实例化播放器，并绑定 XC id
        this.mBDCloudMediaPlayer = new BDCloudMediaPlayer(context, this.xComponentId);

     // 如果是hls token加密片源，先设置token
     // this.mBDCloudMediaPlayer.setDecryptTokenForHLS("your-token");

      // 设置播放的URL
      this.mBDCloudMediaPlayer.setDataSource(this.videoUrl);

      // 准备播放，播放器准备完毕后，会通过onPrepared回调通知，收到回调后调用start即可开始播放
      this.mBDCloudMediaPlayer.prepareAsync();
    })
    .width('100%')
    .height(200)
  }

  aboutToDisappear(): void {
      this.mBDCloudMediaPlayer?.stop();
      this.mBDCloudMediaPlayer?.release()
  }
}
```

4、播放器设置监听

```
  // 播放器已经解析出播放源格式时回调
 let mOnPreparedListener: OnPreparedListener = {
  onPrepared: (mp: BDCloudMediaPlayer) => {
    Logger.info("[PlayVideoModel] onPrepared");
    mp.start();
  }
 }
 this.mBDCloudMediaPlayer.setOnPreparedListener(mOnPreparedListener);

 // 播放进度回调
 let mOnTimedTextListener: OnTimedTextListener = {
  onTimedText: (mp: BDCloudMediaPlayer) => {
  }
 }
 this.mBDCloudMediaPlayer.setOnTimedTextListener(mOnTimedTextListener)

 // 播放完成事件回调
 let mOnCompletionListener: OnCompletionListener = {
  onCompletion: (mp: BDCloudMediaPlayer) => {
    Logger.info("OnCompletionListener-->go")
  }
 }
  this.mBDCloudMediaPlayer.setOnCompletionListener(mOnCompletionListener);

 // 总体加载进度回调，返回为已加载进度占视频总时长的百分比
 let mOnBufferingUpdateListener: OnBufferingUpdateListener = {
  onBufferingUpdate: (mp: BDCloudMediaPlayer, percent: number) => {
    Logger.info("OnBufferingUpdateListener-->go:" + percent);
  }
 }
 this.mBDCloudMediaPlayer.setOnBufferingUpdateListener(mOnBufferingUpdateListener);

// seek快速调节播放位置，完成后回调
 let mOnSeekCompleteListener: OnSeekCompleteListener = {
  onSeekComplete: (mp: BDCloudMediaPlayer) => {
    Logger.info("OnSeekCompleteListener-->go");
  }
 }
 this.mBDCloudMediaPlayer.setOnSeekCompleteListener(mOnSeekCompleteListener);

// 视频宽高变化时回调, 首次解析出播放源的宽高时也会回调
 let mOnVideoSizeChangedListener: OnVideoSizeChangedListener = {
  onVideoSizeChanged: (mp: BDCloudMediaPlayer, width: number, height: number) => {
    Logger.info("onVideoSizeChanged-->go:" + width + "===" + height)
  }
 }
 this.mBDCloudMediaPlayer.setOnVideoSizeChangedListener(mOnVideoSizeChangedListener)

// 播放器信息回调，如缓冲开始、缓冲结束
 let mOnInfoListener: OnInfoListener = {
  onInfo: (mp: BDCloudMediaPlayer, what: number, extra: number) => {
    Logger.info("OnInfoListener-->go:" + what + "===" + extra);
  }
 }
 this.mBDCloudMediaPlayer.setOnInfoListener(mOnInfoListener);

// 错误事件监听
 let mOnErrorListener: OnErrorListener = {
  onError: (mp: BDCloudMediaPlayer, what: number, extra: number) => {
    Logger.info("OnErrorListener-->go:" + what + "===" + extra)
  }
 }
 this.mBDCloudMediaPlayer.setOnErrorListener(mOnErrorListener);
 this.mBDCloudMediaPlayer.setMessageListener();
```

5、设置片源URL并准备播放，播放器准备完毕后，会通过onPrepared回调通知，收到回调后调用start即可开始播放

```
  // 如果是hls token加密片源，先设置token
 this.mBDCloudMediaPlayer.setDecryptTokenForHLS("your-token");
 this.mBDCloudMediaPlayer.setDataSource(this.iUrl);
 this.mBDCloudMediaPlayer.prepareAsync();
```

6、播放控制接口

```
  //开始播放
 this.mBDCloudMediaPlayer.start();
 // 暂停
 this.mBDCloudMediaPlayer.pause();
 // 停止
 this.mBDCloudMediaPlayer.stop();
 // seek
 this.mBDCloudMediaPlayer.seekTo(msec);
 // 倍速
 this.mBDCloudMediaPlayer.setSpeed("2f");
 // 循环
 this.mBDCloudMediaPlayer.setLoopCount(true);
 // 音量调节
 this.mBDCloudMediaPlayer.setVolume(leftVolume, rightVolume);
```

7、销毁

```
  this.mBDCloudMediaPlayer.release();
```

快速进阶

播放控制条

简单播放控制条可参考demo中的PlayControl类。该类由播放按钮、播放进度条等组成。 涉及到的接口有：

接口名	参数	返回值	说明
start	无	void	开始/恢复播放
pause	无	void	暂停播放
isPlaying	无	boolean	查看是否正在播放状态
seekTo	msec: string	void	seek到指定位置播放
setVolume	leftVolume: string,rightVolume: string	void	设置音量
setSpeed	speed: string	void	设置播放倍速
getSpeed	无	number	获取设置的倍速
getDuration	无	number	获取视频时长，单位为毫秒
getCurrentPosition	无	number	获取当前播放位置，单位为毫秒
setLoopCount	looping: boolean	void	设置循环播放
isLooping	无	boolean	查看当前是否循环播放
setOnBufferingUpdateListener	listener: OnBufferingUpdateListener	void	监听，回调时返回已缓存时长占视频播放时长的百分比，根据该值更新二级进度条（缓存进度）

暂停与播放

```
 // 暂停播放

this.mBDCloudMediaPlayer.pause()

// 恢复播放

this.mBDCloudMediaPlayer.start()

// 查看是否是播放状态

this.mBDCloudMediaPlayer.isPlaying
```

Seek 到指定位置播放

```
 // 演示 seek 到 1 秒的位置
this.mBDCloudMediaPlayer.start('1000')
```

设置音量 // 静音 this.mBDCloudMediaPlayer.setVolume('0','0') // 取消静音 this.mBDCloudMediaPlayer.setVolume('1','1')

倍速播放 // 设置倍速 可取值：'0.75f'、 '1.0f'、'1.5f' 、'1.75f'、'2.0f' this.mBDCloudMediaPlayer.setSpeed('1.5f'); // 获取倍速 this.mBDCloudMediaPlayer.getSpeed()

获取视频时长 // 获取播放的总时长 this.mBDCloudMediaPlayer.getDuration()

```
 // 获取播放当前时长
this.mBDCloudMediaPlayer.getCurrentPosition()
```

循环播放 // 设置循环播放 this.mBDCloudMediaPlayer.setLoopCount(true)

```
 // 查看当前是否循环播放
this.mBDCloudMediaPlayer.isLooping()
```

多码率有感切换

当播放的是HLS多码率视频时，播放器支持在播放过程中实时切换码率。 **获取多码率视频(Master M3U8)的index数目**

```
this.mBDCloudMediaPlayer?.getVariantInfo() as  Array<string>;
```

- 该variantinfo直接从Master m3u8文件中取值，根据(#EXT-X-STREAM-INF)格式的不同，取到的值 可能为：1920x1080,3541000也可能为,232370(不规范的视频)。逗号前是分辨率，逗号后是码率；
- 数组的大小即为多码率的数目；

选择多码率 播放过程中，选择多码率：

```
this.mBDCloudMediaPlayer?.selectResolutionByIndex(index);
```

- 该函数会使得player内部的状态变化stop --> prepareAsync --> prepared。
- BDCloudMediaPlayer还开放了selectVariantByIndex接口。该接口不帮忙维护mediaplayer状态。如果想使用该接口，需要保证在prepareAsync之前调用。

多码率无缝切换 播放器不仅支持HLS的多码率快速切换，同时也支持MP4等主流媒体格式的无缝切换功能。具体操作如下：

Master playlist 的HLS格式多码率无缝切换

1. 设置输入格式: this.mBDCloudMediaPlayer?.setMediaInputType(mode)。
2. 在监听到onPrepared状态后，调用this.mBDCloudMediaPlayer?.getMediaItems()获取多码率信息列表，更新UI。
3. 用户根据对应索引切换到指定码率: this.mBDCloudMediaPlayer?.selectMediaByIndex(index)。
4. 当播放器缓冲播放完成后即会切换到新的码率显示会收到onVideoSizeChanged(...) 回调表示视频分辨率发生变化。

MP4等非嵌套码率的无缝切换

1. 设置输入格式: this.mBDCloudMediaPlayer?.setMediaInputType(mode)。

2. 设置多码率视频链接（注意这里和HLS的区别）：this.mBDCloudMediaPlayer?.setMediaItems(mediaItems)。

3. 在onPrepared状态后，调用getVariantInfo()获取多码率信息列表，更新UI。

4. 用户根据对应索引切换到指定码this.mBDCloudMediaPlayer?.selectMediaByIndex(index)。

5. 当播放器缓冲播放完成后即会切换到新的码率显示会收到收到onVideoSizeChanged(...)回调表示视频分辨率发生变化。

播放HLS加密视频

百度智能云MCP服务和VideoWorks服务支持转码成HLS加密视频。

不同的加密方式，播放时的方法略有不同：

- PlayerBinding加密模式

  按普通视频播放即可，必须使用百度播放器

- Token模式

  token需要您的服务器与百度智能云服务器合作来生成，您的App从您的服务器拿到token后，设置给播放器即可。 prepare之前需要先设置token:

  this.mBDCloudMediaPlayer.setDecryptTokenForHLS("your-token");

视频下载 播放器支持对M3U8、mp4、flv、mp3等常见的音视频进行下载和管理。

下载管理器 1、oh-package.json5添加sdk依赖

```
"dependencies": {
 "baidudownloader":"file:../libs/baidudownloader.har",
}
```

2、项目中引入baidudownloader组件sdk：

```
import { IDownloadEngine, Download, DownloadEventListener, DownloadProgress, DownloadResult ,CacheItem,CacheManager,LocalHlsSec} from 'baidudownloader';
```

3、下载管理器以单例的形式提供：

```
private downloadEngine: IDownloadEngine = Download.getEngine();
```

使用步骤 1、调用downloadEngine: IDownloadEngine = Download.getEngine();获得下载管理单例。

2、调用downloadEngine.downloadVideo(VideoDownloadOptions)启动下载任务。

```
  await this.downloadEngine.downloadVideo({
        url: "XX.m3u8",
        title: "XX.m3u8",
        thumbnail: $r('app.media.ic_internet'),
        enableHLS: true,
        enableResume: false,
        drmToken: "XXX"
  } as VideoDownloadOption);
```

VideoDownloadOption接口说明

```
interface VideoDownloadOptions extends DownloadOptions {
  enableHLS?: boolean;   // 是否启用HLS下载（可选，默认自动检测）
  enableResume?: boolean;  // 是否启用断点续传（可选，默认true）
  chunkSize?: number;    // 分块大小，字节（可选，默认2MB）
  drmToken?: string;     // DRM加密token，用于HLS加密视频下载（可选）
  accountId?: string;    // 账号ID（可选），用于用户数据隔离。传了则按账号隔离存储，不传则按原有方式共享存储
}
```

DownloadOptions接口说明

```
interface DownloadOptions {
  url: string;                  // 视频URL（必需）
  title: string;                // 视频标题（必需）
  thumbnail?: Resource;           // 缩略图资源（可选）
  quality?: string;             // 清晰度：'720p' | '1080p' | '4K'（可选）
  headers?: Record<string, string>; // HTTP请求头（可选）
  timeout?: number;               // 超时时间，毫秒（可选）
}
```

3、设置监听事件

```
// 下载进度更新
const progressHandler = (progress: DownloadProgress) => {
};
// 下载完成
const completedHandler = (result: DownloadResult) => {

};
const errorHandler = (error: string) => {
  console.error('下载错误:', error);
};
class DownloadListenerImpl implements DownloadEventListener {
  onProgress = progressHandler;
  onCompleted = completedHandler;
  onError = errorHandler;
}
this.downloadEngine.addEventListener(new DownloadListenerImpl());
```

DownloadProgress接口

```
interface DownloadProgress {
 id: string;
 progress: number;       // 0-100
 downloadSpeed: string;   // '1.5MB/s'
 downloadedSize: number;
 totalSize: number;
 status: 'downloading' | 'paused' | 'completed' | 'error' | 'pending';
}
```

DownloadResult接口

```
interface DownloadResult {
 id: string;
 status: 'success' | 'failed' | 'cancelled';
 localPath?: string;
 fileSize?: number;
 duration?: number;
 error?: string;
}
```

4、相关方法及接口说明

```
  /**
  * 暂停下载
  */
this.downloadEngine.pauseDownload(item.id);

 /**
  * 恢复下载
  */
this.downloadEngine.resumeDownload(item.id);

 /**
  * 取消下载
  */
this.downloadEngine.cancelDownload(item.id); //  取消下载
 /**
  * 获取所有下载任务
  * @param accountId 账号ID（可选），传入则只返回该账号的下载任务，不传则返回所有
  */
this.downloadEngine.getAllDownloads(accountId?: string): Promise<DownloadProgress[]>;
 /**
  * 删除缓存
  * @param id 缓存ID
  * @param accountId 账号ID（可选），传入则验证该缓存是否属于该账号，防止误删其他账号的缓存
  */
this.downloadEngine.deleteCache(id: string, accountId?: string): Promise<boolean>;

 /**
  * 清空所有缓存
  * @param accountId 账号ID（可选），传入则只清空该账号的缓存，不传则清空共享缓存
  */
 clearAllCache(accountId?: string): Promise<void>;

 /**
  * 获取缓存大小
  * @param accountId 账号ID（可选），传入则统计该账号的缓存大小，不传则统计共享缓存大小
  */
 getCacheSize(accountId?: string): Promise<number>;

 /**
  * 检查是否已缓存
  * @param url 视频URL
  * @param accountId 账号ID（可选），传入则检查该账号是否已缓存，不传则检查共享缓存
  */
 isCached(url: string, accountId?: string): Promise<boolean>;
```

相关demo示例在src/main/ets/views/HomeTabContentListItem.ets和entry/src/main/ets/pages/CachePage.ets。

边播边录制

```
  // 是否正在录制中
let isRecord = this.mBDCloudMediaPlayer?.isRecord();

// 开始录制
this.mBDCloudMediaPlayer.startRecord(outputfd:number)

// 结束录制
await this.mBDCloudMediaPlayer?.stopRecord()
```

使用方式详见demo中VideoController.ets，录制完成保存到相册

外挂字幕 通过下面的接口可以添加外挂字幕，当前支持的字幕格式包括srt\ssa\ass\webvtt。需要注意的是，该接口需要在收到播放器onPrepared回调后调用，并且对于同一个播放器，同一时刻仅能添加一个外挂字幕轨道，添加新的外挂字幕会自动代替旧的外挂字幕

```
 // subtitleSrc对应外挂字幕地址
this.mBDCloudMediaPlayer.addExtSubtitleUrl(subtitleSrc:string);
```

在播放内核中，当外挂字幕成功读取后，会通过下面的回调进行通知

```
  let mOnExtSubtitleOpenListener:OnExtSubtitleOpenListener = {
  onExtSubtitleOpen:(info:string) => {
    console.log(`外挂字幕url：${info}`)
    promptAction.showToast({
      duration: PlayConstants.PLAY_ERROR_TIME,
      message: "外挂字幕已开启"
    });
  }
}
```

具体的字幕内容也会在OnTimedTextListener中回调，这一点与内嵌字幕相同。

```
  let mOnTimedTextListener: OnTimedTextListener = {
  onTimedText: (mp: BDCloudMediaPlayer,text:string) => {
    Logger.info("[PlayVideoModel] onTimedText"+text);
    this.context.eventHub.emit(Events.SUBTITLE_TEXT,text)
  }
}
this.mBDCloudMediaPlayer.setOnTimedTextListener(mOnTimedTextListener)
```

高级版功能接入

超低延时直播功能接入

接入准备 接入超低延时直播功能，需要使用播放器SDK高级版，并申请高级版License。

功能介绍 在高级版SDK中，提供了超低延时直播流的播放能力，该能力由baidurtcplayer.har SDK提供，请确保该组件已集成到你的App中。

SDK当前支持的音视频编码格式如下：

- 视频：H.264
- 音频：AAC

Demo体验

前往SDK简介与下载页面扫码安装DEMO

快速开始

1. 播放器引入

```
  import { BRTCPlayerImpl，BRTCPlayerParameters,BRTCPlayerEvent,MediaStreamTrack,VideoRenderController,ScalingMode} from 'baidurtcplayer'
```

2. 创建播放器对象及播放器初始化

```
  const LicenseID = "XXp";
// 创建播放器对象，需要传入您申请的高级版证书LicenseID，ID可以在百度智能云控制台(https://console.bce.baidu.com/bvc/#/bvc/player-license/list)查看
this.brtcPlayerImpl = new BRTCPlayerImpl(LicenseID)
let config:BRTCPlayerParameters = {
  pullUrl:"webrtc://by-test.bj-webrtc-pl001.bigenemy.cn/myapp/test_realtime1"
}
this.brtcPlayerImpl.on(BRTCPlayerEvent.REMOTE_STREAM_ADD,(track:MediaStreamTrack)=>{
  if(track.kind == 'video'){
    this.LocalXComponentController.setVideoTrack(track);
    this.LocalXComponentController.setScalingMode(ScalingMode.AspectFit)
  }
})
this.brtcPlayerImpl.initPlayer(config)
```

3. 播放控制

```
  this.brtcPlayerImpl?.play() // 开始播放
this.brtcPlayerImpl?.pause() // 暂停播放
this.brtcPlayerImpl?.releasePlayer() //销毁
this.LocalXComponentController.setScalingMode(ScalingMode.AspectFit) //设置显示模式
```

4. 播放回调事件

```
  // 流添加
 this.brtcPlayerImpl.on(BRTCPlayerEvent.REMOTE_STREAM_ADD,(track:MediaStreamTrack)=>{
       if(track.kind == 'video'){
         this.LocalXComponentController.setVideoTrack(track);
           this.LocalXComponentController.setScalingMode(ScalingMode.AspectFit)
       }
 })
```

5. 释放播放器

```
  this.brtcPlayerImpl?.releasePlayer() //销毁
```

端上超分功能接入

接入准备 接入端上超分功能，需要使用播放器SDK高级版（也可以单独接入端上超分SDK），并申请高级版License。

功能介绍 在高级版SDK中，提供了端上超分能力，利用端侧推理能力，实现对低分辨率画面的清晰度提升、噪声和块效应去除，适用于视频播放场景场景。

该能力目前集成在播放器sdkbaiduPlayerSDK.har，如何需要单独引入超分sdk，可提工单支持。

功能	说明
支持的超分倍数	固定2倍
支持的输入分辨率	无限制

Demo体验

前往SDK简介与下载页面下载运行DEMO

快速开始

播放器中集成超分

```
this.mBDCloudMediaPlayer?.startSrkit()  // 开启超分
this.mBDCloudMediaPlayer?.stopSrkit() // 关闭超分
```

可以参考demo实现ets/controller/VideoController.ets。

接口速查

BDCloudMediaPlayer类

接口名	参数	返回值	说明
BDCloudMediaPlayer	context: object, id?: string	BDCloudMediaPlayer	构造函数，设置XComponent回调的context, 设置XComponent的id属性值(可选)
setDebug	open: boolean	void	设置日志开关
setDataSource	url: string	void	设置视频源地址
setDataSourceHeader	headers: Map<string, string>	void	设置视频源的HTTP请求头
setDecryptTokenForHLS	token: string	void	设置HLS加密流的token
prepareAsync	无	void	加载视频
start	无	void	播放视频
stop	无	void	停止播放
pause	无	void	暂停播放
reset	无	void	视频重置
release	无	void	释放资源
seekTo	msec： string	void	快进、后退
setSpeed	speed: string	void	设置播放倍速
getSpeed	无	number	获取设置的倍速
isPlaying	无	boolean	查看是否正在播放状态
setOnVideoSizeChangedListener	listener: OnVideoSizeChangedListener	void	设置获取视频宽高回调监听
setOnPreparedListener	listener: OnPreparedListener	void	设置视频准备就绪回调监听
setOnInfoListener	listener: OnInfoListener	void	设置播放器的各种状态回调监听
setOnErrorListener	listener: OnErrorListener	void	设置播放异常回调监听
setOnBufferingUpdateListener	listener: OnBufferingUpdateListener	void	设置buffer缓冲回调监听
setOnSeekCompleteListener	listener: OnSeekCompleteListener	void	设置快进后退回调监听
setMessageListener	无	void	设置视频监听器到napi用于接收回调
getVideoWidth	无	number	获取视频宽度
getVideoHeight	无	number	获取视频高度
getDuration	无	number	获取视频总的时长
getCurrentPosition	无	number	获取视频播放当前位置
setVolume	leftVolume: string,rightVolume:string	void	设置音量
setLoopCount	looping: boolean	void	设置循环播放
isLooping	无	boolean	查看当前是否循环播放
getMediaInfo	无	object	获取媒体信息
isRecord	无	object	是否处于录制状态
startRecord	outputfd	无	开始录制
stopRecord	无	无	结束录制
addExtSubtitleUrl	url	无	添加外挂字幕url，可支持格式srt\ssa\ass\webvtt
setInitPlayPosition	string	无	设置起播开始时间，默认为0

版本更新记录

版本	功能描述
v1.0.0	- HarmonyOS NEXT 播放器 SDK 首个版本
v1.1.0	- 新增截图 -新增画中画能力展示 -弹幕 -设置填充、裁剪、铺满显示模式、短视频播放等功能 -修复部分已知问题
v1.2.0	-新增录制 -外挂字幕 -智能防挡弹幕等功能 -修复部分已知问题
v2.0.0	-新增支持清晰度切换 -新增支持下载功能 -新增端上超分能力 -新增支持webrtc超低延迟播放 -设置页面添加横屏/竖屏、循环播放、质感超清配置、添加资源等配置项 -license校验优化 -修复部分已知问题

uniapp播放器

简介

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 Vue.js 编程经验。

简介

uniapp 播放器 SDK 是基于 Android 和 iOS 原生播放器 SDK 的统一封装解决方案，旨在为开发者提供跨平台的播放器集成能力。SDK 支持 Android 和 iOS 平台，提供简单、便捷的媒体应用开发能力。

- 支持广泛的音视频格式

  支持常见的RTMP、HLS（M3U8）、FLV直播场景的视频格式。

  支持常见的HLS（M3U8）、MP4、FLV、TS点播场景的视频格式。

  支持MP3、aac点播场景的音频格式。

- 播放协议

  支持播放 H.264 编码协议的视频流。

  支持播放 H.265 编码协议的视频流。

- 性能强大

CPU/内存占用率低，视频加载速度快。

- 版权保护

支持百度智能云 Token 加密方式 播放 HLS 加密视频。

功能列表

分类	功能	说明
播放协议与格式	直播播放	支持常见的RTMP、HLS（M3U8）、FLV直播场景的视频格式
	点播播放	* 支持常见的HLS（M3U8）、MP4、FLV、TS点播场景的视频格式 * 支持MP3、aac点播场景的音频格式
	H.264编码协议	支持播放H.264编码协议的视频流
	H.265编码协议	支持播放H.265编码协议的视频流
	URL播放	支持在线视频、本地视频以URL的方式播放
播放控制	基础控制	支持开始、结束、暂停、seek、循环播放等播放控制功能
	Seek	支持拖动到指定位置/快进、后退
	精确Seek	支持精确到帧级别拖动到指定位置
音频效果	音量调节	支持调用系统接口调节观看视频的音量
	静音	支持开启和关闭静音功能
视频安全	token加密	支持百度智能云私有化token加密播放hls封装格式音视频流
质量服务	日志上报	支持上报播放器SDK日志，统计音视频点播、直播相关播放埋点信息
	事件回调	支持对视频准备就绪回调、播放状态回调、播放完成、seek完成、音频中断事件或失败回调

SDK集成

开发环境

- 开发环境：HbuilderX，推荐使用最新版本。在 DCloud 开发者中心注册后登录 HBuilderX 编辑器。

获取 License

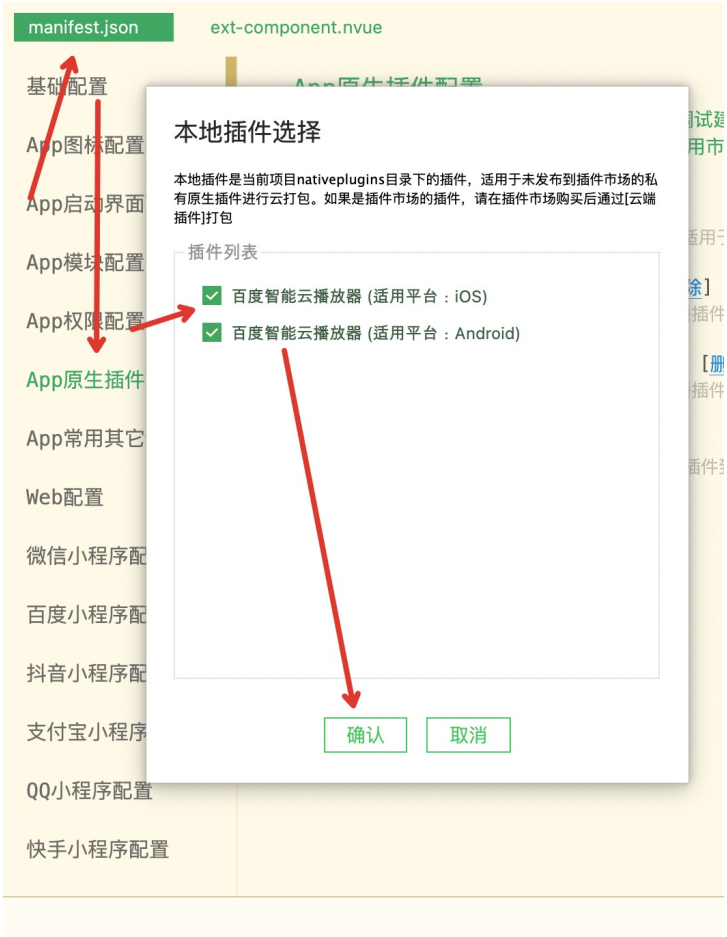
视频播放通过 License 管理播放器 SDK，使用播放器前需要在百度智能云播放器控制台申请license。您需完成以下操作：

- 登录百度智能云控制台申请获取播放器SDK license

集成步骤

集成 uni-app 原生插件

1. 将nativeplugins原生插件下载至本地
2. 将文件解压至 uni-app 项目的中。
3. 通过 HBuilder 打开 uni-app 项目。
4. 在项目根目录下 manifest.json文件的 App 原生插件配置项下单击选择本地插件，然后在列表中选择需要打包生效的插件，如下图所示：



5.demo参考,可点击下载 **添加 License 文件** 从百度智能云播放器控制台下载 License 文件

后，将其作为原生资源文件， 放入 uni-app 项目的 nativeResources 文件夹下（需自行创建），如下图所示。

快速开始

创建播放器 播放器控件为 bd-player，仅允许在 nvue 中声明使用。播放器为纯播放器无状态栏皮肤，开发者可自行添加。可参考demo。

```
<template>
  <bd-player ref="bdplayerContainer"  class="vod-player"
             :file="defaultConfig.file"
             :iosAppId="defaultConfig.iosAppId"
             :androidAppId="defaultConfig.androidAppId"
             :bundleId="defaultConfig.bundleId"
             @onPlayStatus="onPlayStatus"
             @onVideoSizeChanged="onVideoSizeChanged"
             @onBufferingUpdate="onBufferingUpdate"
             @onPlayError="onPlayError"
             @onLevelUpdate="onLevelUpdate">
  </bd-player>
</template>
```

自定义属性

file

- 类型：string
- 描述：播放url，ios端必须设置。

iosAppId

- 类型：string
- 描述：ios绑定的appId，百度智能云播放器控制台申请播放器license后对应的licenseID

androidAppId

- 类型：string
- 描述：android绑定的appId，百度智能云播放器控制台申请播放器license后对应的licenseID

bundleId

- 类型：string
- 描述：ios绑定的bundleId，百度智能云播放器控制台申请播放器license对应Bundle ID。用于license校验

调用setUp初始化播放器

```
this.$refs.bdplayerContainer.setUp(playConfig)
```

playConfig初始化配置参数详细说明如下表所示：

参数	是否必填	类型	默认值	说明
file	是	String	无	播放链接
loop	否	Boolean	false	是否开启循环播放
token	否	String	无	播放drm token加密必传参数
initPlayPosition	否	Number	无	设置播放开始时间，单位为秒

播放控制 // 开始播放或者继续播放均使用start接口。 this.$refs.bdplayerContainer.start(); // 暂停 this.$refs.bdplayerContainer.pause(); // seek到某个时间点，跳转到当前音视频播放的时间，单位秒，必须大于等于 0 this.$refs.bdplayerContainer.seekTo({ seconds: time }); // 释放后，重新播放需创建新的player。 this.$refs.bdplayerContainer.stop();

快速进阶

播放控制接口

```
 // 开始播放或者继续播放均使用start接口。
this.$refs.bdplayerContainer.start();
// 暂停
this.$refs.bdplayerContainer.pause();
// seek到某个时间点，跳转到当前音视频播放的时间，单位秒，必须大于等于 0
this.$refs.bdplayerContainer.seekTo({
    seconds: time
});
// 释放后，重新播放需创建新的player。
this.$refs.bdplayerContainer.stop();
```

获取当前播放状态

```
  this.$refs.bdplayerContainer.isPlaying(null,(res)=>{
    const { isPlaying } = res;
    const title = isPlaying ? "正在播放中" : "已暂停";
 });
```

获取音视频时长

```
 this.$refs.bdplayerContainer.getDuration(null, ret => {
   // 单位：s
  console.log(ret.duration)
});
```

获取当前播放时间 this.$refs.bdplayerContainer.getCurrentPosition(null,ret=>{ // 单位：ms console.log(ret.currentPosition) })

设置/获取音量 // volume：number类型，音量大小，取值范围 0 ~ 1 this.$refs.bdplayerContainer.setVolume({ volume: 0.5 }); this.$refs.bdplayerContainer.getVolume(null, ret => { const {volume} = ret });

设置/获取倍速

```
 // speed: number类型，取值范围 0.5 ~ 2.0，默认值 1.0
this.$refs.bdplayerContainer.setSpeed({
    speed:1.5
});
this.$refs.bdplayerContainer.getSpeed(null, ret => {
    const {speed} = ret
});
```

切换屏幕方向到横/竖屏 isFull ? this.$refs.bdplayerContainer.changeToLandscape() : this.$refs.bdplayerContainer.changeToPortrait(); **截图** 保存播放器播放视频当前画面截图到相册。要开启相册权限。

```
this.$refs.bdplayerContainer.snapshot(null, result => {
  if (result.success) {
      uni.showToast({ title: '播放器截屏成功' });
    } else {
      uni.showToast({ title: '失败: ' + result.error, icon: 'none' });
  }
});
```

设置画面填充模式

```
this.$refs.bdplayerContainer.setScalingMode({scalingMode:1});
```

scalingMode

- 类型：number类型
- 描述：（可选项）拉伸模式， 取值范围1-填充， 2-裁剪， 3-铺满。默认为1-填充。 **切换清晰度** this.$refs.bdplayerContainer.changeLevel({ level:index },(res)=> { const { errMsg } = res; if(!errMsg){ this.isChangeLevel = true; uni.showToast({ title: "正在切换清晰度...", icon: "none" }) } }); **level**
- 类型：整型数字
- 描述：（可选项）指定码率，取值范围 1， 2， 3。level值为：onLevelUpdate事件返回清晰度下标 **播放私有加密视频** this.$refs.bdplayerContainer.setUp({ file: this.defaultConfig.file, token:this.defaultConfig.token, }, (ret) => { this.text = JSON.stringify(ret); if (ret.errMsg != null) { uni.showToast({ title: ret.errMsg, icon: "none" }) } }); **token**

  token需要您的服务器与百度智能云服务器合作来生成，您的App从您的服务器拿到token后，设置给播放器即可。 **播放自定义事件监听 播放状态变化事件监听 onPlayStatus**

playerState

- 类型：字符串
- 描述：「prepared-已准备完成， pause-暂停， stop-停止，playing-播放」

seek

类型：字符串 描述：「complete-seek完成」

```
onPlayStatus(e) {
    const {
        skin,
        bdPlayer
    } = this;
    const state = e.detail.playerState;
    const preparedToPlay = e.detail.preparedToPlay;
    const seekstate =  e.detail.seek;
    if (state != null) {
        this.skin.changePlayStatus(state === 'playing');
        if(state == "prepared"){
            uni.showToast({
                title: "视频已准备就绪",
                icon: "none"
            })
        }else if(state == "complete"){
            console.log("播放完成");
             uni.showToast({
                title: "播放完成",
                icon: "none"
             })
        }else if(state == "pause"){
            uni.showToast({
                title: "视频已暂停",
                icon: "none"
            })
        }else if(state == "playing"){
            uni.showToast({
                title: "视频已播放",
                icon: "none"
            })
        }
    } else if (preparedToPlay != null) {
        this.updateDuration();
    }else if(seekstate != null){
        if(seekstate == "complete"){
            uni.showToast({
                title: "seek完成",
                icon: "none"
            })
        }
    }
}
```

视频分辨率改变 onVideoSizeChanged 视频分辨率改变通知

width

- 类型：number
- 描述：返回当前视频宽度

height

- 类型：number
- 描述：返回当前视频高度

```
onVideoSizeChanged(e){
  const {width,height} = e.detail;
  if(this.isChangeLevel){
      uni.showToast({
          title: "清晰度切换成功",
          icon: "none"
      })
  }
  console.log(`width is ${width},height is ${height}`);
}
```

当前播放视频所支持码率 onLevelUpdate

resolution

- 类型：array
- 描述：返回当前播放视频所支持码率列表。参数["360p","720p"]

```
onLevelUpdate(e){
  const {resolution} = e.detail;
  this.$refs.skin.updateLevels(resolution);
  console.log(`resolution  is ${resolution}`,);
}
```

错误事件 onPlayError

error

- 类型：string
- 描述：错误信息

```
onPlayError(e){
  const {error} = e.detail;
}
```

版本更新记录

版本	功能描述
v1.0.0	uniapp 播放器 首个版本

SDK&Demo下载

合规指南

- 开发者： 北京百度网讯科技有限公司
- SDK名称： 百度智能云播放器SDK
- 版本号： 见各平台SDK版本更新记录
- 主要功能： android端、IOS端、Web端、HarmonyOS端播放器SDK，在各类终端设备上实现媒体播放功能
- 个人信息处理规则： 见 播放器SDK隐私政策
- 合规使用说明：见 播放器SDK开发者个人信息保护合规指引

下载SDK&Demo

本文档提供最新版本的SDK和Demo。如果您需要老版本，请联系技术支持获取。

平台与版本	SDK文件名	下载
Android流媒体标准版 包名： com.baidu.cloud.media.player MD5： 97dd429ba58f045b4ed8d58f99c398d4	Baidu-Cloud-Player-Android-LITE-3.9.0.zip	点击下载
Android流媒体高级版 包名：com.baidu.cloud.media.player MD5： 4c20c46ae2b6f2ebc205b150f863240b	Baidu-Cloud-Player-Android-LITE-Advance-3.9.0.zip	点击下载
Android全媒体标准版 包名：com.baidu.cloud.media.player MD5：7b85c10308dee6fee501a840207878b4	Baidu-Cloud-Player-Android-FULL-3.9.0.zip	点击下载
Android全媒体高级版 包名：com.baidu.cloud.media.player MD5：2036f0029843b72cad72a0804f0c8b1d	Baidu-Cloud-Player-Android-FULL-Advance-3.9.0.zip	点击下载
iOS流媒体标准版 包名：com.baidu.cloud.BDCloudMediaPlayer MD5：5a837d9fd5e49f9b3108819d39247054	Baidu-Cloud-Player-iOS-LITE-3.9.0.zip	点击下载
iOS流媒体高级版 包名：com.baidu.cloud.BDCloudMediaPlayer MD5：3f2afac4266d4eda09db0cdc59567634	Baidu-Cloud-Player-iOS-LITE-Advance-3.9.0.zip	点击下载
iOS全媒体标准版 包名：com.baidu.cloud.BDCloudMediaPlayer MD5：3e4ebe539bd83117a6510a2eac73d9e2	Baidu-Cloud-Player-iOS-FULL-3.9.0.zip	点击下载
iOS全媒体高级版 包名：com.baidu.cloud.BDCloudMediaPlayer MD5：4c45035960d47e79a77b4f055fc47224	Baidu-Cloud-Player-iOS-FULL-Advance-3.9.0.zip	点击下载
Web	cyberplayer_v4.4.5.1_80224d8.zip	点击下载
Unity	BaiduCloudUnityPlayerSDK.zip	点击下载
HarmonyOS NEXT流媒体标准版	player_harmonyos_V2.0.0_e934e27.zip	点击下载
uniapp播放器	uniBDPlayerDemo.zip	点击下载

相关协议

播放器SDK隐私政策

欢迎使用播放器软件开发工具包（SDK）（简称"播放器 SDK"）服务！

播放器 SDK 是一款为移动应用开发者（以下简称"开发者"）提供简单、便捷的开发接口，帮助开发者在移动设备上实现视频播放功能（SDK）。开发者在其移动应用内集成播放器 SDK后，可通过播放器 SDK平台向其移动应用（以下简称"开发者应用"）的最终用户（以下简称"最终用户"）提供本隐私政策所说明的功能及服务。开发者在其移动应用集成并使用播放器 SDK服务时，委托播放器 SDK处理开发者应用相关数据信息，其中可能包括开发者应用最终用户（以下简称"最终用户"）的个人信息。此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策"附录1：名词解释"来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用播放器 SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用播放器 SDK服务的情况，以及播放器 SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于播放器 SDK收集、使用最终用户相关个人信息的完整、合法、在使用播放器 SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于播放器 SDK收集、使用最终用户相关个人信息的完整、合法、在使用播放器 SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

（4）关于上述承诺的具体落地执行可参考《播放器 SDK开发者个人信息保护合规指引》。

2、我们希望集成并使用播放器 SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的播放器 SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，播放器 SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用播放器 SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订
8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用播放器 SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌（必选）	采用加密传输的安全处理方式	iOS及Android
2	区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号（必选）	采用加密传输的安全处理方式	iOS及Android
3	区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本（必选）	采用加密传输的安全处理方式	iOS及Android
4	区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息（必选）	采用加密传输的安全处理方式	iOS及Android
5	区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况（必选）	采用加密传输的安全处理方式	iOS及Android
6	区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息（必选）	采用加密传输的安全处理方式	iOS及Android
7	区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率（必选）	采用加密传输的安全处理方式	iOS及Android

2. 设备权限调用

为了保证最终用户能正常使用播放器 SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能，各项功能及功能对设备权限的调用情况如下：

Android系统版本

设备权限	功能及服务	权限授权方式
相册	截图/录制功能将图片/视频存储到相册	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启

iOS系统版本

设备权限	功能及服务	权限授权方式
相册	截图/录制功能将图片/视频存储到相册	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启

Harmony系统版本

设备权限	功能及服务	权限授权方式
相册	截图/录制功能将图片/视频存储到相册	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用播放器 SDK产品和服务时遇到的问题，改进及优化播放器 SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用播放器 SDK服务时遇到的问题，改进及优化播放器 SDK的服务体验，提高播放器 SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用播放器 SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与播放器 SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息：

仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；
2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；
3. 如果我们或我们的关联公司涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；
2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；
6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；
8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在播放器 SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。
2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。
3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系，**并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。
4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过播放器 SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加密，注意确保您个人信息的安全。**
5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**
6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案，**由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。

2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。

3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

播放器 SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成播放器 SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及**《百度儿童个人信息保护声明》**，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过工单联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用播放器 SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；

2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；

3. 如果我们停止提供产品或者服务，或者保存期限已届满；

4. 如果最终用户撤回同意；

5. 法律、行政法规规定的其他情形。

当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

播放器 SDK愿一直陪伴开发者和最终用户，若因特殊原因导致播放器 SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过【八、如何联系我们】中的方式与我们取得联系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在播放器 SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用播放器 SDK产品和服务，若开发者继续集成并使用播放器 SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用播放器 SDK服务期间持续有效的授权同意。**若最终用户不同意该等变更，可以停止使用播放器 SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用播放器 SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；

2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；

3. 个人信息共享、转让或公开披露的主要对象发生变化；

4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；

5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；

6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

播放器 SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对播放器 SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过工单反馈开发者及最终用户对播放器 SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台(http://help.baidu.com/personalinformation) 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。 敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行播放器 SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等唯一设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问播放器 SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的**精确位置信息**，也包括通过IP地址或其他网络信息等获取的**粗略地理位置信息**。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。

匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度手机助手及其他百度系产品https://www.baidu.com/more/)等网站、程序、服务、工具及客户端。

关联公司是指播放器 SDK的经营者【北京百度网讯科技有限公司】及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对播放器 SDK的信任和使用！

北京百度网讯科技有限公司

【2023】年【12】月【15】日更新

【2023】年【12】月【15】日生效

播放器SDK开发者个人信息保护合规指引

亲爱的开发者：

感谢您在所开发的移动应用中集成并使用百度旗下软件开发工具包（SDK）。

百度非常重视用户个人信息保护，包括集成百度旗下播放器软件开发工具包（SDK）（以下简称“百度SDK”）的移动应用的最终用户（以下简称“最终用户”）个人信息保护，特制定《播放器 SDK个人信息保护合规开发者指引》，以供您在所开发的移动应用（以下简称“移动应用”）中集成并使用百度SDK时参照执行。

一、 个人信息保护合规基本要求

在所开发的移动应用中集成并使用百度SDK，您需要首先遵守以下个人信息保护合规基本要求：

1. 您应遵守收集、使用最终用户个人信息有关的所有可适用法律法规及规范性文件要求，包括但不限于《个人信息保护法》、《网络安全法》、《App违法违规收集使用个人信息行为认定方法》、《工业和信息化部关于开展APP侵害用户权益专项整治工作的通知》（工信部信管函［2019］337号）、《工业和信息化部关于开展纵深推进APP侵害用户权益专项整治行动的通知》（工信部信管函〔2020〕164号）、《工业和信息化部关于进一步提升移动互联网应用服务能力的通知》（工信部信管函〔2023〕26号）等，保护用户个人信息安全。
2. 您的APP需要制定一份独立的隐私政策。隐私政策的内容建议通俗易懂，对您的APP收集、使用个人信息的目的、方式、范围，清晰、完整、准确地向个人信息主体进行明示告知，并充分给予最终用户独立的选择权，确保在获得个人信息主体授权同意后方可进行个人信息的处理活动。《隐私政策》应由用户自主选择是否同意，不应以默认勾选同意的方式或是以欺骗诱导的方式取得用户授权。但请您注意，仅是改善服务质量、提升用户体验、定向推送信息、研发新产品还不足以能成为要求用户同意收集其个人信息的理由。
3. 您应将在移动应用中集成并使用百度SDK服务的情况，以及百度SDK对最终用户必要个人信息的收集、使用和保护规则（具体请见播放器 SDK隐私政策），在移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（具体请参考本指引第二部分 “如何告知最终用户”及第三部分“告知文案示例”），并获得最终用户对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。如果最终用户是未满14周岁的未成年人，请您务必确保获得最终用户的父母或其他监护人对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。
4. 您应向最终用户提供易于操作的访问、更正、删除其个人信息，撤销或更改其授权同意、注销其个人账号等用户权利实现机制。
5. 您应确保在移动应用首次运行时，应在最终用户阅读并同意移动应用隐私政策之后，方可初始化百度SDK进行最终用户信息采集。
6. 百度SDK会根据产品升级优化、提升安全性能、法律及监管要求等原因，不断升级迭代SDK版本，不同版本的SDK获取的字段信息可能会有差异。为了保证合法合规开展合作，并切实履行保护用户个人信息的责任与义务，请您务必确保您已将APP中的百度SDK升级至官方最新版本，以避免因使用旧版本SDK而出现违法违规的问题，导致您的APP遭受监管处罚的风险。百度SDK更新后会及时通过官网通知、站内信、公告、邮件、短信等有效方式对您进行告知，请您及时关注并尽快更新。

除了上述个人信息保护合规基本要求外，您还应遵守您所开发的移动应用所集成并使用的播放器 SDK隐私政策。

SDK基本业务功能：

百度智能云提供Web、Android、iOS及鸿蒙平台的播放器SDK，为开发者提供简单、便捷的开发接口，帮助开发者在各类终端设备上实现媒体播放功能。

Web端播放器 （cyberplayer）在标准版SDK之外，提供了高级版SDK，功能包含有H.265 编码格式软硬解播放、AV1编码格式软硬解播放、MPEG-TS播放。为用户带来更丰富的音视频体验。

Android、iOS、鸿蒙平台在标准版SDK之外，提供了高级版SDK，包含有全景声（WANOS）音频格式解码与音效处理、HDR多标准视频解码与渲染、超低延时直播、VR视频播放、智能防挡弹幕、投屏、绿幕抠图、端上超分等高级功能，这些高级功能也可以作为独立组件单独集成，为用户带来更丰富的音视频体验。

在Android、iOS平台，我们还提供了基于Unity框架的播放器SDK，帮助开发者在元宇宙、游戏、VR/AR等场景中快速实现高性能且丰富的媒体播放功能。不仅如此，借助全景声技术，我们在Unity框架上还提供了6DoF空间音频能力，为用户带来视觉、听觉上完全的3D体验。

以下是播放器 SDK获取您的APP的最终用户的个人信息以及权限，由于不同SDK版本采集的字段与是否可选可能存在一定差异，具体采集情况以实际接入的SDK版本为准：

SDK收集个人信息情况：

功能及服务	个人信息类型	个人信息字段（区分必选信息和可选信息）
区分不同设备品牌，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备品牌	必选
区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号	必选
区分不同设备系统版本，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	操作系统版本	必选
区分不同设备CPU型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	CPU信息	必选
区分不同设备的内存，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	内存使用情况	必选
区分设备的电量信息，确保产品服务在设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	电池电量信息	必选
区分不同设备的屏幕分辨率，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	屏幕分辨率	必选

安卓操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相册	截图、录制	将截图、录制的图片或视频存储到相册	使用录制、截图功能时

IOS操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相册	截图、录制	选择相册中的图片进行解码，将解码的图片存储到相机	使用录制、截图功能时

鸿蒙操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相册	截图、录制	选择相册中的图片进行解码，将解码的图片存储到相机	使用录制、截图功能时

二、 如何告知最终用户

为帮助您明确地告知最终用户百度SDK个人信息收集、使用和保护相关事宜，我们为您提供了以下告知方式，供您参考执行：

1. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并显示相应**百度SDK隐私政策链接**以告知最终用户，百度SDK收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。
2. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并参考相应百度SDK隐私政策内容，**以条款或表格等形式列明**收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。
3. 当最终用户在移动应用中首次打开/使用相应功能/服务时，以**弹窗、页面提示**方式显示相应**百度SDK隐私政策链接**，以告知最终用户相应功能/服务由百度SDK提供，百度SDK为提供相应功能/服务而收集、使用的最终用户个人信息类型、目的及用途，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

三、 告知文案示例

为帮助您明确地告知最终用户百度SDK个人信息保护规则相关事宜，我们为您提供了以下告知方文案示例，供您参考执行：

文案示例A

为向您提供视频播放功能/服务，我们集成了北京百度网讯科技有限公司的播放器 SDK。在为您提供视频播放功能/服务时，北京百度网讯科技有限公司播放器 SDK需要收集、使用您必要的个人信息，包括Wi-Fi状态、设备型号、IP地址、传感器，用于网络质量、设备信息和性能检测目的/用途。关于北京百度网讯科技有限公司播放器 SDK收集、使用的个人信息类型、目的及用途，以及播放器 SDK将如何保护所收集、使用的个人信息，请您仔细阅读《播放器 SDK隐私政策》了解。

文案示例B

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	个人信息类型	使用目的	官网链接/隐私政策链接
播放器SDK	北京百度网讯科技有限公司	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用播放等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/smcx2yd2l

文案示例C

为向您提供播放功能/服务，我们集成了北京百度网讯科技有限公司的播放器 SDK。在为您提供视频播放功能/服务时，北京百度网讯科技有限公司的播放器 SDK收集、使用您的Wi-Fi状态、设备型号、IP地址、传感器，用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用播放等功能。具体请您仔细阅读《播放器 SDK隐私政策》了解。

文案示例D

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	业务功能	个人信息类型	调用权限类型	具体目的/用途	隐私政策链接
播放器SDK	北京百度网讯科技有限公司	视频播放	设备品牌、设备型号、操作系统版本、CPU信息、内存使用情况、电池电量信息、屏幕分辨率信息	相册	用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用播放等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/smcx2yd2l

四、 使用SDK服务的合规注意事项

1. 您接入百度SDK前，应当仔细阅读SDK的协议约定、本合规规范、用户协议、隐私政策等内容，并依据相关内容对您APP的《隐私政策》及您APP收集、存储、使用、共享等处理个人信息的情况进行合规自查。

2. 您知悉并认可百度SDK具备收集个人信息的功能，并认可该等信息的收集均为双方合作之必要目的所需。
3. 您承诺已制定并按照相关要求公示您APP的《隐私政策》，并已清晰、明确、显著地说明有关通过SDK收集个人信息的必要性、收集数据的范围、方式以及用途。同时，您应确保在APP首次运行时以弹窗等合规方式显著提示用户阅读您APP的《隐私政策》并取得用户的合法授权同意，经过合法授权后再初始化百度SDK进行个人信息的收集与处理。
4. 您已认真阅读并理解百度SDK平台协议、合作规范、隐私政策、接入文档等约定和要求，并承诺在您使用百度SDK服务期间，针对百度SDK收集、使用、处理、共享、转让相关个人信息，您已取得了用户持续有效的授权和同意，并保证您不会违反国家相关法律法规、相关国家标准以及双方约定的目的。
5. 如果您的APP面向不满十四周岁的儿童及未成年人用户提供服务，您承诺遵守儿童个人信息保护及未成年人保护相关的法律法规，您承诺已采取相关措施并保证已获得其监护人的授权同意。
6. 如因您违反百度SDK的平台协议、合作规范、隐私政策、接入文档等约定，导致您的用户或第三方对百度主张任何形式的索赔或权利要求，或导致百度因此产生任何法律纠纷的，您将负责解决并承担全部责任，如因此给百度及其关联主体造成损失的，您应赔偿因此给百度及其关联主体造成的全部损失。
7. 您保证对于您从百度SDK获取的数据，无论是在合作期间或是合作停止后，均承担保密义务，不擅自提供、泄露、透露给任何第三方，并应采取一切合法措施以使上述数据免于散发、传播、披露、复制、滥用及被无关人员接触，避免导致相关数据被超出双方合作目的使用。

图片加载SDK

产品简介

百度智能云提供图片加载SDK，为开发者提供简单、便捷的开发接口，帮助开发者在移动设备上实现图片解码播放功能。

当前图片加载SDK支持HEIF静图格式解码。

> HEIF/HEIC图片压缩格式是一种使用HEVC编码技术存储图像数据的方式，在同等质量下相比JPEG可节省50%以上空间。百度智能云BOS支持HEIF/HEIC格式的转换、存储。

相比Android系统提供的原生接口，百度智能云HEIF图片加载SDK具备以下优点

- 无系统版本限制
- 速度快：SDK采用了先进的图像处理技术和优化算法，确保了图片解码的速度和效率，相比原生接口有数倍的性能提升
- 颜色准：SDK内置了先进的色彩管理模块，支持多种颜色空间，可以更准确地还原图片的色彩
- 功能多：SDK同时支持YUV400/420格式解码、Alpha通道、缩放、SO后下载等功能，具有更广泛的应用场景和更高的灵活性

> 关于iOS端的HEIF图片解码，从iOS11及以上版本系统API原生支持HEIF图片解码，所以无需单独进行接入。

功能列表

- 支持Android平台
- 支持armeabi-v7a、arm64-v8a架构
- 支持HEIF静图解码
- 支持缩放
- 支持的采样格式
  - YUV400
  - YUV420
- 支持的颜色空间
  - BT.601 full range / video range
  - BT.709 full range / video range
- 支持Alpha通道
- 支持集成到Glide、Freso
- 支持SO后下载

平台类别	Demo体验	下载地址	版本记录
Android		SDK + Demo	版本更新记录

个人信息处理规则

图片加载SDK隐私政策

合规使用说明

图片加载SDK开发者个人信息保护合规指引

快速开始

下面介绍Android端HEIF图片解码SDK的基本用法，以解码本地图片为例。

```
 // 读取Asset中的图片
private static InputStream loadInputStreamFromAssetFile(Context context, String fileName){
    AssetManager am = context.getAssets();
    try {
        return am.open(fileName);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}

private static ByteArrayOutputStream readDataFromInpuStream(InputStream inputStream) {
    try {
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = inputStream.read(buffer)) != -1) {
            output.write(buffer, 0, bytesRead);
        }
        return output;
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}

// 使用百度智能云HEIF图片加载SDK进行解码
private void decodeImageWithBaidu(byte[] heifData) {
    ImageView image = mRootView.findViewById(R.id.image_bdcloud);
    try {
        // 获取HEIF图片元数据
        HeifInfo heifInfo = new HeifInfo();
        HeifDecoder.getInfo(heifData.length, heifData, heifInfo);
        // 当前仅支持单帧静图解码
        HeifSize heifSize = heifInfo.getFrameList().get(0);
        // 获取HEIF图片宽高
        int width = heifSize.getWidth();
        int height = heifSize.getHeight();
        // 创建Android Bitmap
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        // 将HEIF图片解码为Bitmap
        HeifDecoder.toRgba(heifData.length, heifData, bitmap);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return;
}

// 完整业务流程展示
private void showLocalImage() {
    InputStream inputStream = loadInputStreamFromAssetFile(mContext, DEFAULT_LOCAL_IMG_URL);
    if (inputStream == null) {
        return;
    }

    ByteArrayOutputStream baos = readDataFromInpuStream(inputStream);
    if (baos == null) {
        return;
    }

    byte[] heifData = baos.toByteArray();
    decodeImageWithBaidu(heifData);

    try {
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        baos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return;
}
```

在Demo工程的LocalImageFragment.java中对上述流程做了详细的展示，可以参考。

快速进阶

下面介绍Android端HEIF图片加载SDK配合Glide和Freso库的使用方法，以及SO后下载功能的使用方法。

配合Glide实现HEIF图片解码

1. 在gradle中引入Glide库

```
 implementation 'com.baidubce.mediasdk:libheif:1.1.0'
implementation 'com.github.bumptech.glide:glide:4.15.1'
annotationProcessor 'com.github.bumptech.glide:compiler:4.15.1'
```

2. 基于百度智能云HEIF图片加载SDK编写Glide解码器插件

```
 // 以Bytebuffer为输入
public class HeifByteBufferBitmapDecoder implements ResourceDecoder<ByteBuffer, Bitmap> {
    private static final String TAG = "HeifByteBufferDecoder";
    private BitmapPool bitmapPool;

    public HeifByteBufferBitmapDecoder(BitmapPool pool) {
        bitmapPool = Preconditions.checkNotNull(pool);
    }

    @Override
    // 判断输入图片是否为HEIF格式
    public boolean handles(@NonNull ByteBuffer source, @NonNull Options options) throws IOException {
        byte[] header = new byte[13];
        ByteStreams.readHeaderFromStream(13, ByteBufferUtil.toStream(source), header);
        boolean isHeic = HeifDecoder.isHeic(header.length, header);
        Log.i(TAG, "isHeic " + isHeic);
        return isHeic;
    }

    @Override
    // 使用百度智能云HEIF图片加载SDK进行解码
    public Resource<Bitmap> decode(@NonNull ByteBuffer source, int width, int height,
                                   @NonNull Options options) throws IOException {
        byte[] bytes = ByteBufferUtil.toBytes(source);
        HeifInfo heifInfo = new HeifInfo();
        HeifDecoder.getInfo(bytes.length, bytes, heifInfo);
        HeifSize heifSize = heifInfo.getFrameList().get(0);
        int sourceWidth = heifSize.getWidth();
        int sourceHeight = heifSize.getHeight();
        // 缩放处理
        int requestedWidth = width;
        int requestedHeight = height;
        DownsampleStrategy downsampleStrategy =
                (DownsampleStrategy) options.get(DownsampleStrategy.OPTION);
        DownsampleStrategy.SampleSizeRounding rounding
                = downsampleStrategy.getSampleSizeRounding(sourceWidth, sourceHeight, requestedWidth, requestedHeight);
        float exactScaleFactor = downsampleStrategy.getScaleFactor(sourceWidth, sourceHeight,
                requestedWidth, requestedHeight);
        int outWidth = round((double) (exactScaleFactor * (float) sourceWidth));
        int outHeight = round((double) (exactScaleFactor * (float) sourceHeight));
        int widthScaleFactor = sourceWidth / outWidth;
        int heightScaleFactor = sourceHeight / outHeight;
        int scaleFactor = rounding == DownsampleStrategy.SampleSizeRounding.MEMORY ?
                Math.max(widthScaleFactor, heightScaleFactor) : Math.min(widthScaleFactor, heightScaleFactor);
        int powerOfTwoSampleSize = Math.max(1, Integer.highestOneBit(scaleFactor));
        if (rounding == DownsampleStrategy.SampleSizeRounding.MEMORY
                && (float) powerOfTwoSampleSize < 1.0F / exactScaleFactor) {
            powerOfTwoSampleSize <<= 1;
        }
        Log.i(TAG, "heifSize: " + heifSize.getWidth() + "," + heifSize.getHeight() +
                ",sample size: " + powerOfTwoSampleSize);
        // 创建Bitmap
        Bitmap bitmap = Bitmap.createBitmap(sourceWidth / powerOfTwoSampleSize, sourceHeight / powerOfTwoSampleSize,
                Bitmap.Config.ARGB_8888);
        // 将HEIF图片解码为Bitmap
        HeifDecoder.toRgba(bytes.length, bytes, bitmap);
        return BitmapResource.obtain(bitmap, bitmapPool);
    }

    private static int round(double value) {
        return (int) (value + 0.5);
    }
}

// 以InputStream为输入，内部直接复用上面的HeifByteBufferBitmapDecoder
public class HeifStreamBitmapDecoder implements ResourceDecoder<InputStream, Bitmap> {
    private static final String TAG = "HeifStreamBitmapDecoder";

    private final List<ImageHeaderParser> parsers;
    private final HeifByteBufferBitmapDecoder heifByteBufferBitmapDecoder;
    private final ArrayPool arrayPool;

    public HeifStreamBitmapDecoder(
            List<ImageHeaderParser> parsers,
            HeifByteBufferBitmapDecoder heifByteBufferBitmapDecoder,
            ArrayPool arrayPool) {
        this.parsers = parsers;
        this.heifByteBufferBitmapDecoder = Preconditions.checkNotNull(heifByteBufferBitmapDecoder);
        this.arrayPool = Preconditions.checkNotNull(arrayPool);
    }

    @Override
    @Nullable
    public Resource<Bitmap> decode(InputStream source, int width, int height, Options options)
            throws IOException {
        return heifByteBufferBitmapDecoder.decode(ByteBufferUtil.fromStream(source), width, height, options);
    }

    @Override
    public boolean handles(InputStream source, Options options) throws IOException {
        byte[] header = new byte[13];
        ByteStreams.readHeaderFromStream(13, source, header);
        boolean isheif =  HeifDecoder.isHeic(header.length, header);
        Log.i(TAG, "isHeic " + isheif);
        return isheif;
    }
}
```

3. 注册解码器插件

```
@GlideModule
public class HeifGlideModule extends AppGlideModule {
    @Override
    public void registerComponents(@NonNull Context context, @NonNull Glide glide, @NonNull Registry registry) {
        // 添加刚才编写的两个解码器插件
        HeifByteBufferBitmapDecoder heifByteBufferBitmapDecoder
                = new HeifByteBufferBitmapDecoder(glide.getBitmapPool());
        registry.prepend(ByteBuffer.class, Bitmap.class, heifByteBufferBitmapDecoder);

        HeifStreamBitmapDecoder streamBitmapDecoder =
                new HeifStreamBitmapDecoder(
                        registry.getImageHeaderParsers(), heifByteBufferBitmapDecoder, glide.getArrayPool());
        registry.prepend(InputStream.class, Bitmap.class, streamBitmapDecoder);
    }

    // Disable manifest parsing to avoid adding similar modules twice.
    @Override
    public boolean isManifestParsingEnabled() {
        return false;
    }
}
```

4. 使用

```
ImageView image = mRootView.findViewById(R.id.image_glide);
Glide.with(this)
        .asBitmap()
        .load(url)
        .override(100, 100) // 缩放
        .into(image)
```

配合Freso实现HEIF图片解码

1. 在gradle中引入Freso库

```
implementation 'com.baidubce.mediasdk:libheif:1.1.0'
implementation 'com.facebook.fresco:fresco:2.6.0'
```

2. 基于百度智能云HEIF加载SDK编写Freso解码器插件

```
public class HeifFrescoDecoder implements ImageDecoder {
  private static final String TAG = "HeifFrescoDecoder";

  @Override
  public CloseableImage decode(
        EncodedImage encodedImage,
        int length,
        QualityInfo qualityInfo,
        ImageDecodeOptions options) {
    // Decode the given encodedImage and return a
    // corresponding (decoded) CloseableImage.
    if (encodedImage == null) {
      return null;
    } else {
      InputStream inputStream = encodedImage.getInputStream();
      if (inputStream == null) {
        return null;
      }
      Bitmap bitmap;
      ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
      try {
        byte[] buffer = new byte[8192];
        int bytesRead;

        while ((bytesRead = inputStream.read(buffer)) != -1) {
          outputStream.write(buffer, 0, bytesRead);
        }

        byte[] heifData = outputStream.toByteArray();
        // 获取HEIF元数据
        HeifInfo heifInfo = new HeifInfo();
        HeifDecoder.getInfo(heifData.length, heifData, heifInfo);
        HeifSize heifSize = heifInfo.getFrameList().get(0);
        int sampleSize = encodedImage.getSampleSize();
        int width = heifSize.getWidth();
        int height = heifSize.getHeight();
        Log.i(TAG, "heifSize: " + heifSize.getWidth() + "," + heifSize.getHeight() + ", "
            + "out sample size: " + sampleSize);
        // 创建Android Bitmap，考虑缩放
        bitmap = Bitmap.createBitmap(width / sampleSize, height / sampleSize,
            Bitmap.Config.ARGB_8888);
        // 使用百度智能云HEIF图片加载SDK将HEIF图片解码为Bitmap
        HeifDecoder.toRgba(heifData.length, heifData, bitmap);

        CloseableReference<Bitmap> closeableReference =
            CloseableReference.of(Preconditions.checkNotNull(bitmap),
                SimpleBitmapReleaser.getInstance());

        return new CloseableStaticBitmap(
            closeableReference,
            ImmutableQualityInfo.FULL_QUALITY,
            encodedImage.getRotationAngle(),
            encodedImage.getExifOrientation());
      } catch (Exception e) {
        e.printStackTrace();
        return null;
      } finally {
        Closeables.closeQuietly(inputStream);
      }
    }
  }
}
```

3. 注册解码器插件

```
ImagePipelineConfig config = ImagePipelineConfig.newBuilder(this)
        .setDownsampleEnabled(true)
        .setImageDecoderConfig(ImageDecoderConfig.newBuilder()
            .addDecodingCapability(DefaultImageFormats.HEIF,
                new DefaultImageFormatChecker(),
                new HeifFrescoDecoder()) // 添加前面编写的解码器插件
            .build())
        .build();
Fresco.initialize(this, config);
```

4. 使用

```
SimpleDraweeView draweeView = (SimpleDraweeView) mRootView.findViewById(R.id.image_freso);
DraweeController draweeController = Fresco.newDraweeControllerBuilder()
        .setImageRequest(
            ImageRequestBuilder.newBuilderWithSource(Uri.parse(url))
                .setCacheChoice(ImageRequest.CacheChoice.DEFAULT)
                .setResizeOptions(ResizeOptions.forDimensions(100 , 100)) // 缩放
                .build())
        .build();
draweeView.setController(draweeController);
```

在Demo工程的OnlineImageFragment.java中对上面两种用法都做了详细的展示，可以参考。

SO后下载

通过SO后下载，可以减少包体积增量。

1. 配置gradle集成无SO的HEIF图片加载SDK

```
dependencies {
    implementation 'com.baidubce.mediasdk:libheif-lite:1.1.0'
}
```

2. 下载SDK压缩包，解压找到arm64-v8a_heif.zip和armeabi-v7a_heif.zip，这两个压缩包中保存有对应架构的SO文件。

3. 将SO文件上传到您的服务器（或使用百度智能云对象存储BOS），并记录下载地址，例如 https://xxx.com/arm64-v8a_heif.zip。

4. 调用HeifSoLoadManager从指定位置加载对应架构的SO，加载成功后再进行后续的解码

```
HeifSoLoadManager.getInstance(this).loadLibraries("https://xxx.com/arm64-v8a_heif.zip", "arm64-v8a", mLoadListener);

private HeifSoLoadManager.LoadListener mLoadListener = new HeifSoLoadManager.LoadListener() {

        @Override
        public void onLoadError(int errCode, String errMsg) {
            Log.d(TAG, "external load library error " + errCode + errMsg);
        }

        @Override
        public void onLibsDownloadCompleted() {
            Log.d(TAG, "libs download completed.");
        }

        @Override
        public void onLoadSuccess() {
            Log.d(TAG, "external load library success.");
        }

        @Override
        public void onLoadProgress(float progress) {
            Log.d(TAG, "external load library so progress " + progress);
        }
    };
```

SDK集成

下面介绍Android端HEIF图片加载SDK的集成方法

配置工程

您可以选择使用maven配置，也可以通过手动集成将aar包加入到工程中。

maven配置

在根级gradle中添加mavenCentral仓库，如下所示

```
buildscript {
    repositories {
        mavenCentral()
    }
}
allprojects {
    repositories {
        mavenCentral()
    }
}
```

在模块gradle中添加具体SDK的依赖，如下所示

```
dependencies {
    implementation 'com.baidubce.mediasdk:libheif:1.1.0'
}
```

手动配置aar包

将HeifDecoder.aar复制到您工程的app/libs目录下，并在gradle文件的dependencies模块注明aar包路径，如下所示：

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

混淆配置

添加如下的混淆配置内容

```
-keep class com.baidu.cloud.imgdec.libheif.** {*;}
```

接口速查

HeifDecoder类

接口名	描述
public static native boolean toRgba(long len, byte[] buf, Bitmap bitmap)	将输入HEIF图片数据解码为Bitmap len：对应输入HEIF图片数据的byte数组长度 buf：对应输入HEIF图片数据的byte数组 bitmap：解码后的Bitmap 解码成功返回true，否则返回false
public static native boolean isHeic(long len, byte[] buf)	判断输入图片数据是否为HEIF格式 len：输入图片数据byte数组长度 buf：输入图片数据byte数组 如果输入图片数据是HEIF格式则返回true，否则返回false
public static native boolean getInfo(long len, byte[] buf, HeifInfo info)	获取HEIF图片信息 len：输入图片数据byte数组长度 buf：输入图片数据byte数组 info：解析得到的图片信息 解析成功返回true，否则返回false

HeifSoLoadManager类

接口名	描述
public static synchronized HeifSoLoadManager getInstance(Context context)	获取单例，用于SO后下载
public synchronized void loadLibraries(String soLaterLoadUrl, String cpuType, LoadListener loadListener)	从指定URL下载SO并加载，架构由cpuType参数指定（如"arm64-v8a"或"armeabi-v7a"），加载进度通过loadListener回调

版本更新记录

发布时间	版本号	功能描述
2023-7	1.1.0	初版本发布 1.支持HEIF静图解码 2.支持Alpha通道 3.支持SO后下载

SDK&Demo下载

合规指南

- 开发者： 北京百度网讯科技有限公司
- SDK名称： 百度智能云图片加载SDK
- 版本号： 见版本更新记录
- 主要功能： heif图片解码播放
- 个人信息处理规则： 见 图片加载SDK隐私政策
- 合规使用说明：见 图片加载SDK开发者个人信息保护合规指引

下载SDK&Demo

本文档提供最新版本的 SDK。如果您需要老版本，请联系技术支持获取。

平台	SDK文件名	下载
Android 包名：com.baidu.cloud.imgdec.libheif md5：f7cddb2256963644eb616d9e77c10c32	Baidu-Cloud-HeifDecoder-Android-1.1.0.zip	点击下载

相关协议

图片加载SDK隐私政策

欢迎使用图片加载软件开发工具包（SDK）（简称"图片加载 SDK"）服务！

图片加载 SDK 是一款为移动应用开发者（以下简称"开发者"）提供简单、便捷的开发接口，帮助开发者在移动设备上实现图片解码播放功能（SDK）。开发者在其移动应用内集成图片加载 SDK后，可通过图片加载SDK平台向其移动应用（以下简称"开发者应用"）的最终用户（以下简称"最终用户"）提供本隐私政策所说明的功能及服务。开发者在其移动应用集成并使用图片加载 SDK服务时，委托图片加载 SDK处理开发者应用相关数据信息，其中可能包括开发者应用最终用户（以下简称"最终用户"）的个人信息。此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策"附录1：名词解释"来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用图片加载 SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用图片加载 SDK服务的情况，以及图片加载 SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于图片加载 SDK收集、使用最终用户相关个人信息的完整、合法、在使用图片加载 SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于图片加载 SDK收集、使用最终用户相关个人信息的完整、合法、在使用图片加载 SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

（4）关于上述承诺的具体落地执行可参考《图片加载 SDK开发者个人信息保护合规指引》。

2、我们希望集成并使用图片加载 SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的图片加载 SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，图片加载 SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用图片加载 SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订
8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用图片加载 SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	网络质量监测，根据网络状态调整网络策略	Wi-Fi状态（必选）	采用加密传输的安全处理方式	iOS及Android
2	区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号（必选）	采用加密传输的安全处理方式	iOS及Android
3	根据IP地址，检测SDK通信质量	IP地址（可选）	采用加密传输的安全处理方式	iOS及Android

2. 设备权限调用

为了保证最终用户能正常使用图片加载 SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能，各项功能及功能对设备权限的调用情况如下：

Android系统版本

设备权限	功能及服务	权限授权方式
相册	选择相册中的图片进行解码，将解码的图片存储到相机	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启

iOS系统版本

设备权限	功能及服务	权限授权方式
相册	选择相册中的图片进行解码，将解码的图片存储到相机	授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用图片加载 SDK产品和服务时遇到的问题，改进及优化图片加载 SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用图片加载 SDK服务时遇到的问题，改进及优化图片加载 SDK的服务体验，提高图片加载 SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用图片加载 SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与图片加载

SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息：

仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；
2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；
3. 如果我们或我们的关联公司涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；
2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；
6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；
8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在图片加载 SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。
2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。
3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系，**并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。
4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过图片加载 SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加**

密，注意确保您个人信息的安全。

5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**

6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案，**由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。

2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。

3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

图片加载 SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成图片加载 SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及**《百度儿童个人信息保护声明》**，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过工单联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用图片加载 SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；

2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；

3. 如果我们停止提供产品或者服务，或者保存期限已届满；

4. 如果最终用户撤回同意；

5. 法律、行政法规规定的其他情形。

当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过【八、如何联系我们】中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

图片加载 SDK愿一直陪伴开发者和最终用户，若因特殊原因导致图片加载 SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过【八、如何联系我们】中的方式与我们取得联系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在图片加载 SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用图片加载 SDK产品和服务，若开发者继续集成并使用图片加载 SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用图片加载 SDK服务期间持续有效的授权同意。**若最终用户不同意该等变更，可以停止使用图片加载 SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用图片加载 SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；

2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；

3. 个人信息共享、转让或公开披露的主要对象发生变化；

4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；

5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；

6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

图片加载 SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对图片加载 SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过工单反馈开发者及最终用户对图片加载 SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台(http://help.baidu.com/personalinformation) 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。 敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行图片加载 SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等唯一设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问图片加载 SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的**精确位置信息**，也包括通过IP地址或其他网络信息等获取的**粗略地理位置信息**。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。

匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度手机助手及其他百度系产品https://www.baidu.com/more/)等网站、程序、服务、工具及客户端。

关联公司是指图片加载 SDK的经营者【北京百度网讯科技有限公司】及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对图片加载 SDK的信任和使用！

北京百度网讯科技有限公司

【2023】年【12】月【15】日更新

【2023】年【12】月【15】日生效

```
### 图片加载 SDK开发者个人信息保护合规指引


亲爱的开发者：

感谢您在所开发的移动应用中集成并使用百度旗下软件开发工具包（SDK）。

百度非常重视用户个人信息保护，包括集成百度旗下图片加载软件开发工具包（SDK）（以下简称“百度SDK”）的移动应用的最终用户（以下简称“最终用户”）个人信息保护，特制定《图片加载 SDK个人信息保护合规开发者指引》，以供您在所开发的移动应用（以下简称“移动应用”）中集成并使用百度SDK时参照执行。

##### 一、 个人信息保护合规基本要求

在所开发的移动应用中集成并使用百度SDK，您需要首先遵守以下个人信息保护合规基本要求：

1. 您应遵守收集、使用最终用户个人信息有关的所有可适用法律法规及规范性文件要求，包括但不限于《个人信息保护法》、《网络安全法》、《App违法违规收集使用个人信息行为认定方法》、《工业和信息化部关于开展APP侵害用户权益专项整治工作的通知》（工信部信管函［2019］337号）、《工业和信息化部关于开展纵深推进APP侵害用户权益专项整治行动的通知》（工信部信管函〔2020〕164号）、《工业和信息化部关于进一步提升移动互联网应用服务能力的通知》（工信部信管函〔2023〕26号）等，保护用户个人信息安全。

2. 您的APP需要制定一份独立的隐私政策。隐私政策的内容建议通俗易懂，对您的APP收集、使用个人信息的目的、方式、范围，清晰、完整、准确地向个人信息主体进行明示告知，并充分给予最终用户独立的选择权，确保在获得个人信息主体授权同意后方可进行个人信息的处理活动。《隐私政策》应由用户自主选择是否同意，不应以默认勾选同意的方式或是以欺骗诱导的方式取得用户授权。但请您注意，仅是改善服务质量、提升用户体验、定向推送信息、研发新产品还不足以能成为要求用户同意收集其个人信息的理由。

3. 您应将在移动应用中集成并使用百度SDK服务的情况，以及百度SDK对最终用户必要个人信息的收集、使用和保护规则（具体请见[图片加载 SDK隐私政策]
```

(https://cloud.baidu.com/doc/VideoCreatingSDK/s/Dmcx0rfi8)），在移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（具体请参考本指引第二部分 “如何告知最终用户”及第三部分“告知文案示例”），并获得最终用户对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。如果最终用户是未满14周岁的未成年人，请您务必确保获得最终用户的父母或其他监护人对于百度SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度SDK服务期间持续有效的授权同意。

4. 您应向最终用户提供易于操作的访问、更正、删除其个人信息，撤销或更改其授权同意、注销其个人账号等用户权利实现机制。

5. 您应确保在移动应用首次运行时，应在最终用户阅读并同意移动应用隐私政策之后，方可初始化百度SDK进行最终用户信息采集。

6. 百度SDK会根据产品升级优化、提升安全性能、法律及监管要求等原因，不断升级迭代SDK版本，不同版本的SDK获取的字段信息可能会有差异。为了保证合法合规开展合作，并切实履行保护用户个人信息的责任与义务，请您务必确保您已将APP中的百度SDK升级至官方最新版本，以避免因使用旧版本SDK而出现违法违规的问题，导致您的APP遭受监管处罚的风险。百度SDK更新后会及时通过官网通知、站内信、公告、邮件、短信等有效方式对您进行告知，请您及时关注并尽快更新。

除了上述个人信息保护合规基本要求外，您还应遵守您所开发的移动应用所集成并使用的图片加载 SDK隐私政策。

SDK基本业务功能：

图片加载 SDK基本业务功能为提供HEIF静图格式解码、播放，支持YUV400/YUV420采样格式，支持Alpha通道。

以下是图片加载 SDK获取您的APP的最终用户的个人信息以及权限，由于不同SDK版本采集的字段与是否可选可能存在一定差异，具体采集情况以实际接入的SDK版本为准：

SDK收集个人信息情况：

功能及服务	个人信息类型	个人信息字段（区分必选信息和可选信息）
网络质量监测，根据网络状态调整网络策略	Wi-Fi状态	必选
区分不同设备型号，确保产品服务在不同设备上的兼容性，对兼容/崩溃问题进行适配和故障排查	设备型号	必选
根据IP地址，检测SDK通信质量	IP地址	可选

安卓操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相册	选择图片、存储图片	选择相册中的图片进行解码，将解码的图片存储到相机	选择解码图片时、解码完成保存到相册

IOS操作系统权限申请情况：

权限	功能	用途和目的	申请时机
相册	选择图片、存储图片	选择相册中的图片进行解码，将解码的图片存储到相机	选择解码图片时、解码完成保存到相册

SDK初始化设置：

请务必确保您已将图片加载SDK升级到最新版。上述版本调用初始化函数不会采集用户个人信息，也不会向百度联盟后台上报数据。请确保用户同意《隐私政策》后再进行toRgba解码操作，方可采集用户个人信息并上报数据。

调用初始化函数文档： https://cloud.baidu.com/doc/VideoCreatingSDK/s/Slkgora6o

二、 如何告知最终用户

为帮助您明确地告知最终用户百度SDK个人信息收集、使用和保护相关事宜，我们为您提供了以下告知方式，供您参考执行：

1. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并显示相应**百度SDK隐私政策链接**以告知最终用户，百度SDK收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

2. 在移动应用隐私政策中**“个人信息共享”**条款部分或**“所集成的第三方SDK”**条款部分告知最终用户相应功能/服务由百度SDK提供，并参考相应百度SDK隐私政策内容，**以条款或表格等形式列明**收集、使用的最终用户个人信息类型、目的及用途。移动应用隐私政策在用户首次打开移动应用或者在移动应用的注册/登记界面通过**协议在线展示**的方式向用户展示，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

3. 当最终用户在移动应用中首次打开/使用相应功能/服务时，以**弹窗、页面提示**方式显示相应**百度SDK隐私政策链接**，以告知最终用户相应功能/服务由百度SDK提供，百度SDK为提供相应功能/服务而收集、使用的最终用户个人信息类型、目的及用途，并获得最终用户**明示同意**（例如：点击“同意”，或勾选“√”）。

三、 告知文案示例

为帮助您明确地告知最终用户百度SDK个人信息保护规则相关事宜，我们为您提供了以下告知方文案示例，供您参考执行：

文案示例A

为向您提供HEIF图片解码/服务，我们集成了北京百度网讯科技有限公司的图片加载 SDK。在为您提供图片解码播放功能/服务时，北京百度网讯科技有限公司图片加载 SDK需要收集、使用您必要的个人信息，包括Wi-Fi状态、设备型号、IP地址，用于网络质量、设备信息和性能检测目的/用途。关于北京百度网讯科技有限公司图片加载 SDK收集、使用的个人信息类型、目的及用途，以及图片加载 SDK将如何保护所收集、使用的个人信息，请您仔细阅读[《图片加载 SDK隐私政策》](https://cloud.baidu.com/doc/VideoCreatingSDK/s/Dmcx0rfi8)了解。

文案示例B

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	个人信息类型	使用目的	官网链接/隐私政策链接
图片加载 SDK	北京百度网讯科技有限公司	Wi-Fi状态、设备型号、IP地址	用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用图片解码等功能	https://cloud.baidu.com/doc/VideoCreatingSDK/s/Dmcx0rfi8

文案示例C

为向您提供图片解码播放功能/服务，我们集成了北京百度网讯科技有限公司的图片加载 SDK。在为您提供图片解码播放功能/服务时，北京百度网讯科技有限公司的图片加载 SDK收集、使用您的Wi-Fi状态、设备型号、IP地址，用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用图片解码等功能。具体请您仔细阅读[《图片加载 SDK隐私政策》](https://cloud.baidu.com/doc/VideoCreatingSDK/s/Dmcx0rfi8)了解。

文案示例D

为保障App相关功能的实现与应用安全稳定的运行，我们可能会接入由第三方提供的SDK实现相关目的，具体接入的相关第三方SDK列明如下：

第三方SDK名称	公司名称	业务功能	个人信息类型	调用权限类型	具体目的/用途	隐私政策链接

| 图片加载 SDK | 北京百度网讯科技有限公司 | 图片解码 | Wi-Fi状态、设备型号、IP地址 | 相册 | 用于网络质量、设备信息和性能检测目的/用途，保障用户正常使用图片解码等功能 | https://cloud.baidu.com/doc/VideoCreatingSDK/s/Dmcx0rfi8|

四、 使用SDK服务的合规注意事项

1. 您接入百度SDK前，应当仔细阅读SDK的协议约定、本合规规范、用户协议、隐私政策等内容，并依据相关内容对您APP的《隐私政策》及您APP收集、存储、使用、共享等处理个人信息的情况进行合规自查。

2. 您知悉并认可百度SDK具备收集个人信息的功能，并认可该等信息的收集均为双方合作之必要目的所需。

3. 您承诺已制定并按照相关要求公示您APP的《隐私政策》，并已清晰、明确、显著地说明有关通过SDK收集个人信息的必要性、收集数据的范围、方式以及用途。同时，您应确保在APP首次运行时以弹窗等合规方式显著提示用户阅读您APP的《隐私政策》并取得用户的合法授权同意，经过合法授权后再初始化百度SDK进行个人信息的收集与处理。

4. 您已认真阅读并理解百度SDK平台协议、合作规范、隐私政策、接入文档等约定和要求，并承诺在您使用百度SDK服务期间，针对百度SDK收集、使用、处理、共享、转让相关个人信息，您已取得了用户持续有效的授权和同意，并保证您不会违反国家相关法律法规、相关国家标准以及双方约定的目的。

5. 如果您的APP面向不满十四周岁的儿童及未成年人用户提供服务，您承诺遵守儿童个人信息保护及未成年人保护相关的法律法规，您承诺已采取相关措施并保证已获得其监护人的授权同意。

6. 如因您违反百度SDK的平台协议、合作规范、隐私政策、接入文档等约定，导致您的用户或第三方对百度主张任何形式的索赔或权利要求，或导致百度因此产生任何法律纠纷的，您将负责解决并承担全部责任，如因此给百度及其关联主体造成损失的，您应赔偿因此给百度及其关联主体造成的全部损失。

7. 您保证对于您从百度SDK获取的数据，无论是在合作期间或是合作停止后，均承担保密义务，不擅自提供、泄露、透露给任何第三方，并应采取一切合法措施以使上述数据免于散发、传播、披露、复制、滥用及被无关人员接触，避免导致相关数据被超出双方合作目的使用。