音视频终端 SDK(腾讯云视立方)

直播

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2022 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

版权所有：腾讯云计算（北京）有限责任公司

直播
直播推流
iOS

最近更新时间：2022-03-04 11:05:17

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

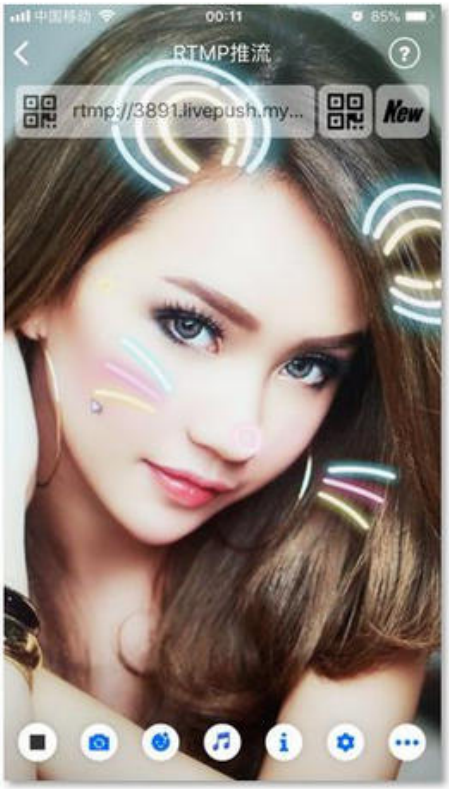
不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

功能概述

摄像头推流，是指采集手机摄像头的画面以及麦克风的声音，进行编码之后再推送到直播云平台上。腾讯云视立方 LiteAVSDK 通过 V2TXLivePusher 接口提供摄像头推流能力，如下是 LiteAVSDK 的简单版 Demo 中演示摄

版权所有：腾讯云计算（北京）有限责任公司

像头推流的相关操作界面：



特别说明

x86 模拟器调试：由于 SDK 大量使用 iOS 系统的音视频接口，这些接口在 Mac 上自带的 x86 仿真模拟器下往往不能工作。所以，如果条件允许，推荐您尽量使用真机调试。

示例代码

所属平台	GitHub 地址	关键类
iOS	Github	CameraPushViewController.m
Android	Github	CameraPushMainActivity.java

> 说明：
> 除上述示例外，针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。
>
> - iOS：MLVB-API-Example
> - Android：MLVB-API-Example

功能对接

1. 下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

2. 给 SDK 配置 License 授权

单击 License 申请 获取测试用的 License，您会获得两个字符串：一个字符串是 licenseURL，另一个字符串是解密 key。
在您的 App 调用 LiteAVSDK 的相关功能之前（建议在 - [AppDelegate application:didFinishLaunchingWithOptions:] 中）进行如下设置：

```
@import TXLiteAVSDK_Professional;
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
NSString * const licenceURL = @"<获取到的licenseUrl>";
NSString * const licenceKey = @"<获取到的key>";

//TXLiveBase 位于 "TXLiveBase.h" 头文件中
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
@end
```

3. 初始化 V2TXLivePusher 组件

创建一个 V2TXLivePusher 对象，需要指定对应的 V2TXLiveMode 。

```
V2TXLivePusher *pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTMP]; // 指定对应的直播协议为 RTMP
```

4. 开启摄像头预览

想要开启摄像头预览，首先需要调用 V2TXLivePusher 中的 setRenderView 接口，改接口需要设置一个用于显示视频画面的 view 对象，然后调用 startCamera 接口开启当前手机摄像头的预览。

```
//创建一个 view 对象，并将其嵌入到当前界面中
UIView *_localView = [[UIView alloc] initWithFrame:self.view.bounds];
[self.view insertSubview:_localView atIndex:0];
```

```
_localView.center = self.view.center;
//启动本地摄像头预览
[_pusher setRenderView:_localView];
[_pusher startCamera:YES];
```

⚠ 注意：

如果要给 view 增加动画效果，需要修改 view 的 transform 属性而不是 frame 属性。

```
[UIView animateWithDuration:0.5 animations:^{
_localView.transform = CGAffineTransformMakeScale(0.3, 0.3); //缩小1/3
}];
```

5. 启动和结束推流

如果已经通过 startCamera 接口启动了摄像头预览，就可以调用 V2TXLivePusher 中的 startPush 接口开始推流。推流地址可以使用 TRTC 地址，或者使用 RTMP 地址，前者使用 UDP 协议，推流质量更高，并支持连麦互动。

```
//启动推流， URL 可以使用 trtc:// 或者 rtmp:// 两种协议，前者支持连麦功能
NSString* url = @"trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx"; //支持连麦
NSString* url = @"rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx"; //不支持连麦，直接推流到直播 CDN
[_pusher startPush:url];
```

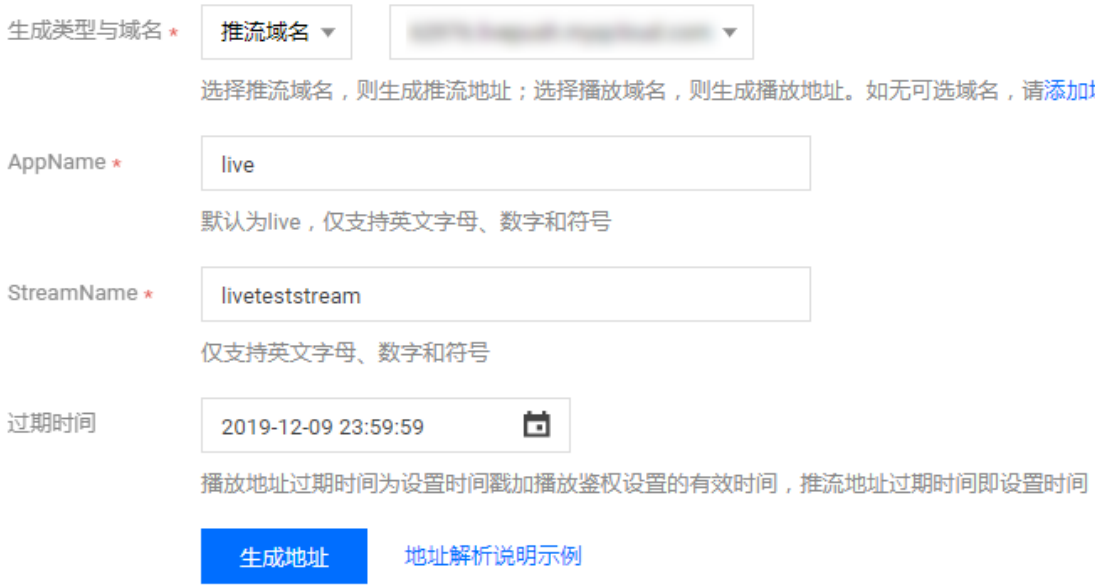
推流结束后，可以调用 V2TXLivePusher 中的 stopPush 接口结束推流。

```
//结束推流
[_pusher stopPush];
```

⚠ 注意：

如果已经启动了摄像头预览，请在结束推流时将其关闭。

- **获取可用的推流 URL**
  开通直播服务后，可以使用**直播控制台 > 辅助工具> 地址生成器** 生成推流地址，详细信息请参见 推拉流 URL。



- **返回 V2TXLIVE_ERROR_INVALID_LICENSE 的原因**
  如果 startPush 接口返回 V2TXLIVE_ERROR_INVALID_LICENSE，则代表您的 License 校验失败了，请检查 第2步：给 SDK 配置 License 授权 中的工作是否有问题。

6. 纯音频推流

如果您的直播场景是纯音频直播，不需要视频画面，那么您可以不执行 第4步 中的操作，或者在调用 startPush 之前不调用 startCamera 接口即可。

```
V2TXLivePusher *_pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTMP];
//启动推流， URL 可以使用 trtc:// 或者 rtmp:// 两种协议，前者支持连麦功能
NSString* url = @"trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx"; //支持连麦
NSString* url = @"rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx"; //不支持连麦，直接推流到直播 CDN
[_pusher startPush:url];
[_pusher startMicrophone];
```

> **说明：**
> 如果您启动纯音频推流，但是 RTMP、FLV 、HLS 格式的播放地址拉不到流，那是因为线路配置问题，请提工单 联系我们帮忙修改配置。

7. 设定画面清晰度

调用 V2TXLivePusher 中的 setVideoQuality 接口，可以设定观众端的画面清晰度。之所以说是观众端的画面清晰度，是因为主播看到的视频画面是未经编码压缩过的高清原画，不受设置的影响。而 setVideoQuality 设定的视频编码器的编码质量，观众端可以感受到画质的差异。详情请参见 设定画面质量。

8. 美颜美白和红润特效

调用 V2TXLivePusher 中的 getBeautyManager 接口可以获取 TXBeautyManager 实例进一步设置美颜效果。

美颜风格

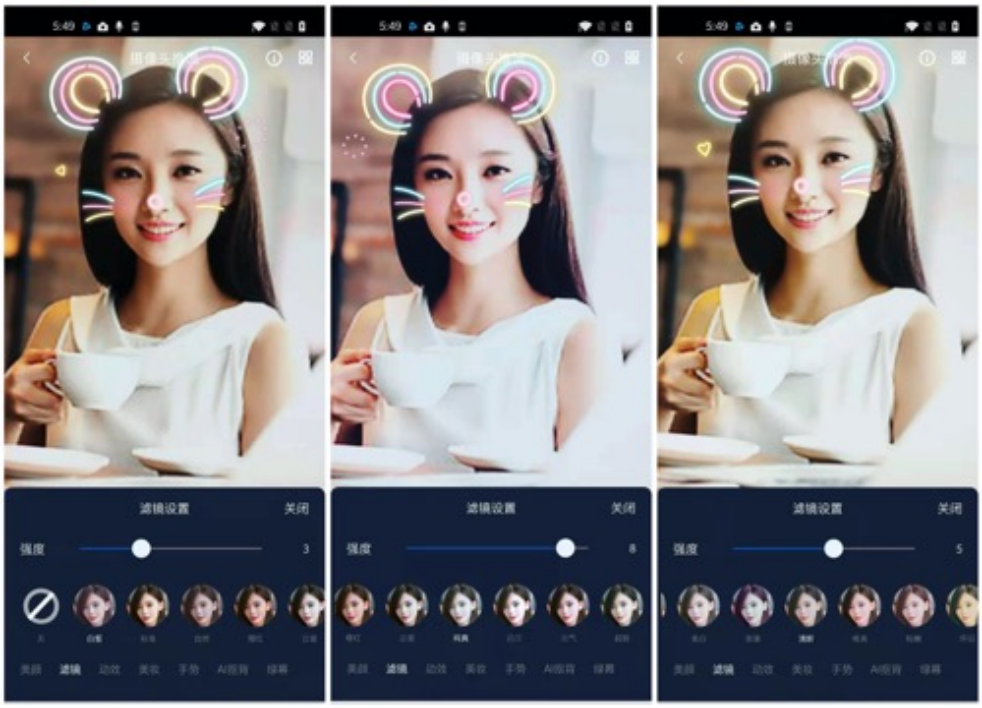
SDK 内置三种不同的磨皮算法，每种磨皮算法即对应一种美颜风格，您可以选择最适合您产品定位的方案。详情请参见 TXBeautyManager.h 文件：

美颜风格	效果说明
TXBeautyStyleSmooth	光滑，适用于美女秀场，效果比较明显
TXBeautyStyleNature	自然，磨皮算法更多地保留了面部细节，主观感受上会更加自然
TXBeautyStylePitu	由上海优图实验室提供的美颜算法，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高

美颜风格可以通过 TXBeautyManager 的 setBeautyStyle 接口设置：

美颜风格	设置方式	接口说明
美颜级别	通过 TXBeautyManager 的 setBeautyLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显
美白级别	通过 TXBeautyManager 的 setWhitenessLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显
红润级别	通过 TXBeautyManager 的 setRuddyLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显

9. 色彩滤镜效果



- 调用 V2TXLivePusher 中的 getBeautyManager 接口可以获取 TXBeautyManager 实例进一步设置美色彩滤镜效果。
- 调用 TXBeautyManager 的 setFilter 接口可以设置色彩滤镜效果。所谓色彩滤镜，是指一种将整个画面色调进行区域性调整的技术，例如将画面中的淡黄色区域淡化实现肤色亮白的效果，或者将整个画面的色彩调暖让视频的效果更加清新和温和。
- 调用 TXBeautyManager 的 setFilterStrength 接口可以设定滤镜的浓度，设置的浓度越高，滤镜效果也就越明显。

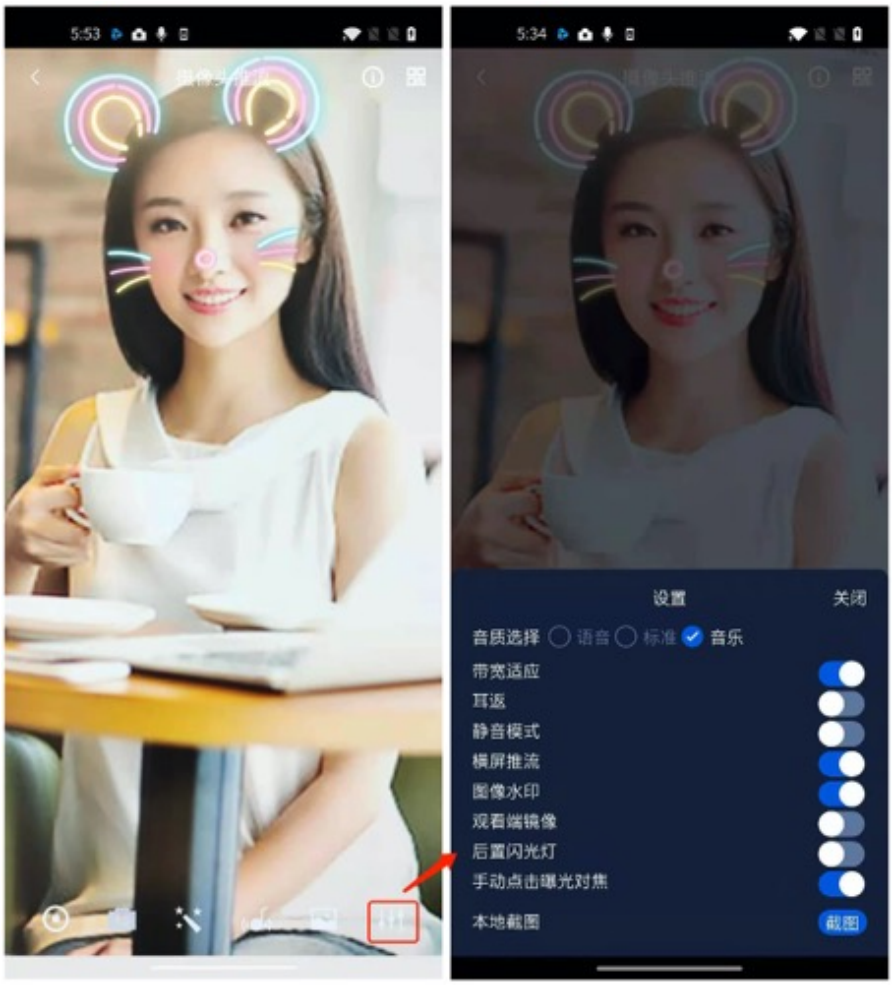
从手机 QQ 和 Now 直播的经验来看，单纯通过 TXBeautyManager 的 setBeautyStyle 调整美颜风格是不够的，只有将美颜风格和 setFilter 配合使用才能达到更加丰富的美颜效果。所以，我们的设计师团队提供了17种默认的色彩滤镜，并将其默认打包在了 Demo 中供您使用。

```
NSString * path = [[NSBundle mainBundle] pathForResource:@"FilterResource" ofType:@"bundle"
];
path = [path stringByAppendingPathComponent:lookupFileName];

UIImage *image = [UIImage imageWithContentsOfFile:path];
[[_pusher getBeautyManager] setFilter:image];
[[_pusher getBeautyManager] setFilterStrength:0.5f];
```

10. 设备管理

V2TXLivePusher 提供了一组 API 用户控制设备的行为，您可通过 getDeviceManager 获取 TXDeviceManager 实例进一步进行设备管理，详细用法请参见 TXDeviceManager API。



11. 观众端的镜像效果

通过调用 V2TXLivePusher 的 setRenderMirror 可以改变摄像头的镜像方式，继而影响观众端观看到的镜像效果。之所以说是观众端的镜像效果，是因为当主播在使用前置摄像头直播时，默认情况下自己看到的画面会被 SDK 反转。

12. 横屏推流

大多数情况下，主播习惯以“竖屏持握”手机进行直播拍摄，观众端看到的也是竖屏分辨率的画面（例如 540 × 960 这样的分辨率）；有时主播也会“横屏持握”手机，这时观众端期望能看到是横屏分辨率的画面（例如 960 × 540 这样的分辨率），如下图所示：

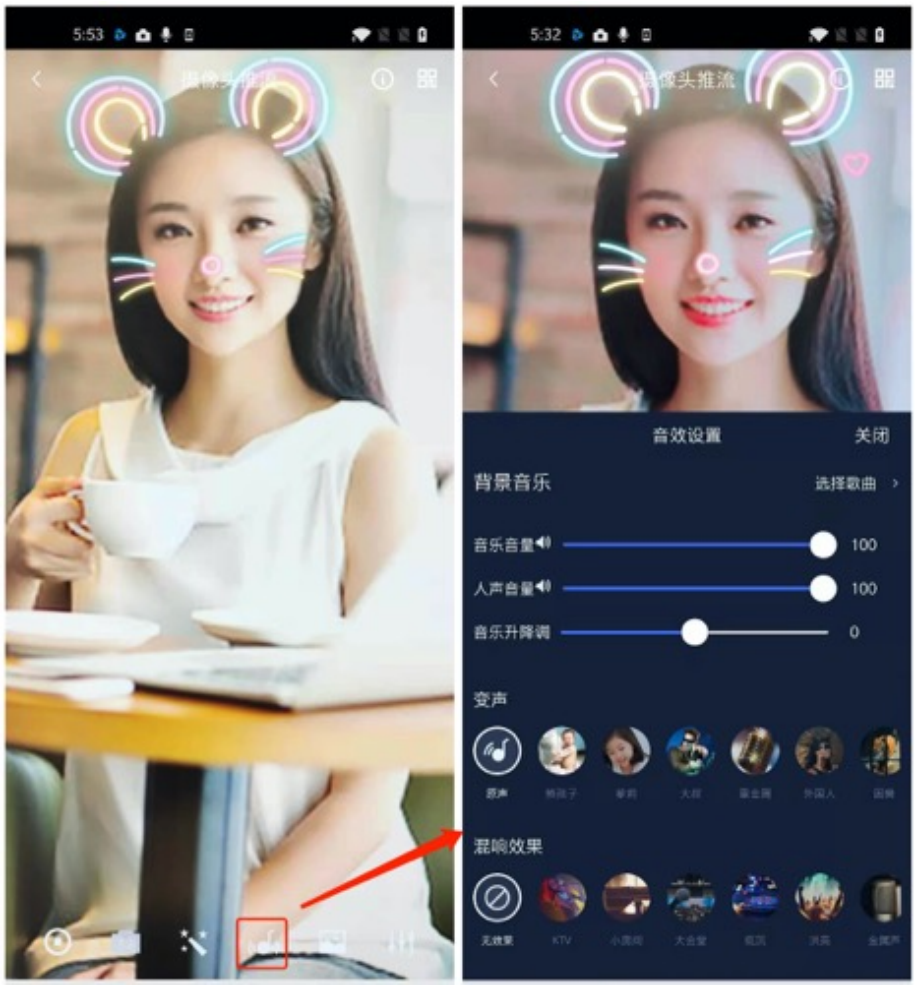
V2TXLivePusher 默认推出的是竖屏分辨率的视频画面，如果希望推出横屏分辨率的画面，可以修改 setVideoQuality 接口的参数来设定观众端的画面横竖屏模式。

```
[_pusher setVideoQuality:videoQuality
resolutionMode:isLandscape ? V2TXLiveVideoResolutionModeLandscape : V2TXLiveVideoResoluti
onModePortrait];
```

13. 音效设置

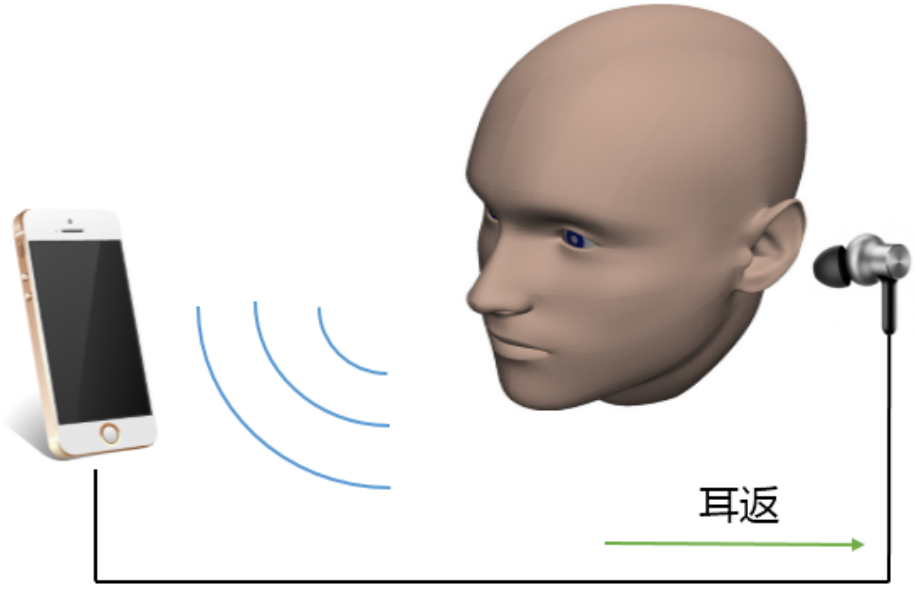
调用 V2TXLivePusher 中的 getAudioEffectManager 获取 TXAudioEffectManager 实例可以实现背景混音、耳返、混响等音效功能。背景混音是指主播在直播时可以选取一首歌曲伴唱，歌曲会在主播的手机端播放出来，

同时也会被混合到音视频流中被观众端听到，所以被称为“混音”。



- 调用 TXAudioEffectManager 中的 enableVoiceEarMonitor 选项可以开启耳返功能，“耳返”指的是当主播带上耳机来唱歌时，耳机中要能实时反馈主播的声音。
- 调用 TXAudioEffectManager 中的 setVoiceReverbType 接口可以设置混响效果，例如 KTV、会堂、磁性、金属等，这些效果也会作用到观众端。
- 调用 TXAudioEffectManager 中的 setVoiceChangerType 接口可以设置变调效果，例如“萝莉音”，“大叔音”等，用来增加直播和观众互动的趣味性，这些效果也会作用到观众端。

版权所有：腾讯云计算（北京）有限责任公司



耳返功能示意图

> **说明：**
> 详细用法请参见 TXAudioEffectManager API。

14. 设置 Logo 水印

设置 V2TXLivePusher 中的 setWatermark 可以让 SDK 在推出的视频流中增加一个水印，水印位置位是由传入参数（x, y, scale）所决定。

- SDK 所要求的水印图片格式为 PNG 而不是 JPG，因为 PNG 这种图片格式有透明度信息，因而能够更好地处理锯齿等问题（将 JPG 图片修改后缀名是不起作用的）。
- (x, y, scale) 参数设置的是水印图片相对于推流分辨率的归一化坐标。假设推流分辨率为：540 × 960，该字段设置为：（0.1，0.1，0.1），那么水印的实际像素坐标为：（540 × 0.1，960 × 0.1，水印宽度 × 0.1，水印高度会被自动计算）。

```
//设置视频水印
[_pusher setWatermark:[UIImage imageNamed:@"watermark"] x:0.03 y:0.015 scale:1];
```

15. 主播端弱网提醒

手机连接 Wi-Fi 网络不一定就非常好，如果 Wi-Fi 信号差或者出口带宽很有限，可能网速不如4G，如果主播在推流时遇到网络很差的情况，需要有一个友好的提示，提示主播应当切换网络。

通过 V2TXLivePusherObserver 里的 onWarning 可以捕获 **V2TXLIVE_WARNING_NETWORK_BUSY** 事件，它代表当前主播的网络已经非常糟糕，出现此事件即代表观众端会出现卡顿。此时就可以像上图一样在 UI 上弹出一个“弱网提示”。

```
- (void)onWarning:(V2TXLiveCode)code
message:(NSString *)msg
extraInfo:(NSDictionary *)extraInfo {
dispatch_async(dispatch_get_main_queue(), ^{
if (code == V2TXLIVE_WARNING_NETWORK_BUSY) {
[_notification displayNotificationWithMessage:
@"您当前的网络环境不佳，请尽快更换网络保证正常直播" forDuration:5];
}
});
}
```

16. 发送 SEI 消息

调用 V2TXLivePusher 中的 sendSeiMessage 接口可以发送 SEI 消息。所谓 SEI，是视频编码数据中规定的一种附加增强信息，平时一般不被使用，但我们可以在其中加入一些自定义消息，这些消息会被直播 CDN 转发到观众端。使用场景有：

- 答题直播：推流端将题目下发到观众端，可以做到“音-画-题”完美同步。
- 秀场直播：推流端将歌词下发到观众端，可以在播放端实时绘制出歌词特效，因而不受视频编码的降质影响。
- 在线教育：推流端将激光笔和涂鸦操作下发到观众端，可以在播放端实时地划圈划线。

由于自定义消息是直接被塞入视频数据中的，所以不能太大（几个字节比较合适），一般常用于塞入自定义的时间戳等信息。

```
int payloadType = 5;
NSString* msg = @"test";
[_pusher sendSeiMessage:payloadType data:[msg dataUsingEncoding:NSUTF8StringEncoding]];
```

常规开源播放器或者网页播放器是不能解析 SEI 消息的，必须使用 LiteAVSDK 中自带的 V2TXLivePlayer 才能解析这些消息：

1. 设置：

```
int payloadType = 5;
[_player enableReceiveSeiMessage:YES payloadType:payloadType];
```

2. 当 V2TXLivePlayer 所播放的视频流中有 SEI 消息时，会通过 V2TXLivePlayerObserver 中的 onReceiveSeiMessage 回调来接收该消息。

事件处理

事件监听

SDK 通过 V2TXLivePusherObserver 代理来监听推流相关的事件通知和错误通知，详细的事件表和错误码表请参见 错误码表。

错误通知

SDK 发现部分严重问题，推流无法继续。

事件 ID	数值	含义说明
V2TXLIVE_ERROR_FAILED	-1	暂未归类的通用错误

事件 ID	数值	含义说明
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法
V2TXLIVE_ERROR_REFUSED	-3	API 调用被拒绝
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	当前 API 不支持调用
V2TXLIVE_ERROR_INVALID_LICENSE	-5	license 不合法，调用失败
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	请求服务器超时
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求

警告事件

SDK 发现部分警告问题，但 WARNING 级别的事件都会触发一些尝试性的保护逻辑或者恢复逻辑，而且有很大概率能够恢复。

事件 ID	数值	含义说明
V2TXLIVE_WARNING_NETWORK_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	视频回放期间出现滞后
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	摄像头打开失败
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	摄像头正在被占用中，可尝试打开其他摄像头
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	麦克风打开失败
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败

事件 ID	数值	含义说明
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	当前系统不支持屏幕分享
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	录屏被系统中断

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2022-03-04 11:05:24

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

功能概述

摄像头推流，是指采集手机摄像头的画面以及麦克风的声音，进行编码之后再推送到直播云平台上。腾讯云视立方 LiteAVSDK 通过 V2TXLivePusher 接口提供摄像头推流能力，如下是 LiteAVSDK 腾讯云工具包 App 中演示摄像头推流的相关操作界面：

特别说明

真机调试： 由于 SDK 大量使用 Android 系统的音视频接口，这些接口在仿真模拟器下往往不能工作，推荐您尽量使用真机调试。

示例代码

所属平台	GitHub 地址	关键类
iOS	Github	CameraPushViewController.m
Android	Github	CameraPushMainActivity.java

> 说明：
> 除上述示例外，针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。
>
> - iOS：MLVB-API-Example
> - Android：MLVB-API-Example

功能对接

1. 下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

2. 给 SDK 配置 License 授权

单击 License 申请 获取测试用 License，您会获得两个字符串：其中一个字符串是 licenceURL ，另一个字符串是解密 licenceKey。
在您的 App 调用企业版 SDK 相关功能之前（建议在 Application类中）进行如下设置：

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // 获取到的 licence url
String licenceKey = ""; // 获取到的 licence key
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
```

```
}
}
```

3. 初始化 V2TXLivePusher 组件

创建一个 V2TXLivePusher 对象，该对象负责完成推流的主要工作。

```
V2TXLivePusher mLivePusher = new V2TXLivePusherImpl(this, V2TXLiveDef.V2TXLiveMode.TXLiveMode_RTMP); //指定对应的直播协议为 RTMP
```

4. 开启摄像头预览

想要开启摄像头的预览画面，您需要先给 SDK 提供一个用于显示视频画面的 TXCloudVideoView 对象，由于 TXCloudVideoView 是继承自 Android 中的 FrameLayout，所以您可以：

1. 直接在 xml 文件中添加一个视频渲染控件：

```
<com.tencent.rtmp.ui.TXCloudVideoView
android:id="@+id/pusher_tx_cloud_view"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

2. 通过调用 V2TXLivePusher 中的startCamera接口开启当前手机摄像头的预览画面。

```
//启动本地摄像头预览
TXCloudVideoView mPusherView = (TXCloudVideoView) findViewById(R.id.pusher_tx_cloud_view);
mLivePusher.setRenderView(mPusherView);
mLivePusher.startCamera(true);
```

5. 启动和结束推流

如果已经通过 startCamera 接口启动了摄像头预览，就可以调用 V2TXLivePusher 中的 startPush 接口开始推流。推流地址可以使用 TRTC 地址，或者使用 RTMP 地址，前者使用 UDP 协议，推流质量更高，并支持连麦互动。

```
//启动推流， URL 可以使用 trtc:// 或者 rtmp:// 两种协议，前者支持连麦功能
String rtmpURL = "trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx"; //支持连麦
String rtmpURL = "rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx"; //不支持连
```

```
麦，直接推流到直播 CDN
int ret = mLivePusher.startPush(rtmpURL.trim());
if (ret == V2TXLIVE_ERROR_INVALID_LICENSE) {
Log.i(TAG, "startRTMPPush: license 校验失败");
}
```

推流结束后，可以调用 V2TXLivePusher 中的 stopPush 接口结束推流。

```
//结束推流
mLivePusher.stopPush();
```

> ⚠ 注意：
> 如果已经启动了摄像头预览，请在结束推流时将其关闭。

- **如何获取可用的推流 URL**
  开通直播服务后，可以使用 **直播控制台 > 直播工具箱 > 地址生成器** 生成推流地址，详细信息请参见 推拉流 URL。

- **返回 V2TXLIVE_ERROR_INVALID_LICENSE 的原因?**
  如果 startPush 接口返回 V2TXLIVE_ERROR_INVALID_LICENSE，则代表您的 License 校验失败了，请检查 第2步：给 SDK 配置 License 授权 中的工作是否有问题。

6. 纯音频推流

如果您的直播场景是纯音频直播，不需要视频画面，那么您可以不执行 第4步 中的操作，或者在调用 startPush 之前调用 stopCamera 接口即可。

```
V2TXLivePusher mLivePusher = new V2TXLivePusherImpl(this, V2TXLiveDef.V2TXLiveMode.TXLiveMode_RTMP); //指定对应的直播协议为 RTMP
mLivePusher.startMicrophone();
//启动推流， URL 可以使用 trtc:// 或者 rtmp:// 两种协议，前者支持连麦功能
String rtmpURL = "trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx"; //支持连麦
String rtmpURL = "rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx"; //不支持连麦，直接推流到直播 CDN
int ret = mLivePusher.startPush(rtmpURL.trim());
```

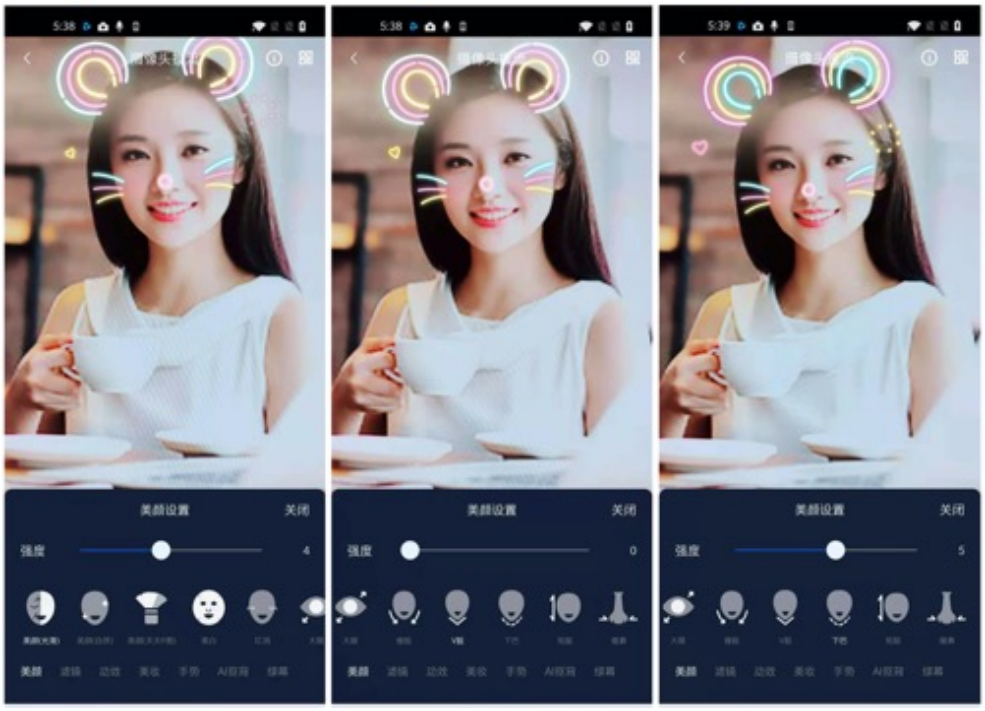
说明：

如果您启动纯音频推流，但是 RTMP、FLV 、HLS 格式的播放地址拉不到流，那是因为线路配置问题，请提工单 联系我们帮忙修改配置。

7. 设定画面清晰度

调用 V2TXLivePusher 中的 setVideoQuality 接口，可以设定观众端的画面清晰度。之所以说是观众端的画面清晰度，是因为主播看到的视频画面是未经编码压缩过的高清原画，不受设置的影响。而 setVideoQuality 设定的视频编码器的编码质量，观众端可以感受到画质的差异。详情请参见 设定画面质量。

8. 美颜美白和红润特效



调用 V2TXLivePusher 中的 getBeautyManager 接口可以获取 TXBeautyManager 实例进一步设置美颜效果。

美颜风格

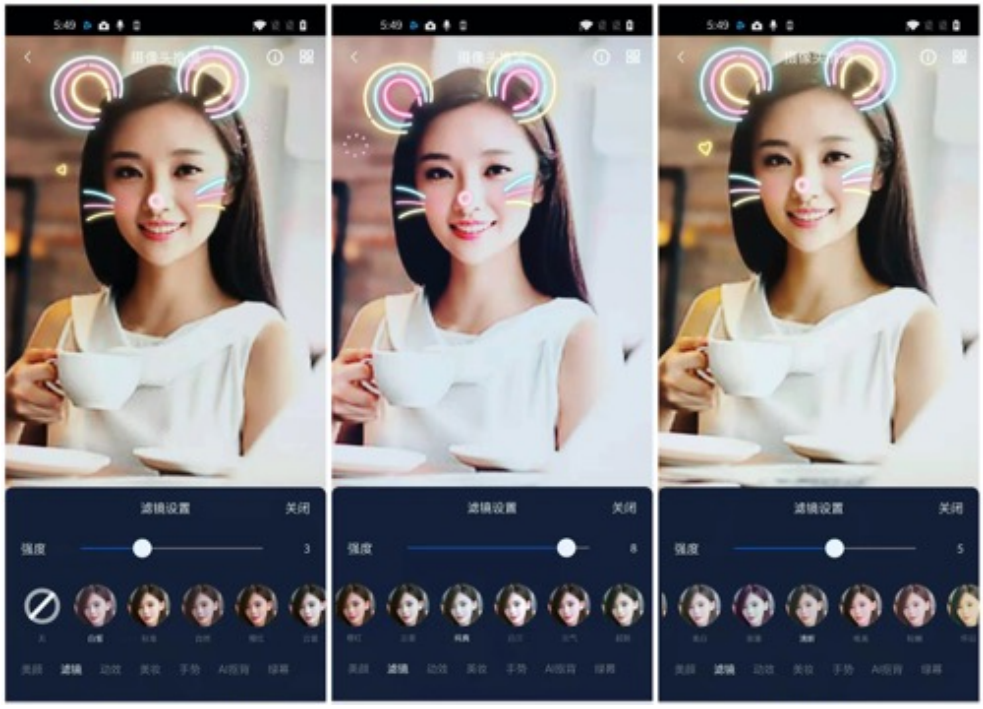
SDK 内置三种不同的磨皮算法，每种磨皮算法即对应一种美颜风格，您可以选择最适合您产品定位的方案。定义见 TXLiveConstants.java 文件：

美颜风格	效果说明
BEAUTY_STYLE_SMOOTH	光滑，适用于美女秀场，效果比较明显
BEAUTY_STYLE_NATURE	自然，磨皮算法更多地保留了面部细节，主观感受上会更加自然
BEAUTY_STYLE_PITU	由上海优图实验室提供的美颜算法，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高

美颜风格可以通过 TXBeautyManager 的 setBeautyStyle 接口设置：

美颜风格	设置方式	接口说明
美颜级别	通过 TXBeautyManager 的 setBeautyLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显
美白级别	通过 TXBeautyManager 的 setWhitenessLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显
红润级别	通过 TXBeautyManager 的 setRuddyLevel 设置	取值范围0 - 9； 0表示关闭，1 - 9值越大，效果越明显

9. 色彩滤镜效果



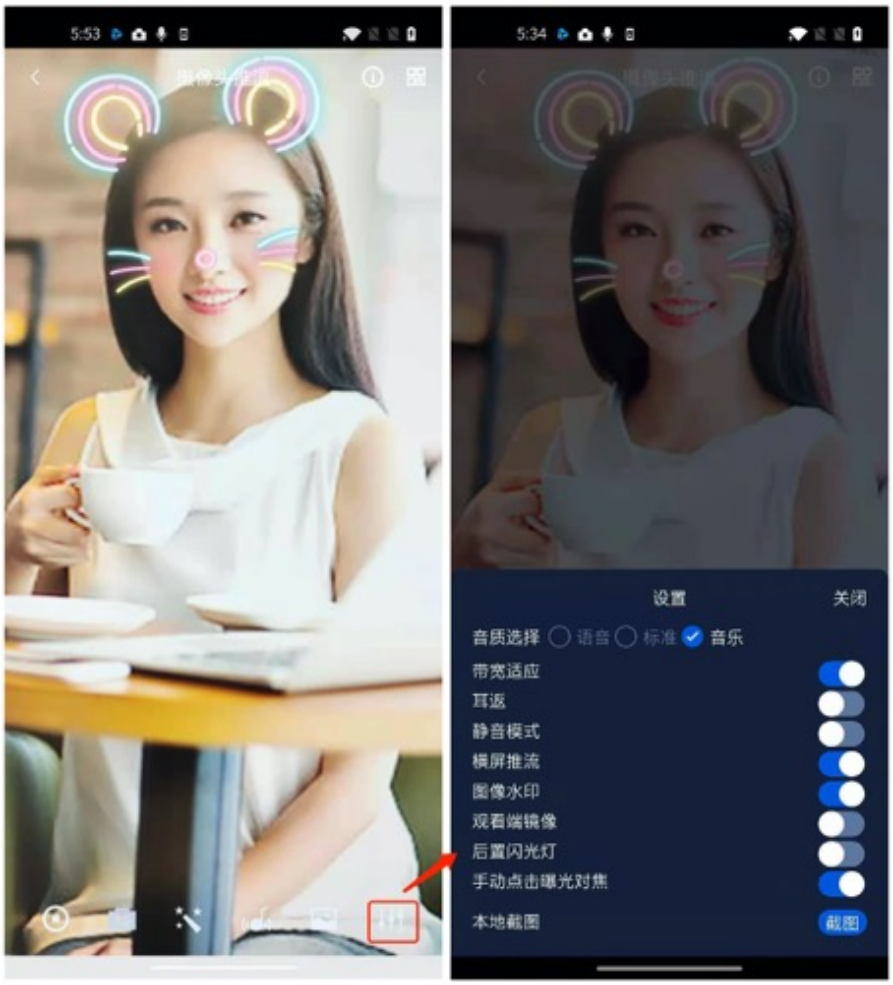
- 调用 V2TXLivePusher 中的 getBeautyManager 接口可以获取 TXBeautyManager 实例进一步设置美色彩滤镜效果。
- 调用 TXBeautyManager 的 setFilter 接口可以设置色彩滤镜效果。所谓色彩滤镜，是指一种将整个画面色调进行区域性调整的技术，例如将画面中的淡黄色区域淡化实现肤色亮白的效果，或者将整个画面的色彩调暖让视频的效果更加清新和温和。
- 调用 TXBeautyManager 的 setFilterStrength 接口可以设定滤镜的浓度，设置的浓度越高，滤镜效果也就越明显。

从手机 QQ 和 Now 直播的经验来看，单纯通过 TXBeautyManager 的 setBeautyStyle 调整美颜风格是不够的，只有将美颜风格和 setFilter 配合使用才能达到更加丰富的美颜效果。所以，我们的设计师团队提供了17种默认的色彩滤镜，并将其默认打包在了 Demo 中供您使用。

```
//选择期望的色彩滤镜文件
Bitmap filterBmp = decodeResource(getResources(), R.drawable.filter_biaozhun);
mLivePusher.getBeautyManager().setFilter(filterBmp);
mLivePusher.getBeautyManager().setFilterStrength(0.5f);
```

10. 设备管理

V2TXLivePusher 提供了一组 API 用户控制设备的行为。您通过 getDeviceManager 获取 TXDeviceManager 实例进一步进行设备管理，详细用法请参见 TXDeviceManager API。



11. 观众端的镜像效果

通过调用 V2TXLivePusher 的 setRenderMirror 可以改变摄像头的镜像方式，继而影响观众端观看到的镜像效果。之所以说是观众端的镜像效果，是因为当主播在使用前置摄像头直播时，默认情况下自己看到的画面会被 SDK 反转，这时主播就像照镜子一样，观众看到的效果和主播看到的是一致的。如下图所示：

12. 横屏推流

大多数情况下，主播习惯以“竖屏持握”手机进行直播拍摄，观众端看到的也是竖屏分辨率的画面（例如 540 × 960 这样的分辨率）；有时主播也会“横屏持握”手机，这时观众端期望能看到是横屏分辨率的画面（例如 960 × 540 这样的分辨率），如下图所示：

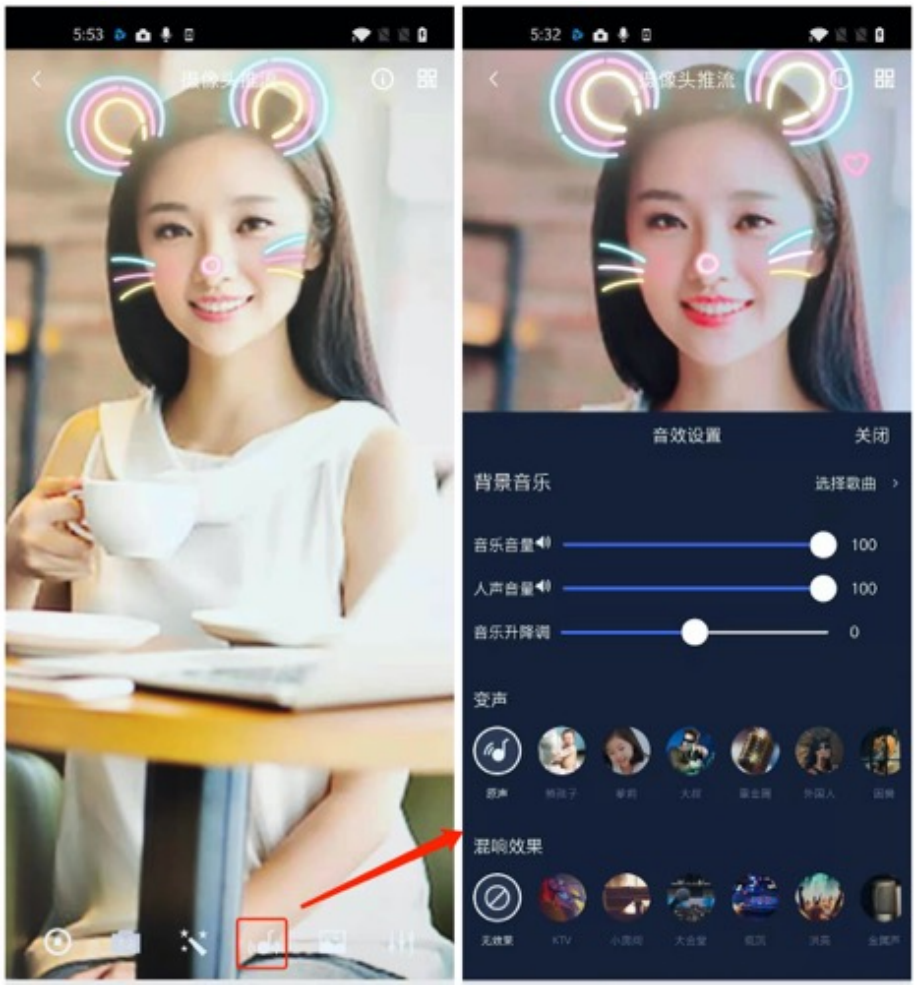
V2TXLivePusher 默认推出的是竖屏分辨率的视频画面，如果希望推出横屏分辨率的画面，可以修改 setVideoQuality 接口的参数来设定观众端的画面横竖屏模式。

```
mLivePusher.setVideoQuality(mVideoResolution, isLandscape ? V2TXLiveVideoResolutionModeLandscape : V2TXLiveVideoResolutionModePortrait);
```

13. 音效设置

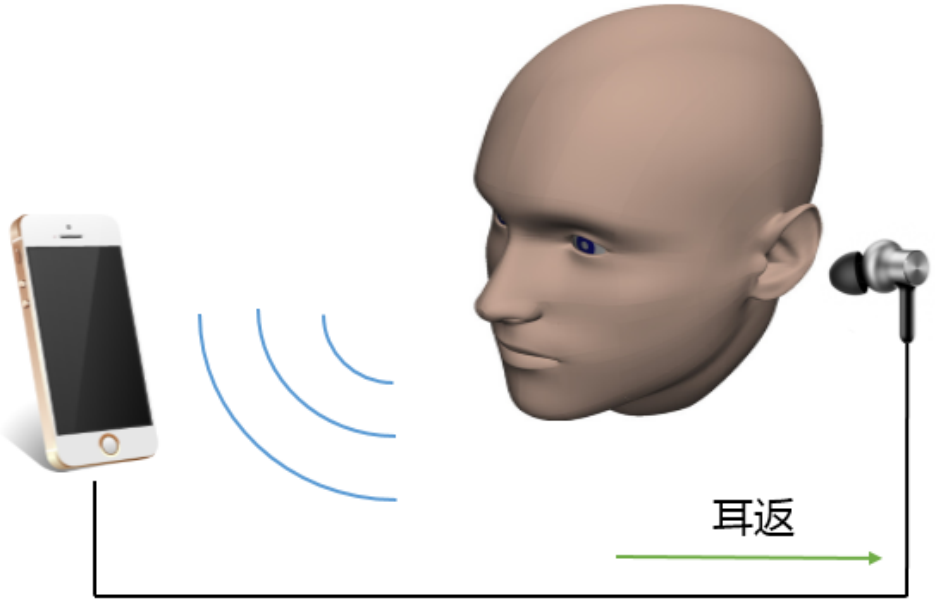
调用 V2TXLivePusher 中的 getAudioEffectManager 获取 TXAudioEffectManager 实例可以实现背景混音、耳返、混响等音效功能。背景混音是指主播在直播时可以选取一首歌曲伴唱，歌曲会在主播的手机端播放出来，

同时也会被混合到音视频流中被观众端听到，所以被称为“混音”。



- 调用 TXAudioEffectManager 中的 enableVoiceEarMonitor 选项可以开启耳返功能，“耳返”指的是当主播带上耳机来唱歌时，耳机中要能实时反馈主播的声音。
- 调用 TXAudioEffectManager 中的 setVoiceReverbType 接口可以设置混响效果，例如 KTV、会堂、磁性、金属等，这些效果也会作用到观众端。
- 调用 TXAudioEffectManager 中的 setVoiceChangerType 接口可以设置变调效果，例如“萝莉音”，“大叔音”等，用来增加直播和观众互动的趣味性，这些效果也会作用到观众端。

版权所有：腾讯云计算（北京）有限责任公司



耳返功能示意图

> 说明：
> 详细用法请参见 TXAudioEffectManager API。

14. 设置 Logo 水印

设置 V2TXLivePusher 中的 setWatermark 可以让 SDK 在推出的视频流中增加一个水印，水印位置位是由传入参数（x, y, scale）所决定。

- SDK 所要求的水印图片格式为 PNG 而不是 JPG，因为 PNG 图片格式有透明度信息，因而能够更好地处理锯齿等问题（将 JPG 图片修改后缀名是不起作用的）。
- (x, y, scale) 参数设置的是水印图片相对于推流分辨率的归一化坐标。假设推流分辨率为：540 × 960，该字段设置为：（0.1，0.1，0.1），则水印的实际像素坐标为：（540 × 0.1，960 × 0.1，水印宽度 × 0.1，水印高度会被自动计算）。

```
//设置视频水印
mLivePusher.setWatermark(BitmapFactory.decodeResource(getResources(),R.drawable.watermark), 0.03f, 0.015f, 1f);
```

15. 主播端弱网提醒

如果主播在推流时遇到网络很差的情况，需要有一个友好的提示，提示主播应当检查网络。
通过 V2TXLivePusherObserver 里的 onWarning 可以捕获 V2TXLIVE_WARNING_NETWORK_BUSY 事件，它代表当前主播的网络已经非常糟糕，出现此事件即代表观众端会出现卡顿。此时可以在 UI 上弹出一个“弱网提示”来强提醒主播检查网络。

```
@Override
public void onWarning(int code, String msg, Bundle extraInfo) {
if (code == V2TXLiveCode.V2TXLIVE_WARNING_NETWORK_BUSY) {
showNetBusyTips(); // 显示网络繁忙的提示
}
}
```

16. 发送 SEI 消息

调用 V2TXLivePusher 中的 sendSeiMessage 接口可以发送 SEI 消息。所谓 SEI，是视频编码数据中规定的一种附加增强信息，平时一般不被使用，但我们可以在其中加入一些自定义消息，这些消息会被直播 CDN 转发到观众端。使用场景有：

- 答题直播：推流端将题目下发到观众端，可以做到“音-画-题”完美同步。
- 秀场直播：推流端将歌词下发到观众端，可以在播放端实时绘制出歌词特效，因而不受视频编码的降质影响。
- 在线教育：推流端将激光笔和涂鸦操作下发到观众端，可以在播放端实时地划圈划线。

由于自定义消息是直接被塞入视频数据中的，所以不能太大（几个字节比较合适），一般常用于塞入自定义的时间戳等信息。

```
//Android 示例代码
int payloadType = 5;
String msg = "test";
mTXLivePusher.sendSeiMessage(payloadType, msg.getBytes("UTF-8"));
```

常规开源播放器或者网页播放器是不能解析 SEI 消息的，必须使用 LiteAVSDK 中自带的 V2TXLivePlayer 才能解析这些消息：

1. 设置：

```
int payloadType = 5;
mTXLivePlayer.enableReceiveSeiMessage(true, payloadType)
```

2. 当 V2TXLivePlayer 所播放的视频流中有 SEI 消息时，会通过 V2TXLivePlayerObserver 中的 onReceiveSeiMessage 回调来接收该消息。

事件处理

事件监听

SDK 通过 V2TXLivePusherObserver 代理来监听推流相关的事件通知和错误通知，详细的事件表和错误码表请参见 错误码表。

错误通知

SDK 发现部分严重问题，推流无法继续。

事件 ID	数值	含义说明
V2TXLIVE_ERROR_FAILED	-1	暂未归类的通用错误
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法
V2TXLIVE_ERROR_REFUSED	-3	API 调用被拒绝
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	当前 API 不支持调用
V2TXLIVE_ERROR_INVALID_LICENSE	-5	license 不合法，调用失败
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	请求服务器超时
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求

警告事件

SDK 发现部分警告问题，但 WARNING 级别的事件都会触发一些尝试性的保护逻辑或者恢复逻辑，而且有很大概率能够恢复。

事件 ID	数值	含义说明
V2TXLIVE_WARNING_NETWORK_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	当前视频播放出现卡顿
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	摄像头打开失败
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	摄像头正在被占用中，可尝试打开其他摄像头
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了

事件 ID	数值	含义说明
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	麦克风打开失败
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	当前系统不支持屏幕分享
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	录屏被系统中断

小程序

最近更新时间：2021-08-12 19:54:21

<live-pusher> 是小程序内部用于支持音视频上行能力的功能标签，本文主要介绍该标签的使用方法。

版本支持

- 微信 App iOS 最低版本要求：6.5.21。
- 微信 App Android 最低版本要求：6.5.19。
- 小程序基础库最低版本要求：1.7.0。

> 说明：
> 通过 wx.getSystemInfo 可以获取当前基础库版本信息。

使用限制

出于政策和合规的考虑，微信暂时没有放开所有小程序对 <live-pusher> 和 <live-player> 标签的支持：

- 个人账号和企业账号的小程序暂时只开放如下表格中的类目：

一级类目/主体类型	二级类目	资质要求	类目适用范围	小程序直播内容场景
社交	直播	（3选1）： 1.《信息网络传播视听节目许可证》 2.《网络文化经营许可证》(经营范围含网络表演） 3.《统一社会信用代码》及《情况说明函件》（适用于政府主体）	适用于提供在线直播等服务 注： 1.如提供时政信息服务，需补充：时政信息类目 2.选择该类目后首次提交代码审核，需经当地互联网主管机关审核确认，预计审核时长7天左右	涉及娱乐性质，如明星直播、生活趣事直播、宠物直播等。选择该类目后首次提交代码审核，需经当地互联网主管机关审核确认，预计审核时长7天左右
教育	在线视频课程	（5选1）： 1.《事业单位法人证书》（适用公立学校） 2. 区、县级教育部门颁发的《民办学校办学许可证》（适用培训机构） 3.《信息网络传播视听节目许可证》 4. 全国校外线上培训管理服	适用于教育行业提供，网课、在线培训、讲座等教育类视频/直播等服务	网课、在线培训、讲座等教育类直播

		务平台备案 5. 教育部门的批准文件		
医疗	互联网医院	（2选1）： 1. 卫生健康部门的《设置医疗机构批准书》； 2. 合作医院的《医疗机构执业许可证》与执业登记机关的审核合格文件	适用于互联网医院主体/医疗服务平台提供在线看诊、疾病咨询等线上医疗服务	问诊、大型健康讲座等直播
	公立医疗机构	《医疗机构执业许可证》与《事业单位法人证书》	适用于公立医疗机构提供的就医、健康咨询/问诊、医疗保健信息等服务	
金融	银行	（2选1）： 1.《金融许可证》 2.《金融机构许可证》	适用于提供银行业务在线服务或交易等服务	金融产品视频客服理赔、金融产品推广直播等
	信托	（2选1）： 1.《金融许可证》 2.《金融机构许可证》	适用于提供信托理财业务在线服务或交易等服务	
	公募基金	（3选1）： 1.《经营证券期货业务许可证》且业务范围必须包含“基金” 2.《基金托管业务许可证》 3.《基金销售业务资格证书》	适用于基金管理公司从事股票、债券等金融工具的投资服务	
	私募基金	（2选1）： 1.《私募基金备案证明》 2.《私募投资基金管理人登记证书》	仅适用于私募基金展示、介绍、咨询等服务 注：暂不支持涉及私募产品公开募集或在线交易等服务	
	证券/期货	《经营证券期货业务许可证》	适用于提供证券资讯、证券咨询、证券期货经营等的在线服务	
	证券、期货投资咨询	（2选1）： 1.《证券投资咨询业务资格证书》 2.《经营证券期货业务许可证》	适用于提供证券、期货投资等在线咨询服务	
	保险	（8选1）： 1.《保险公司法人许可证》 2.《经营保险业务许可证》	适用于提供保险业务在线服务或交易等服务	

		3.《保险营销服务许可证》 4.《经营保险代理业务许可证》 5.《经营保险经纪业务许可证》 6.《经营保险公估业务许可证》 7.《经营保险资产管理业务许可证》 8.《保险兼业代理业务许可证》		
	征信业务	（2选1）： 1.经营个人征信业务：《个人征信业务经营许可证》、《营业执照》 2.经营企业征信业务：经所在地的中国人民银行及其派出机构备案的《企业征信业务经营备案证》、《营业执照》	适用于银行或征信机构提供征信业务服务，包括：信贷记录、逾期记录、失信人查询等	
	新三板信息服务平台	全国中小企业股份转让系统有限责任公司的书面许可与《非经营性互联网信息服务备案核准》	适用于提供新三板信息行情资讯等服务	
	股票信息服务平台（港股/美股）	《非经营性互联网信息服务备案核准》	适用于提供港股、美股行情资讯、行情分析等服务 注：如提供股票交易服务，需补充：金融业-证券/期货类目	
	消费金融	银监会核准开业的审批文件与《金融许可证》与《营业执照》	适用于提供消费金融线上服务或交易等服务	
汽车	汽车预售服务	（3选1）： 1.汽车厂商：《营业执照》与《工信部道路机动车辆生产企业准入许可》 2.汽车经销商/4s店：《营业执照》与《厂商授权销售文件》与《工信部道路机动车辆生产企业准入许可》 3.下属子/分公司：《营业执照》与《工信部道路机动车	适用于提供汽车在线预付款等服务 注：平台暂不支持在线整车销售，如涉及整车销售服务，建议改为价格指导或移除相关功能	汽车预售、推广直播

		辆生产企业准入许可》与《股权关系证明函》（含双方盖章）		
政府主体帐号	–	–	–	政府相关工作推广直播、领导讲话直播等
工具	视频客服	–	适用于提供企业售后客服一对一视频等服务	不涉及以上几类内容的一对一视频客服服务，如企业售后一对一视频服务等

说明：

可申请直播标签的小程序类目以 微信文档 说明为主，小程序类目的资质要求详见 非个人主体类目申请。

- 符合类目要求的小程序，需要在小程序管理后台的【 开发管理 】>【 接口设置 】中自助开通该组件权限，如下图所示：

注意：

如果以上设置都正确，但小程序依然不能正常工作，可能是微信内部的缓存没更新，请删除小程序并重启微信后，再进行尝试。

属性定义

属性名	类型	默认值	说明
url	String	-	用于音视频上行的推流 URL
mode	String	RTC	SD，HD，FHD，RTC
autopush	Boolean	false	是否自动启动推流
muted	Boolean	false	是否静音
enable-camera	Boolean	true	开启\关闭摄像头
auto-focus	Boolean	true	手动\自动对焦
orientation	String	vertical	vertical，horizontal
beauty	Number	0	美颜指数，取值 0 - 9，数值越大效果越明显
whiteness	Number	0	美白指数，取值 0 - 9，数值越大效果越明显
aspect	String	9：16	3：4，9：16
zoom	Boolean	false	是否正常焦距，true 表示将摄像头放大
device-position	String	front	front 前置摄像头，back 后置摄像头
min-bitrate	Number	200	最小码率，该数值决定了画面最差的清晰度表现
max-bitrate	Number	1000	最大码率，该数值决定了画面最好的清晰度表现
audio-quality	String	low	low 适合语音通话，high 代表高音质
waiting-image	String	-	当微信切到后台时的垫片图片
waiting-image-hash	String	-	当微信切到后台时的垫片图片的校验值
background-mute	Boolean	false	当微信切到后台时是否禁用声音采集
bindstatechange	String	-	用于指定一个 javascript 函数来接收音视频事件
debug	Boolean	false	是否开启调试模式

示例代码

```
<view id='video-box'>
<live-pusher
```

```
id="pusher"
mode="RTC"
url="{{pusher.push_url}}"
autopush='true'
bindstatechange="onPush">
</live-pusher>
</view>
```

属性详解

- **url**
  用于音视频上行的推流 URL，以 rtmp:// 协议前缀开头，腾讯云推流 URL 的获取方法见 快速获取 URL 文档。

  > 说明：
  > 小程序内部使用的 RTMP 协议是支持 UDP 加速的版本，在同样网络条件下，UDP 版本的 RTMP 会比开源版本的有更好的上行速度和抗抖动能力。

- **mode**
  SD、HD 和 FHD 主要用于直播类场景，例如赛事直播、在线教育、远程培训等等。SD、HD 和 FHD 分别对应三种默认的清晰度。该模式下，小程序会更加注重清晰度和观看的流畅性，不会过分强调低延迟，也不会为了延迟牺牲画质和流畅性。

  RTC 则主要用于双向视频通话或多人视频通话场景，例如金融开会、在线客服、车险定损、培训会议等。该模式下，小程序会更加注重降低点到点的时延，也会优先保证声音的质量，在必要的时候会对画面清晰度和画面的流畅性进行一定的缩水。

- **orientation 和 aspect**
  横屏（horizontal）模式还是竖屏（vertical）模式，默认是竖屏模式，即 home 键朝下。这时，小程序推出的画面的宽高比是3：4或者9：16这两种竖屏宽高比的画面，也就是宽 < 高。如果改成横屏模式，小程序推出的画面宽高比即变为4：3或者16：9这种横屏宽高比的画面，也就是宽 > 高。

  具体的宽高比是由 aspect 决定的 ，默认是9：16，也可以支持3：4。这是在 orientation 的属性值为 vertical 的情况下。如果 orientation 的属性值为 horizontal，那么3：4的效果等价于4：3，9：16的效果等价于16：9。

- **min-bitrate 和 max-bitrate**
  这里首先要科普一个概念 —— 视频码率，指视频编码器每秒钟输出的视频数据的多少。在视频分辨率确定的情况下，视频码率越高，即每秒钟输出的数据越多，相应的画质也就越好。

所以 min-bitrate 和 max-bitrate 这两个属性，分别用于决定输出画面的最低清晰度和最高清晰度。这两个数值并非越大越好，因为用户的网络上行不是无限好的。但也不是越小越好，因为实际应用场景中，清晰与否是用户衡量产品体验的一个重要指标。具体的数值设定我们会在 参数设置 部分详细介绍。

小程序内部会自动处理好分辨率和码率的关系，例如2Mbps的码率，小程序会选择720p的分辨率进行匹配，而300kbps的码率下，小程序则会选择较低的分辨率来提高编码效率。所以您只需要关注 min-bitrate 和 max-bitrate 这一对参数就可以掌控画质。

- **waiting-image 和 waiting-image-hash**
  出于用户隐私的考虑，在微信切到后台以后，小程序希望停止摄像头的画面采集。但是对于另一端的用户而言，画面会变成黑屏或者冻屏（停留在最后一帧），这种体验是非常差的。为了解决这个问题，我们引入了 waiting-image 属性，您可以设置一张有 “稍候” 含义的图片（waiting-image 是该图片的 URL，waiting-image-hash 则是该图片对应的 md5 校验值）。当微信切到后台以后，小程序会使用该图片作为摄像头画面的替代，以极低的流量占用维持视频流3分钟时间。

- **debug**
  调试音视频相关功能，如果没有很好的工具会是一个噩梦，所以小程序为 live-pusher 标签支持了 debug 模式，开始 debug 模式之后，原本用于渲染视频画面的窗口上，会显示一个半透明的 log 窗口，用于展示各项音视频指标和事件，降低您调试相关功能的难度，具体使用方法我们在 FAQ 中有详细说明。

参数设置

这么多参数，具体要怎样设置才比较合适，我们给出如下建议值：

场景	mode	min-bitrate	max-bitrate	audio-quality	说明
标清直播	SD	300kbps	800kbps	high	窄带场景，例如户外或者网络不稳定的情况下适用
高清直播	HD	600kbps	1200kbps	high	目前主流的 App 所采用的参数设定，普通直播场景推荐使用这一档
超清直播	FHD	600kbps	1800kbps	high	对清晰度要求比较苛刻的场景，普通手机观看使用 HD 即可
视频客服（用户）	RTC	200kbps	500kbps	high	这是一种声音为主，画面为辅的场景，所以画质不要设置的太高
车险定损（车主）	RTC	200kbps	1200kbps	high	由于可能要看车况详情，画质上限会设置的高一些
多人会议（主讲）	RTC	200kbps	1000kbps	high	主讲人画质可以适当高一些，参与的质量可以设置的低一些

场景	mode	min-bitrate	max-bitrate	audio-quality	说明
多人会议（参与）	RTC	150kbps	300kbps	low	作为会议参与者，不需要太高的画质和音质

> 说明：
> 如果不是对带宽特别没有信心的应用场景，audio-quality 选项请不要选择 low，其音质和延迟感都会比 high 模式差很多。

对象操作

参数	说明
wx.createLivePusherContext()	通过 wx.createLivePusherContext() 可以将 <live-pusher> 标签和 javascript 对象关联起来，之后即可操作该对象
start	开始推流，如果 <live-pusher> 的 autopush 属性设置为 false（默认值），那么就可以使用 start 来手动开始推流
stop	停止推流
pause	暂停推流
resume	恢复推流，请与 pause 操作配对使用
switchCamera	切换前后摄像头
snapshot	推流截图，截图大小跟组件的大小一致。截图成功图片的临时路径为 ret.tempImagePath

```
var pusher = wx.createLivePusherContext('pusher');
pusher.start({
success: function(ret){
console.log('start push success!')
}
fail: function(){
console.log('start push failed!')
}
complete: function(){
console.log('start push complete!')
```

```
  }
});
```

内部事件

通过 <live-pusher> 标签的 bindstatechange 属性可以绑定一个事件处理函数，该函数可以监听推流模块的内部事件和异常通知。

常规事件

code	事件定义	含义说明
1001	PUSH_EVT_CONNECT_SUCC	已经成功连接到云端服务器
1002	PUSH_EVT_PUSH_BEGIN	与服务器握手完毕，一切正常，准备开始上行推流
1003	PUSH_EVT_OPEN_CAMERA_SUCC	已成功启动摄像头，摄像头被占用或者被限制权限的情况下无法打开

严重错误

code	事件定义	含义说明
-1301	PUSH_ERR_OPEN_CAMERA_FAIL	打开摄像头失败
-1302	PUSH_ERR_OPEN_MIC_FAIL	打开麦克风失败
-1303	PUSH_ERR_VIDEO_ENCODE_FAIL	视频编码失败
-1304	PUSH_ERR_AUDIO_ENCODE_FAIL	音频编码失败
-1305	PUSH_ERR_UNSUPPORTED_RESOLUTION	不支持的视频分辨率
-1306	PUSH_ERR_UNSUPPORTED_SAMPLERATE	不支持的音频采样率
-1307	PUSH_ERR_NET_DISCONNECT	网络断连，且经三次重连无效，可以放弃，更多重试请自行重启推流

警告事件

内部警告并非不可恢复的错误，小程序内部的音视频 SDK 会启动相应的恢复措施，警告的目的主要用于提示开发者或者最终用户，例如：

- **PUSH_WARNING_NET_BUSY**
  上行网速不给力，建议提示用户改善当前的网络环境，例如让用户离家里的路由器近一点，或者切到 Wi-Fi 环境下

再使用。

- **PUSH_WARNING_SERVER_DISCONNECT**
  请求被后台拒绝，出现这个问题一般是由于 URL 里的 txSecret 计算错，或者是 URL 被其他人占用（跟播放不同，一个推流 URL 同时只能有一个用户使用）。
- **PUSH_WARNING_HANDUP_STOP**
  当用户单击小程序右上角的圆圈或者返回按钮时，微信会将小程序挂起，此时 <live-pusher> 会抛出5000这个事件。

code	事件定义	含义说明
1101	PUSH_WARNING_NET_BUSY	上行网速不够用，建议提示用户改善当前的网络环境
1102	PUSH_WARNING_RECONNECT	网络断连，已启动重连流程（重试失败超过三次会放弃）
1103	PUSH_WARNING_HW_ACCELERATION_FAIL	硬编码启动失败，自动切换到软编码
1107	PUSH_WARNING_SWITCH_SWENC	由于机器性能问题，自动切换到硬件编码
3001	PUSH_WARNING_DNS_FAIL	DNS 解析失败，启动重试流程
3002	PUSH_WARNING_SEVER_CONN_FAIL	服务器连接失败，启动重试流程
3003	PUSH_WARNING_SHAKE_FAIL	服务器握手失败，启动重试流程
3004	PUSH_WARNING_SERVER_DISCONNECT	RTMP 服务器主动断开，请检查推流地址的合法性或防盗链有效期
3005	PUSH_WARNING_READ_WRITE_FAIL	RTMP 读/写失败，将会断开连接
5000	PUSH_WARNING_HANDUP_STOP	小程序被用户挂起，停止推流

示例代码

```
Page({
onPush: function(ret) {
if(ret.detail.code == 1002) {
console.log('推流成功了',ret);
}
},
```

```
/**
* 生命周期函数--监听页面加载
*/
onLoad: function (options) {
//...
}
})
```

录屏推流
iOS

最近更新时间：2022-03-18 19:04:52

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

概述

录屏功能是 iOS 10 新推出的特性，苹果在 iOS 9 的 ReplayKit 保存录屏视频的基础上，增加了视频流实时直播功能，官方介绍见 Go Live with ReplayKit。iOS 11 增强为 ReplayKit2，进一步提升了 Replaykit 的易用性和通用性，并且可以对整个手机实现屏幕录制，并非只是支持 ReplayKit 功能，因此录屏推流建议直接使用 iOS 11 的 ReplayKit2 屏幕录制方式。系统录屏采用的是扩展方式，扩展程序有单独的进程，iOS 系统为了保证系统流畅，给扩展程序的资源相对较少，扩展程序内存占用过大也会被 Kill 掉。腾讯云 LiteAV SDK 在原有直播的高质量、低延迟的基础上，进一步降低系统消耗，保证了扩展程序稳定。

> ⚠ 注意：
> 本文主要介绍 iOS 11 的 ReplayKit2 录屏使用 SDK 推流的方法，涉及 SDK 的使用介绍同样适用于其它方式的自定义推流。更详细的使用说明可以参考 Demo 里 TXReplayKit_Screen 文件夹示例代码。

功能体验

体验 iOS 录屏可以通过扫码进行安装 腾讯云视立方 App。

> ⚠ 注意：
> 录屏推流功能仅11.0以上系统可体验。

示例代码

针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

使用步骤

1. 打开控制中心，长按屏幕录制按钮，选择 **视频云工具包**。
2. 打开 **视频云工具包** > **推流演示（录屏推流）**，输入推流地址或单击 New 自动获取推流地址，单击 **开始推流**。

推流设置成功后，顶部通知栏会提示推流开始，此时您可以在其它设备上看到该手机的屏幕画面。单击手机状态栏的红条，即可停止推流。

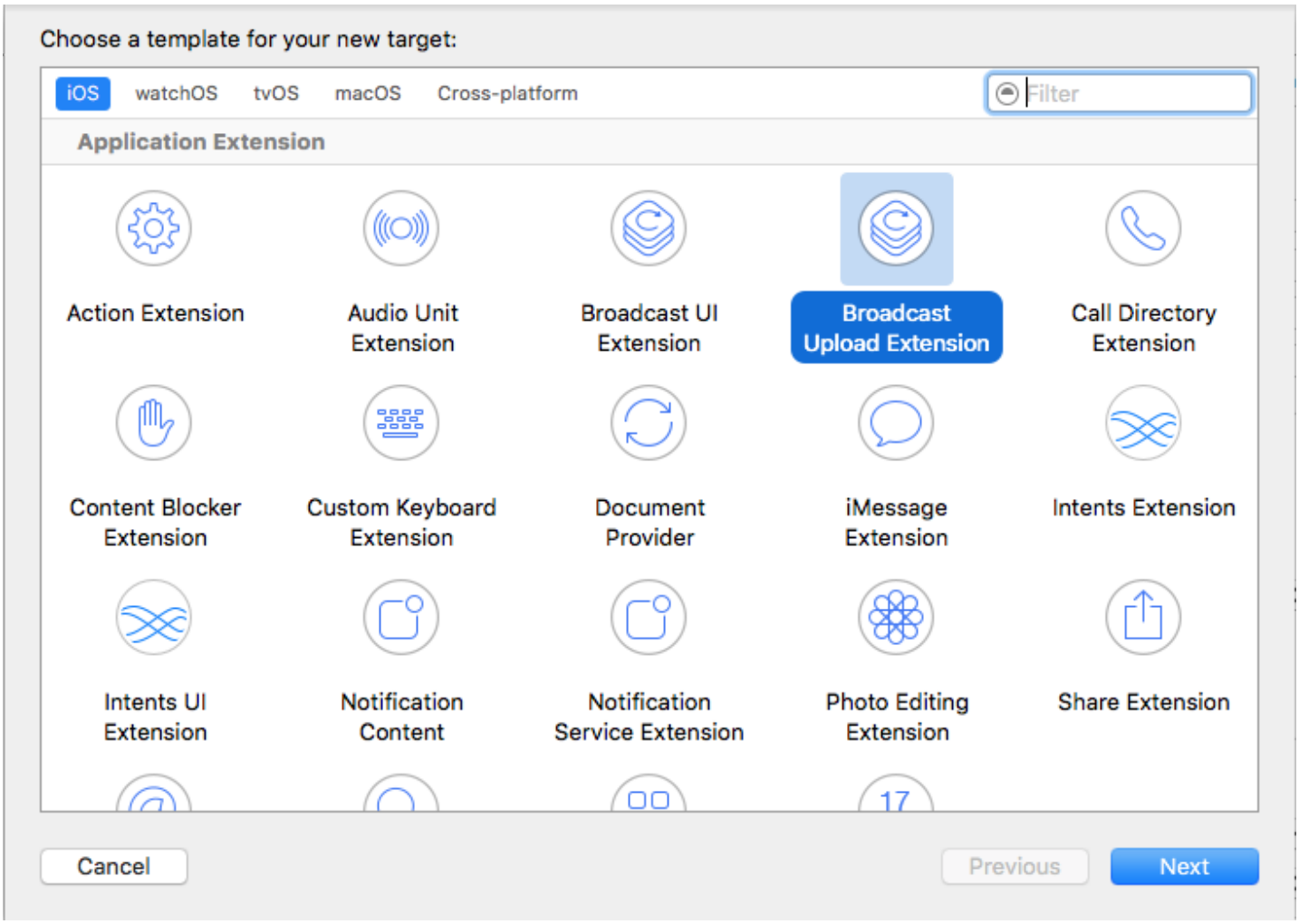
开发环境准备

Xcode 准备

Xcode 9 及以上的版本，手机也必须升级至 iOS 11 以上，否则模拟器无法使用录屏特性。

创建直播扩展

在现有工程选择 **New > Target…**，选择 **Broadcast Upload Extension**，如图所示。



配置好 Product Name。单击 **Finish** 后可以看到，工程多了所输 Product Name 的目录，目录下有个系统自动生成的 SampleHandler 类，这个类负责录屏的相关处理。

导入 LiteAV SDK

直播扩展需要导入 TXLiteAVSDK.framework。扩展导入 framework 的方式和主 App 导入方式相同，SDK 的系统依赖库也没有区别。具体请参见腾讯云官网 工程配置（iOS）。

对接流程

步骤1：创建推流对象

在 SampleHandler.m 中添加下面代码：

```
#import "SampleHandler.h"
#import "V2TXLivePusher.h"

static V2TXLivePusher *s_txLivePublisher;
static NSString *s_rtmpUrl;
```

```
- (void)initPublisher {
if (s_txLivePublisher) {
[s_txLivePublisher stopPush];
}
s_txLivePublisher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTMP];
[s_txLivePublisher setObserver:self];
[s_txLivePublisher startPush:s_rtmpUrl];
}
```

- s_txLivePublisher 是我们用于推流的对象，因为系统录屏回调的 sampleHandler 实例有可能不只一个，因此对变量采用静态声明，确保录屏推流过程中使用的是同一个推流器。
- 实例化 s_txLivePublisher 的最佳位置是在 -[SampleHandler broadcastStartedWithSetupInfo:] 方法中，直播扩展启动后会回调这个函数，就可以进行推流器初始化开始推流。但在 ReplayKit2 的屏幕录制扩展启动时，回调给 s_txLivePublisher 的 setupInfo 为 nil，无法获取启动推流所需要的推流地址等信息，因此通常回调此函数时发通知给主 App，在主 App 中设置好推流地址，横竖屏清晰度等信息后再传递给扩展并通知扩展启动推流。
- 扩展与主 App 间的通信请参见 扩展与宿主 App 之间的通信与数据传递方式 。

步骤2：横屏推流与分辨率设置

手机录屏直播提供了多个级别的分辨率可供选择。setVideoQuality 方法用来设置分辨率及横竖屏推流，以下录屏推流分辨率与横屏推流设置示例：

```
static BOOL s_landScape; //YES:横屏， NO:竖屏
[s_txLivePublisher setVideoQuality:V2TXLiveVideoResolution960x540
resolutionMode:s_landScape ? V2TXLiveVideoResolutionModeLandscape : V2TXLiveVideoResoluti
onModePortrait];
[s_txLivePublisher startPush:s_rtmpUrl];
```

步骤3：发送视频

Replaykit 会将视频以回调的方式传给 -[SampleHandler processSampleBuffer:withType] 。

```
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:(RPSampleBufferType)s
ampleBufferType {
switch (sampleBufferType) {
case RPSampleBufferTypeVideo:
```

```
// Handle video sample buffer
{
if (!CMSampleBufferIsValid(sampleBuffer))
return;
//保存一帧在 startPush 时发送,防止推流启动后或切换横竖屏因无画面数据而推流不成功
if (s_lastSampleBuffer) {
CFRelease(s_lastSampleBuffer);
s_lastSampleBuffer = NULL;
}
s_lastSampleBuffer = sampleBuffer;
CFRetain(s_lastSampleBuffer);
V2TXLiveVideoFrame *videoFrame = [V2TXLiveVideoFrame new];
videoFrame.bufferType = V2TXLiveBufferTypePixelBuffer;
videoFrame.pixelFormat = V2TXLivePixelFormatNV12;
videoFrame.pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);
videoFrame.rotation = V2TXLiveRotation0;
[s_txLivePublisher sendCustomVideoFrame:videoFrame];
}
}
```

视频 sampleBuffer 只需要调用 -[V2TXLivePusher sendCustomVideoFrame:] 发送即可。

系统分发视频 sampleBuffer 的频率并不固定，如果画面静止，可能很长时间才会有一帧数据过来。SDK 考虑到这种情况，内部会做补帧逻辑。

> ⚠ 注意：
> 建议保存一帧给推流启动时使用，防止推流启动或切换横竖屏时因无新的画面数据采集发送，因为画面没有变化时系统可能会很长时间才采集一帧画面。

步骤4：设置 Logo 水印

据相关政策规定，直播视频必须加上水印。腾讯视频云目前支持两种水印设置方式：一种是在推流 SDK 进行设置，原理是在 SDK 内部进行视频编码前就给画面打上水印。另一种方式是在云端打水印，也就是云端对视频进行解析并添加水印 Logo。

这里我们特别建议您使用 SDK 添加水印，因为在云端打水印有三个明显的问题：

- 这是一种很耗云端机器的服务，而且不是免费的，会拉高您的费用成本。
- 在云端打水印对于推流期间切换分辨率等情况的兼容并不理想，会有很多花屏的问题发生。
- 在云端打水印会引入额外的3s以上的视频延迟，这是转码服务所引入的。

SDK 所要求的水印图片格式为 PNG，因为 PNG 这种图片格式有透明度信息，因而能够更好地处理锯齿等问题（建议您不要在 Windows 下将 JPG 格式的图片修改后缀名就直接使用，因为专业的 PNG 图标都是需要由专业的美工设计师处理的）。

```
//设置视频水印
[s_txLivePublisher setWatermark:image x:0 y:0 scale:1];
```

步骤5：结束推流

结束推流 ReplayKit 会调用 -[SampleHandler broadcastFinished]，示例代码：

```
- (void)broadcastFinished {
// User has requested to finish the broadcast.
if (s_txLivePublisher) {
[s_txLivePublisher stopPush];
s_txLivePublisher = nil;
}
}
```

因为用于推流的 V2TXLivePusher对象同一时刻只能有一个在运行，所以结束推流时要做好清理工作。

事件处理

事件监听

SDK 通过 V2TXLivePusherObserver 代理来监听推流相关的事件通知和错误通知，详细的事件表和错误码表请参见 错误码表。

错误通知

SDK 发现部分严重问题，推流无法继续

事件 ID	数值	含义说明
V2TXLIVE_ERROR_FAILED	-1	暂未归类的通用错误。
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法。
V2TXLIVE_ERROR_REFUSED	-3	API 调用被拒绝。
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	当前 API 不支持调用。

事件 ID	数值	含义说明
V2TXLIVE_ERROR_INVALID_LICENSE	-5	license 不合法，调用失败。
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	请求服务器超时。
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求。

警告事件

SDK 发现部分警告问题，但 WARNING 级别的事件都会触发一些尝试性的保护逻辑或者恢复逻辑，而且有很大概率能够恢复。

事件 ID	数值	含义说明
V2TXLIVE_WARNING_NETWORK_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻。
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	当前视频播放出现卡顿
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	摄像头打开失败。
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	摄像头正在被占用中，可尝试打开其他摄像头。
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了。
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	麦克风打开失败。
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败。

事件 ID	数值	含义说明
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了。
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	当前系统不支持屏幕分享。
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了。
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	录屏被系统中断。

附: 扩展与宿主 App 之间的通信与数据传递方式参考

ReplayKit2 录屏只唤起 upload 直播扩展，直播扩展不能进行 UI 操作，也不适于做复杂的业务逻辑，因此通常宿主 App 负责鉴权及其它业务逻辑，直播扩展只负责进行屏幕的音画采集与推流发送，扩展就经常需要与宿主 App 之间进行数据传递与通信。

1. 发本地通知

扩展的状态需要反馈给用户，有时宿主 App 并未启动，此时可通过发送本地通知的方式进行状态反馈给用户与激活宿主 App 进行逻辑交互，如在直播扩展启动时通知宿主 App：

```
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *> *)setupInfo {
[self sendLocalNotificationToHostAppWithTitle:@"腾讯云录屏推流" msg:@"录屏已开始，请从这里单击回到Demo->录屏幕推流->设置推流URL与横竖屏和清晰度" userInfo:@{kReplayKit2UploadingKey:kReplayKit2Uploading}];
}

- (void)sendLocalNotificationToHostAppWithTitle:(NSString*)title msg:(NSString*)msg userInfo:(NSDictionary*)userInfo
{
UNUserNotificationCenter* center = [UNUserNotificationCenter currentNotificationCenter];

UNMutableNotificationContent* content = [[UNMutableNotificationContent alloc] init];
content.title = [NSString localizedUserNotificationStringForKey:title arguments:nil];
```

```
content.body = [NSString localizedUserNotificationStringForKey:msg arguments:nil];
content.sound = [UNNotificationSound defaultSound];
content.userInfo = userInfo;

// 在设定时间后推送本地推送
UNTimeIntervalNotificationTrigger* trigger = [UNTimeIntervalNotificationTrigger
triggerWithTimeInterval:0.1f repeats:NO];

UNNotificationRequest* request = [UNNotificationRequest requestWithIdentifier:@"ReplayKit2
Demo"
content:content trigger:trigger];

//添加推送成功后的处理!
[center addNotificationRequest:request withCompletionHandler:^(NSError * _Nullable error) {

}];
}
```

通过此通知可以提示用户回到主 App 设置推流信息、启动推流等。

2. 进程间的通知 CFNotificationCenter

扩展与宿主 App 之间还经常需要实时的交互处理，本地通知需要用户点击横幅才能触发代码处理，因此不能通过本地通知的方式。而 NSNotificationCenter 不能跨进程，因此可以利用 CFNotificationCenter 在宿主 App 与扩展之前通知发送，但此通知不能通过其中的 userInfo 字段进行数据传递，需要通过配置 App Group 方式使用 NSUserDefault 进行数据传递（也可以使用剪贴板，但剪贴板有时不能实时在进程间获取数据，需要加些延迟规避），如主 App 在获取好推流 URL 等后，通知扩展可以进行推流时，可通过 CFNotificationCenter 进行通知发送直播扩展开始推流：

```
CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter(),
kDarvinNotificationNamePushStart,
NULL,
nil,
YES);
```

扩展中可通过监听此开始推流通知，由于此通知是在 CF 层，需要通过 NSNotificationCenter 发送到 Cocoa 类层方便处理：

```
CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(),
(__bridge const void *)(self),
```

```
onDarwinReplayKit2PushStart,
kDarvinNotificationNamePushStart,
NULL,
CFNotificationSuspensionBehaviorDeliverImmediately);

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(handleReplayKit2P
ushStartNotification:) name:@"Cocoa_ReplayKit2_Push_Start" object:nil];


static void onDarwinReplayKit2PushStart(CFNotificationCenterRef center,
void *observer, CFStringRef name,
const void *object, CFDictionaryRef
userInfo)
{
//转到 cocoa 层框架处理
[[NSNotificationCenter defaultCenter] postNotificationName:@"Cocoa_ReplayKit2_Push_Start"
object:nil];
}

- (void)handleReplayKit2PushStartNotification:(NSNotification*)noti
{
//通过 NSUserDefault 或剪贴板拿到宿主要传递的数据
// NSUserDefaults *defaults = [[NSUserDefaults alloc] initWithSuiteName:kReplayKit2AppGrou
pId];

UIPasteboard* pb = [UIPasteboard generalPasteboard];
NSDictionary* defaults = [self jsonData2Dictionary:pb.string];

s_rtmpUrl = [defaults objectForKey:kReplayKit2PushUrlKey];
s_resolution = [defaults objectForKey:kReplayKit2ResolutionKey];
if (s_resolution.length < 1) {
s_resolution = kResolutionHD;
}
NSString* rotate = [defaults objectForKey:kReplayKit2RotateKey];
if ([rotate isEqualToString:kReplayKit2Portrait]) {
s_landScape = NO;
}
else {
s_landScape = YES;
}
```

```
[self start];
}
```

常见问题

ReplayKit2 屏幕录制在 iOS 11 新推出功能，相关的官方文档比较少，且存在着一些问题，使得每个版本的系统都在不断修复完善中。以下是一些使用中的常见现象或问题：

1. **屏幕录制何时自动会停止?**
   系统在锁屏或有电话打入时，会自动停止屏幕录制，此时 SampleHandler 里的 broadcastFinished 函数会被调用，可在此函数发通知提示用户。

2. **采集推流过程中有时屏幕录制会自动停止问题?**
   通常是因为设置的推流分辨率过高时在做横竖屏切换过程中容易出现。ReplayKit2 的直播扩展目前是有50M的内存使用限制，超过此限制系统会直接杀死扩展进程，因此 ReplayKit2 上建议推流分辨率不高于720P。

3. **iPhoneX 手机的兼容性与画面变形问题?**
   iPhoneX 手机因为有刘海，屏幕采集的画面分辨率不是 9:16。如果设了推流输出分辨率为 9:16 的比例，如高清里是为 960 × 540 的分辨率，这时因为源分辨率不是 9:16 的，推出去的画面就会稍有变形。建议设置分辨率时根据屏幕分辨率比例来设置，拉流端用 AspectFit 显示模式 iPhoneX 的屏幕采集推流会有黑边是正常现象，AspectFill 看画面会不全。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2022-03-04 11:04:29

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

功能介绍

手机录屏直播，即可以直接把主播的手机画面作为直播源，同时可以叠加摄像头预览，应用于游戏直播、移动端 App 演示等需要手机屏幕画面的场景。腾讯云 LiteAVSDK 通过 V2TXLivePusher 接口提供录屏推流能力，如下是 LiteAVSDK 腾讯云视立方 App 中演示摄像头推流的相关操作界面：

> 说明：
> 直播中叠加摄像头预览，即通过在手机上添加浮框，显示摄像头预览画面。录屏的时候会把浮框预览画面一并录制下来，达到叠加摄像头预览的效果。

限制说明

- Android 5.0 系统以后开始支持录屏功能。
- 悬浮窗在部分手机和系统上需要通过手动设置打开。

示例代码

针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

对接攻略

步骤1：创建 Pusher 对象

创建一个 V2TXLivePusher 对象，我们后面主要用它来完成推流工作。

```
V2TXLivePusher mLivePusher = new V2TXLivePusherImpl(context, V2TXLiveDef.V2TXLiveMode.TXLiveMode_RTMP);
```

步骤2：启动推流

```
String rtmpUrl = "rtmp://2157.livepush.myqcloud.com/live/xxxxxx";
mLivePusher.startMicrophone();
mLivePusher.startScreenCapture();
mLivePusher.startPush(rtmpUrl);
```

- **startScreenCapture** 的作用是启动屏幕录制，由于录屏是基于 Android 系统的原生能力实现的，处于安全考虑，Android 系统会在开始录屏前弹出提示，允许即可。
- **startPush** 的作用是告诉 LiteAV SDK 音视频流要推到哪个推流 URL 上去。

步骤3：设置 Logo 水印

设置 V2TXLivePusher 中的 setWatermark 可以让 SDK 在推出的视频流中增加一个水印，水印位置位是由传入参数（x, y, scale）所决定。

- SDK 所要求的水印图片格式为 PNG 而不是 JPG，因为 PNG 这种图片格式有透明度信息，因而能够更好地处理锯齿等问题（将 JPG 图片修改后缀名是不起作用的）。
- (x, y, scale) 参数设置的是水印图片相对于推流分辨率的归一化坐标。假设推流分辨率为：540 × 960，该字段设置为：（0.1，0.1，0.1），那么水印的实际像素坐标为：（540 × 0.1，960 × 0.1，水印宽度 × 0.1，水印高度会被自动计算）。

```
//设置视频水印
mLivePusher.setWatermark(BitmapFactory.decodeResource(getResources(),R.drawable.watermark), 0.03f, 0.015f, 1f);
```

步骤4：推荐的清晰度

调用 V2TXLivePusher 中的 setVideoQuality 接口，可以设定观众端的画面清晰度。之所以说是观众端的画面清晰度，是因为主播看到的视频画面是未经编码压缩过的高清原画，不受设置的影响。而 setVideoQuality 设定的视频编码器的编码质量，观众端可以感受到画质的差异。详情请参见 设定画面质量。

步骤5：提醒主播“网络不好”

手机连接 Wi-Fi 网络不一定就非常好，如果 Wi-Fi 信号差或者出口带宽很有限，可能网速不如4G，如果主播在推流时遇到网络很差的情况，需要有一个友好的提示，提示主播应当切换网络。

通过 V2TXLivePusherObserver 里的 onWarning 可以捕获 **V2TXLIVE_WARNING_NETWORK_BUSY** 事件，它代表当前主播的网络已经非常糟糕，出现此事件即代表观众端会出现卡顿。此时就可以像上图一样在 UI 上弹出一个“弱网提示”。

```
@Override
public void onWarning(int code, String msg, Bundle extraInfo) {
if (code == V2TXLiveCode.V2TXLIVE_WARNING_NETWORK_BUSY) {
showNetBusyTips(); // 显示网络繁忙的提示
}
}
```

步骤6：横竖屏适配

大多数情况下，主播习惯以“竖屏持握”手机进行直播拍摄，观众端看到的也是竖屏分辨率的画面（例如 540 × 960 这样的分辨率）；有时主播也会“横屏持握”手机，这时观众端期望能看到是横屏分辨率的画面（例如 960 ×

540 这样的分辨率），如下图所示：

V2TXLivePusher 默认推出的是竖屏分辨率的视频画面，如果希望推出横屏分辨率的画面，可以修改 setVideoQuality 接口的参数来设定观众端的画面横竖屏模式。

```
mLivePusher.setVideoQuality(mVideoResolution, isLandscape ? V2TXLiveVideoResolutionModeLandscape : V2TXLiveVideoResolutionModePortrait);
```

步骤7：结束推流

因为用于推流的 V2TXLivePusher 对象同一时刻只能有一个在运行，所以结束推流时要做好清理工作。

```
//结束录屏直播，注意做好清理工作
public void stopPublish() {
mLivePusher.stopScreenCapture();
mLivePusher.setObserver(null);
mLivePusher.stopPush();
}
```

事件处理

事件监听

SDK 通过 V2TXLivePusherObserver 代理来监听推流相关的事件通知和错误通知，详细的事件表和错误码表请参见 错误码表。

错误通知

SDK 发现部分严重问题，推流无法继续。

事件 ID	数值	含义说明
V2TXLIVE_ERROR_FAILED	-1	暂未归类的通用错误
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法
V2TXLIVE_ERROR_REFUSED	-3	API 调用被拒绝
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	当前 API 不支持调用
V2TXLIVE_ERROR_INVALID_LICENSE	-5	license 不合法，调用失败
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	请求服务器超时
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求

警告事件

SDK 发现部分警告问题，但 WARNING 级别的事件都会触发一些尝试性的保护逻辑或者恢复逻辑，而且有很大概率能够恢复。

事件 ID	数值	含义说明
V2TXLIVE_WARNING_NETWORK_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	当前视频播放出现卡顿
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	摄像头打开失败
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	摄像头正在被占用中，可尝试打开其他摄像头

事件 ID	数值	含义说明
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	麦克风打开失败
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	当前系统不支持屏幕分享
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	录屏被系统中断

进阶功能
录制和回看

最近更新时间：2021-12-23 17:00:10

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

功能介绍

录制回看是指您可以把用户整个直播过程录制下来，然后作为点播视频用于回看。

在 App 上线的初期阶段，由于主播数量比较少，所以在直播列表中加入录制回看，能够在一定程度上丰富 App 在观众端的信息量。即使到 App 成长起来主播数量形成规模以后，好的直播内容的沉淀依然是必不可少的一个部分，每个

主播的个人介绍里除了有名字、照片和个人信息，历史直播的视频回看更是不可或缺的一个重要组成部分。

开启录制

录制回看功能依托于腾讯云的**云点播服务**支撑，如果您想要对接这个功能，首先需要在腾讯云的管理控制台 开通云点播服务。服务开通之后，新录制的文件就可以在云点播控制台的 视频管理 里找到它们。

开启录制的方法如下：

> 说明：
>
> - 如需通过 API 对直播频道进行录制，详细请参见 创建录制任务。
> - 录制转点播后，文件自动存放于点播平台，故用户需在使用录制功能前，提前开通点播服务并购买相关空间和流量用于存放和播放录制后的视频文件，详细请参见 点播快速入门。

基本步骤

在云直播控制台菜单栏内选择 **功能配置** > **直播录制** ，单击 **创建录制模板** 进行设置。设置完基本信息后，单击 **保存** 。具体操作请参见 录制模板配置。

规格说明：

1. 录制视频针对直播原始码率录制，输出格式有 HLS、MP4、FLV 和 AAC 四种，其中 AAC 为纯音频录制。
2. 录制 MP4、FLV 或 AAC 格式单个文件时长限制为1分钟 - 120分钟。
3. 录制 HLS 格式最长单个文件时长无限制，如果超出续录超时时间则新建文件继续录制。
4. 单个录制文件保存最大时长均为1500天。
5. 仅 HLS 格式支持文件推流中断续录，续录超时时长可设置为0s - 1800s。
6. 直播过程中预计在录制结束5分钟左右可获取对应文件。例如，某直播从12:00开始录制，12:30结束录制，则12:35左右可获取12:00 - 12:30的对应片段。
7. 受限于音视频文件格式（FLV/MP4/HLS）对编码类型的支持，视频编码类型支持 H.264，音频编码类型支持 AAC。

版权所有：腾讯云计算（北京）有限责任公司

关联域名

创建好录制模板后，需在 **域名管理** 中，选择对应的推流域名，进入 **模板配置** 栏，单击 **录制配置 >编辑** 为该域名指定录制模板后，单击 **保存** 即可。更多详情请参见 关联录制模板。

获取文件

一个新的录制视频文件生成后，会相应的生成一个观看地址，您可以按照自己的业务需求对其进行处理。在小直播中，我们直接将录制的文件 URL 和房间列表拼在了一起，以弥补在线主播不足的窘境。

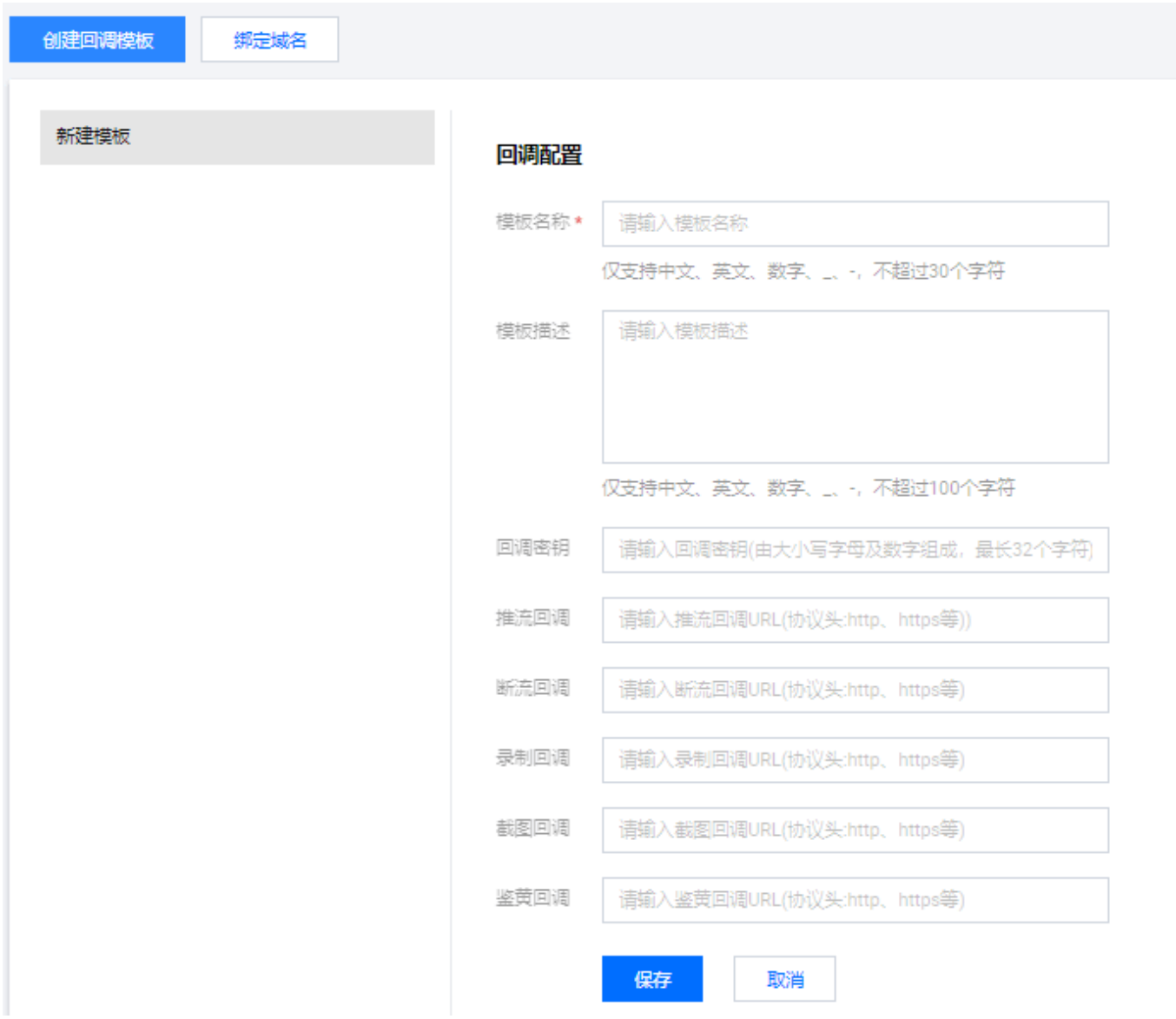
但您可以结合自己的业务场景实现很多的扩展功能，例如：您可以将其追加到主播的资料信息里，作为该主播曾经直播的节目而存在；或者将其放入回放列表中，经过专门的人工筛选，将优质的视频推荐给您的 App 用户。

如何才能拿到文件的地址，有如下两种解决方案：

方案1：被动监听通知

您可以使用腾讯云的 **回调配置**：您的服务器注册一个自己的**回调 URL** 给腾讯云，腾讯云会在一个新的录制文件生成时通过这个 URL 通知给您。

在云直播菜单栏内选择 **事件中心** > **直播回调**，单击 **创建回调模板** 。在回调设置弹框中填写完成回调信息，单击 **保存** 即可。更多详情请参见 创建回调模板。



关联域名

创建好回调模板后，需通在 **域名管理** 中，选择对应的推流域名，进入 **模板配置** 栏，单击 **回调配置 >编辑** 为该域名指定回调模板后，单击 **确定** 即可。

如下是一个典型的通知消息，它的含义是：一个 ID 为 9192487266581821586 的新的 FLV 录制文件已经生成，播放地址为：
http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv 。


```
{
"channel_id": "2121_15919131751",
"end_time": 1473125627,
"event_type": 100,
"file_format": "flv",
"file_id": "9192487266581821586",
"file_size": 9749353,
"sign": "fef79a097458ed80b5f5574cbc13e1fd",
"start_time": 1473135647,
"stream_id": "2121_15919131751",
"t": 1473126233,
"video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
"video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv"
}
```


方案2：主动查询获取

录制文件生成后自动存储到点播系统，您可以直接在点播系统查看，或直接通过点播 API 查询，详情请参见 录制文件获取 。

常见问题

1. 直播录制的原理是什么?
2. 一次直播会有几个录制文件?
3. 如何知道哪些文件属于某一次直播?
4. 如何把碎片拼接起来?

> 说明：
> 更多录制和回放相关问题，请参见 直播录制相关。

转封装及转码

最近更新时间：2021-12-23 17:00:17

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

直播转封装功能

直播转封装功能，是指将直播现场推送出来的原始流（一般使用 RTMP 协议推送到云端），在云端转换为不同的封装格式的视频流推送给观众，同时支持输出纯音频或者纯视频功能，也支持各种不同的 DRM 加密方案，以满足数字版权保护的需求。

支持类型	说明
输出封装格式	RTMP、FLV、HLS、DASH、HDS、TS 流
选择指定媒介输出	• 纯音频输出：即删除视频媒介，只输出音频媒介，封装格式如前文所述 • 纯视频输出：即删除音频媒介，只输出视频媒介，封装格式如前文所述
媒体加密方案	• **Fairplay**：HLS 封装支持 Apple fairplay drm 解决方案 • **Widevine**：DASH 封装支持 Google widevine drm 解决方案 • **HLS 通用 Aes-128 加密**：HLS 封装支持使用通用的 Aes-128 的加密方案

直播转码功能

直播转码功能（包含视频转码和音频转码），是指将直播现场推送出来的原始流，在云端转换为不同编码格式、不同分辨率、不同码率的转码流推送给观众，以满足不同网络环境、不同终端设备等各种场景下的播放需求。

典型应用场景举例

- 将原始视频流转换为不同清晰度的转码流，用户可以根据自己的网络情况选择不同码率的视频流进行播放，以保证播放的流畅性。

- 将原始视频流中打上官方自定义水印，给视频加上独有的标志，以标明版权，也可以起到宣传推广的效果。
- 将视频流转换为编码压缩率更优的视频编码格式，例如某一条原始编码格式为 H264 的视频流，在观看的人数比较大的情况下，可以尝试将原始 H264 的视频流转换为压缩率更高的 H265 视频流，从而能够达到节省带宽，节约成本的效果。
- 将原始视频流转化为不同的编码格式，以适应特殊终端的播放需求，例如有些特殊场景下解码 H264 视频由于性能问题无法达到实时播放的体验，需要将原始 H264 流转换为 Mpeg 编码格式的视频，从而达到终端实时解码播放的体验。

视频转码参数

参数类型	说明
视频编码方式	支持以下视频编码格式： • H264 • H265
视频编码档次	支持以下三种编码档次： • Baseline • Main • High
视频编码码率	• 支持视频输出码率范围：50kbps – 10Mbps。 • 支持当指定输出码率大于输入原始码率时，输出码率保持为原始码率。例如指定输出码率为3000kbps，但是原始输入流的码率只有2000kpbs，这种情况下可以保持输出码率为2000kbps。
视频编码帧率	• 支持视频输出帧率范围：1fps – 60fps。 • 支持当指定输出帧率大于输入原始帧率时，输出帧率保持为原始帧率。例如指定输出帧率为30fps，但是原始输入流的帧率只有20fps，这种情况下可以保持输出帧率为20fps。
视频分辨率	• 支持宽度范围：0 – 3000。 • 支持高度范围：0 – 3000。 • 支持单独指定宽度，高度依照宽度等比例缩放。 • 支持单独指定高度，宽度依照高度等比例缩放。
视频 GOP 长度	支持视频 GOP 长度范围：1秒 – 10秒，一般建议2秒 – 4秒。
视频码率控制方法	支持如下两种码率控制方法： • 固定比特率（CBR）。 • 动态比特率（VBR）。
视频画面旋转	支持将原视频画面顺时针旋转3个角度： • 顺时针旋转90度。 • 顺时针旋转180度。 • 顺时针旋转270度。

音频转码参数介绍

参数类型	说明
音频编码方式	支持以下编码规格： • AAC-LC • AAC-HE • AAC-HEV2
音频采样率	支持以下常用采样率，常用的采样率为48000和44100。 • 96000 • 64000 • 48000 • 44100 • 32000 • 24000 • 16000 • 12000 • 8000
音频编码码率	音频支持码率范围：20kbps - 192kbps，常用音频码率如下： • 48kbps • 64kbps • 128kbps
音频声道数	音频支持以下声道数： • 单声道 • 双声道

视频转码常用预设模板

清晰度	模板名称	视频分辨率	视频码率	视频帧率	视频编码格式
流畅	550	按比例缩放 * 540	500kbps	23	H264
标清	900	按比例缩放 * 720	1000kbps	25	H264
高清	2000	按比例缩放 * 1080	2000kbps	25	H264

极速高清转码功能

基于腾讯视频云多年音视频编码技术积累、智能场景识别、动态编码技术、CTU/行/帧三级码率精准控制模型，为直播、点播等行业以更低的码率（平均节省30%+）提供更高清的流媒体服务。

场景举例

当直播推流码率高并且画面复杂时，可以通过智能动态技术和码率精准控制模型，实现高清低码，保证同等主观画质。

功能优势

随着当今各视频平台用户对视频源清晰度观看体验要求越来越高，当前直播行业1080P、码率3Mbps－10Mbps已逐渐成为主流配置，所以带宽成本已经占据了视频平台成本比例中很大的一部分。而降低视频码率能够十分有效的减少带宽成本。

示例：

一场标准直播码率为3Mbps，直播时间4小时，观看人数200人，编码方式使用 H.264，分辨率1080P，不使用极速高清转码时，使用 直播价格计算器 计算产生的带宽成本为372元。

- 使用极速高清转码降低码率之后，产生的带宽成本大约为：372 ×（100% − 30%）= 260.4元。
- 使用极速高清转码产生的费用：0.2511 × 240 = 60.264元（刊例价格，不计算任何折扣）。
- 成本总计：260.4 + 60.264 = 320.664元。

因此极速高清转码在为用户提供更优质的观看体验的同时，十分有效的降低了平台的带宽成本。

主要参数

极速高清转码与标准直播转码的参数配置方法基本一致，请参见 视频转码参数。

直播添加水印功能介绍

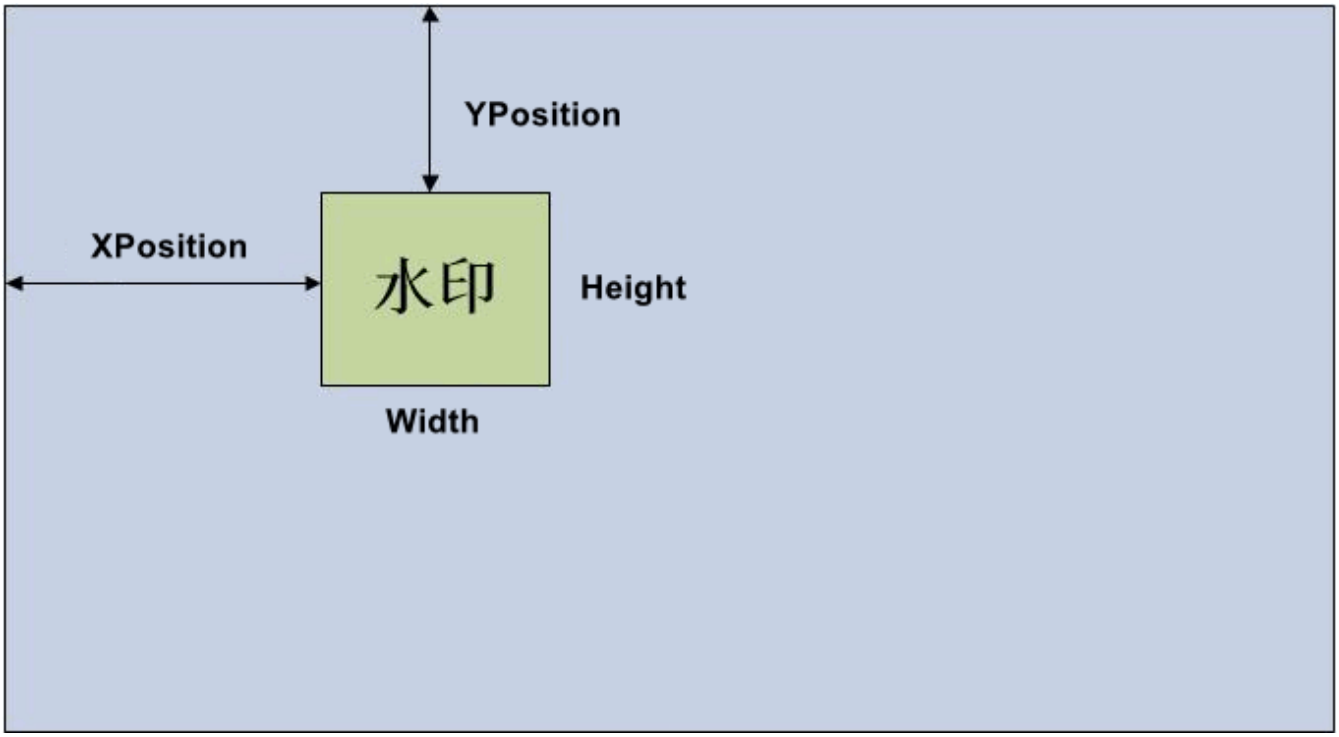
视频流添加水印功能，指的是在原始流的基础上，给视频画面添加上各种预设好的标志图片，给视频加上独有的标志，以标明版权，也可以起到宣传推广的效果。

水印相关参数

水印的主要参数包括水印位置和水印大小，水印位置和大小由参数 XPosition、YPosition、Width 和 Height 决定，各参数含义如下：

- **XPosition**：X 轴偏移，表示水印左侧和视频左侧边沿X轴偏移的百分比距离。
- **YPosition**：Y 轴偏移，表示水印顶部和视频顶部边沿Y轴偏移的百分比距离。
- **Width**：水印宽度，占直播原始画面宽度百分比。
- **Height**：水印高度，占直播原始画面高度百分比。

版权所有：腾讯云计算（北京）有限责任公司



> **注意：**
> 当一个流启用了多种码率转码时（即一路原始流转化为多种分辨率的转码流），这种情况下需要添加水印，可以在 云直播控制台 或 API 中设置 X 轴，Y 轴的百分比位置，系统会自适应水印的位置。

水印参数例子

视频输出画面为1920×1080，水印大小为320×240，使用百分比计算方式 XPosition = 5，YPosition = 5，Width=10。
根据视频输出画面的分辨率计算出水印的绝对位置和大小如下：

```
XPosition_pixel = 1920 * 5% = 96
YPosition_pixel = 1080 * 5% = 54
Width_pixel = 1920 * 10% = 192
Height_pixel = 192 * 240 / 320 = 144
```

因此，水印位置距离视频视频画面左边沿为96像素，距离视频输出画面上边沿为54像素，水印大小为192像素 * 144像素。

使用方法概述

添加水印可以通过两种方式来实现，一种是通过 云直播控制台 来实现，另一种是通过 服务端 API 来实现。根据您的业务需求选择对应的使用方式。

直播控制台

1. 进入**功能配置** > **直播水印** 添加水印配置模板，设置水印相关参数，并生成一个对应的水印模板 ID。具体操作请参见 直播水印。
2. 在**域名管理**对您需操作的推流域名，单击**管理** > **模板配置**，将此域名与水印模板进行关联。具体操作请参见 水印配置。

API 调用

1 调用 AddLiveWatermark 添加水印接口，设置水印的名称和参数信息。
2 调用 CreateLiveWatermarkRule 创建水印规则，设置参数推流域名 DomainName 和 WatermarkId（第1步返回），AppName 与推流和播放地址中的 AppName 保持一致，默认为 live 。

> ⚠ 注意：
> 使用添加水印功能，会产生标准转码费用。

转码参数设置使用方法

使用方法概述

设置转码参数可以通过两种方式来实现，一种是通过 云直播控制台 来实现，另一种是通过 服务端 API 来实现。不管使用哪一种方式，主要涉及水印模板、转码模板、转码规则的相关操作。

直播控制台

1. 进入**功能配置** > **直播转码** 添加转码配置模板，支持添加 标准转码、极速高清转码 和 纯音频转码 模板。
2. 根据您的需求创建对应的转码类型，设置转码相关参数。也可以使用我们系统默认的参数，并生成一个对应的转码模板 ID。
3. 在**域名管理**找到您需操作的拉流域名，单击**管理** > **模板配置**，将此域名与转码模板进行关联。具体操作请参见 转码配置。

API 调用

1. 调用 CreateLiveTranscodeTemplate 创建转码模板接口，设置您需要的转码类型参数信息。
2. 调用 CreateLiveTranscodeRule 创建转码规则，设置参数拉流域名 DomainName 和 TemplateId（第1步返回）。在 AppName 及 StreamName 填写空字符串，表示通配此域名下所有拉流的转码。您还可以将转码模板与不同流名称进行关联，以此实现部分直播流开启转码的效果。
3. 每一个转码模板都有对应一个**唯一的转码模板名称**，转码模板名称作为播放转码流的唯一标识，把转码模板名称添加到播放拉流地址中流 ID 名称的后面，就可以拉取到对应各种转码模板的转码流。

> ⚠ 注意：
> 转码规则的操作，主要用来控制某个域名或者某条流启用某个转码模板，只有创建了转码规则，对应的播放域名拉取相应的转码模板才能生效。若没有创建转码规则，直接使用转码模板名称拼接的拉流地址是无效的。

使用方法举例

播放地址 = 播放域名 + 播放路径 + 流 ID 名称_转码模板名称 + ? + 鉴权串

对于一个推流，流 ID 为 1234_test，通过以下3个地址可以播放不同带水印码流的流：

- **原始流：** http://liveplay.tcloud.com/live/1234_test.flv?鉴权串
- **标清转码流（带水印）：** http://liveplay.tcloud.com/live/1234_test_sd.flv?鉴权串
- **高清流转码流（带水印）：** http://liveplay.tcloud.com/live/1234_test_hd.flv?鉴权串

> ⚠ **注意：**
> 播放带水印的流，需要对应的推流域名绑定创建的水印模板。

使用接口

- **通过控制台管理转码模板：**
  控制台支持查询、添加、修改和删除转码模板，详细操作请参见 直播转码。
- **通过服务端 API 管理转码模板：**

功能模块	API 接口
直播转码	创建转码模板
	修改转码模板配置
	获取单个转码模板
	获取转码模板列表
	删除转码模板
	创建转码规则
	获取转码规则列表
	删除转码规则
直播水印	添加水印
	更新水印
	删除水印
	查询水印列表

自定义采集和渲染
iOS+Android

最近更新时间：2022-03-04 11:06:02

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

示例代码

针对开发者的接入反馈的高频问题，腾讯云视立方提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

定制推流画面

iOS 平台

方案一：修改 OpenGL 纹理

如果您有自定义图像处理的需求（例如：堆加字幕），同时又希望复用 LiteAV SDK 的整体流程，您可以按照如下攻略进行定制。

1. 首先需要调用 V2TXLivePusher 的 enableCustomVideoProcess 开启自定义视频处理，才会收到这个回调通知。
2. 处理图像时分为两种情况。
   - **美颜组件会产生新的纹理：**
     如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函

数中将 dstFrame.textureId 设置为新纹理的 ID。

```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:(V2TXLiveVideoFrame _Nonnull)dstFrame
{
GLuint dstTextureId = renderItemWithTexture(srcFrame.textureId, srcFrame.width, srcFrame.height);
dstFrame.textureId = dstTextureId;
}
```

- **美颜组件并不自身产生新纹理：**
  如果您使用的第三方美颜模块并不生成新的纹理，而是需要您设置给该模块一个输入纹理和一个输出纹理，则可以考虑如下方案：

```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:(V2TXLiveVideoFrame _Nonnull)dstFrame
{
thirdparty_process(srcFrame.textureId, srcFrame.width, srcFrame.height, dstFrame.textureId);
}
```

3. 最后，对于 texture 数据的操作，需要一定的 OpenGL 基础知识，另外计算量不宜太大，因为 onProcessVideoFrame 的调用频率跟 FPS 相同，过于繁重的处理很容易造成 GPU 过热。

方案二：自己采集数据

如果您只希望使用 SDK 来编码和推流（例如已经对接了商汤等产品），视频采集和预处理（即美颜、滤镜这些）全部由自己的代码来控制，可以按如下步骤实现：

1. 调用 V2TXLivePusher 的 enableCustomVideoCapture 接口开启自定义采集。
   这样 SDK 本身就不会再采集视频数据，而只是启动编码、流控、发送等跟推流相关的工作。
2. 通过 V2TXLivePusher 的 sendCustomVideoFrame 向 SDK 填充 Video 数据。

```
/**
* @brief 在自定义视频采集模式下，将采集的视频数据发送到SDK。<br/>
* 在自定义视频采集模式下，SDK不再采集摄像头数据，仅保留编码和发送功能。
* 您可以把采集到的 SampleBuffer 打包到 V2TXLiveVideoFrame 中，然后通过该API定期的发送。
*
* @note 需要在 [startPush](@ref V2TXLivePusher#startPush:) 之前调用 [enableCustomVideoCapture](@ref V2TXLivePusher#enableCustomVideoCapture:) 开启自定义采集。
*
```

```
* @param videoFrame 向 SDK 发送的 视频帧数据 {@link V2TXLiveVideoFrame}
*
* @return 返回值 {@link V2TXLiveCode}
* - V2TXLIVE_OK: 成功
* - V2TXLIVE_ERROR_INVALID_PARAMETER: 发送失败，视频帧数据不合法
* - V2TXLIVE_ERROR_REFUSED: 您必须先调用 enableCustomVideoCapture 开启自定义视频采集。
*/
- (V2TXLiveCode)sendCustomVideoFrame:(V2TXLiveVideoFrame *)videoFrame;
```

Android 平台

方案一：修改 OpenGL 纹理

如果您有自定义图像处理的需求（例如堆加字幕），同时又希望复用 LiteAV SDK 的整体流程，您可以按照如下攻略进行定制。

1. 首先需要调用 V2TXLivePusher 的 enableCustomVideoProcess 开启自定义视频处理，才会收到这个回调通知。
2. 处理图像时分为两种情况。
   - **美颜组件会产生新的纹理：**
     如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函数中将 dstFrame.textureId 设置为新纹理的 ID。

```
private class MyPusherObserver extends V2TXLivePusherObserver {
@Override
public void onGLContextCreated() {
mFURenderer.onSurfaceCreated();
mFURenderer.setUseTexAsync(true);
}

@Override
public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFrame dstFrame) {
dstFrame.texture.textureId = mFURenderer.onDrawFrameSingleInput(
srcFrame.texture.textureId, srcFrame.width, srcFrame.height);
return 0;
}

@Override
public void onGLContextDestroyed() {
```

```
mFURenderer.onSurfaceDestroyed();
}
}
```

- **美颜组件并不自身产生新纹理：**
  如果您使用的第三方美颜模块并不生成新的纹理，而是需要您设置给该模块一个输入纹理和一个输出纹理，则可以考虑如下方案：

```
@Override
public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFrame dstFra
me) {
thirdparty_process(srcFrame.texture.textureId, srcFrame.width, srcFrame.height, dstFrame.t
exture.textureId);
return 0;
}
```

3. **最后，对于 texture 数据的操作，需要一定的 OpenGL 基础知识，另外计算量不宜太大，因为 onProcessVideoFrame 的调用频率跟 FPS 相同，过于繁重的处理很容易造成 GPU 过热。**

方案二：自己采集数据

如果您只希望使用 SDK 来编码和推流（例如已经对接了商汤等产品），视频采集预处理（即美颜、滤镜这些）全部由自己的代码来控制，可以按如下步骤实现：

1. **调用 V2TXLivePusher 的 enableCustomVideoCapture 接口开启自定义采集。**
   这样 SDK 本身就不会再采集视频数据，而只是启动编码、流控、发送等跟推流相关的工作。
2. **通过 V2TXLivePusher 的 sendCustomVideoFrame 向 SDK 填充 Video 数据。**

```
/**
* @brief 在自定义视频采集模式下，将采集的视频数据发送到SDK。 <br/>
* 在自定义视频采集模式下，SDK不再采集摄像头数据，仅保留编码和发送功能。
*
* @note 需要在 {@link V2TXLivePusher#startPush(String)} 之前调用 {@link V2TXLivePusher
#enableCustomVideoCapture(boolean)} 开启自定义采集。
*
* @param videoFrame 向 SDK 发送的 视频帧数据 {@link V2TXLiveVideoFrame}
*
* @return 返回值 {@link V2TXLiveCode}
* - V2TXLIVE_OK: 成功
* - V2TXLIVE_ERROR_INVALID_PARAMETER: 发送失败，视频帧数据不合法
```

```
* - V2TXLIVE_ERROR_REFUSED: 发送失败，您必须先调用 enableCustomVideoCapture 开启自定义视频采集
*/
public abstract int sendCustomVideoFrame(V2TXLiveVideoFrame videoFrame);
```

定制播放数据

iOS 平台

1. 设置 V2TXLivePlayer 的 V2TXLivePlayerObserver 监听。

```
@interface V2TXLivePlayer : NSObject
/**
* @brief 设置播放器回调。<br/>
* 通过设置回调，可以监听 V2TXLivePlayer 播放器的一些回调事件，
* 包括播放器状态、播放音量回调、音视频首帧回调、统计数据、警告和错误信息等。
*
* @param observer 播放器的回调目标对象，更多信息请查看 {@link V2TXLivePlayerObserver}
*/
- (void)setObserver:(id<V2TXLivePlayerObserver>)observer;
```

2. 通过 onRenderVideoFrame 回调捕获 Player 的图像数据。

```
/**
* @brief 视频帧信息。
* V2TXLiveVideoFrame 用来描述一帧视频画面的裸数据，它可以是一帧编码前的画面，也可以是一帧解码后的画面。
* @note 自定义采集和自定义渲染时使用。自定义采集时，需要使用 V2TXLiveVideoFrame 来包装待发送的视频帧；自定义渲染时，会返回经过 V2TXLiveVideoFrame 包装的视频帧。
*/
@interface V2TXLiveVideoFrame : NSObject

///**字段含义**视频帧像素格式
///**推荐取值**V2TXLivePixelFormatNV12
@property(nonatomic, assign) V2TXLivePixelFormat pixelFormat;

///**字段含义**视频数据包装格式
///**推荐取值**V2TXLiveBufferTypePixelBuffer
@property(nonatomic, assign) V2TXLiveBufferType bufferType;
```

```
///**字段含义**bufferType 为 V2TXLiveBufferTypeNSData 时的视频数据
@property(nonatomic, strong, nullable) NSData *data;

///**字段含义**bufferType 为 V2TXLiveBufferTypePixelBuffer 时的视频数据
@property(nonatomic, assign, nullable) CVPixelBufferRef pixelBuffer;

///**字段含义**视频宽度
@property(nonatomic, assign) NSUInteger width;

///**字段含义**视频高度
@property(nonatomic, assign) NSUInteger height;

///**字段含义**视频帧的顺时针旋转角度
@property(nonatomic, assign) V2TXLiveRotation rotation;

///**字段含义**视频纹理ID
@property (nonatomic, assign) GLuint textureId;

@end


@protocol V2TXLivePlayerObserver <NSObject>
@optional
/**
* @brief 自定义视频渲染回调
*
* @note 调用 [enableCustomRendering](@ref V2TXLivePlayer#enableCustomRendering:pixelFormat:bufferType:) 开启自定义渲染之后，会收到这个回调通知
*
* @param videoFrame 视频帧数据 {@link V2TXLiveVideoFrame}
*/
- (void)onRenderVideoFrame:(id<V2TXLivePlayer>)player
frame:(V2TXLiveVideoFrame *)videoFrame;
@end
```

Android 平台

1. 设置 V2TXLivePlayer 的 V2TXLivePlayerObserver 监听。

```
public abstract void setObserver(V2TXLivePlayerObserver observer)
```

2. 通过 onRenderVideoFrame 回调捕获 Player 的图像数据。

```
public final static class V2TXLiveVideoFrame
{
/// 视频像素格式
public V2TXLivePixelFormat pixelFormat = V2TXLivePixelFormat.V2TXLivePixelFormatUnkno
wn;
/// 视频数据包装格式
public V2TXLiveBufferType bufferType = V2TXLiveBufferType.V2TXLiveBufferTypeUnknown;
/// 视频纹理包装类
public V2TXLiveTexture texture;
/// 视频数据
public byte[] data;
/// 视频数据
public ByteBuffer buffer;
/// 视频宽度
public int width;
/// 视频高度
public int height;
/// 视频像素的顺时针旋转角度
public int rotation;
}

public abstract class V2TXLivePlayerObserver {
/**
* 自定义视频渲染回调
*
* @param player 回调该通知的播放器对象
* @param videoFrame 视频帧数据 {@link V2TXLiveVideoFrame}
*/
void onRenderVideoFrame(V2TXLivePlayer player, V2TXLiveVideoFrame videoFrame);
}
```

禁播和流管理

最近更新时间：2021-08-23 11:43:42

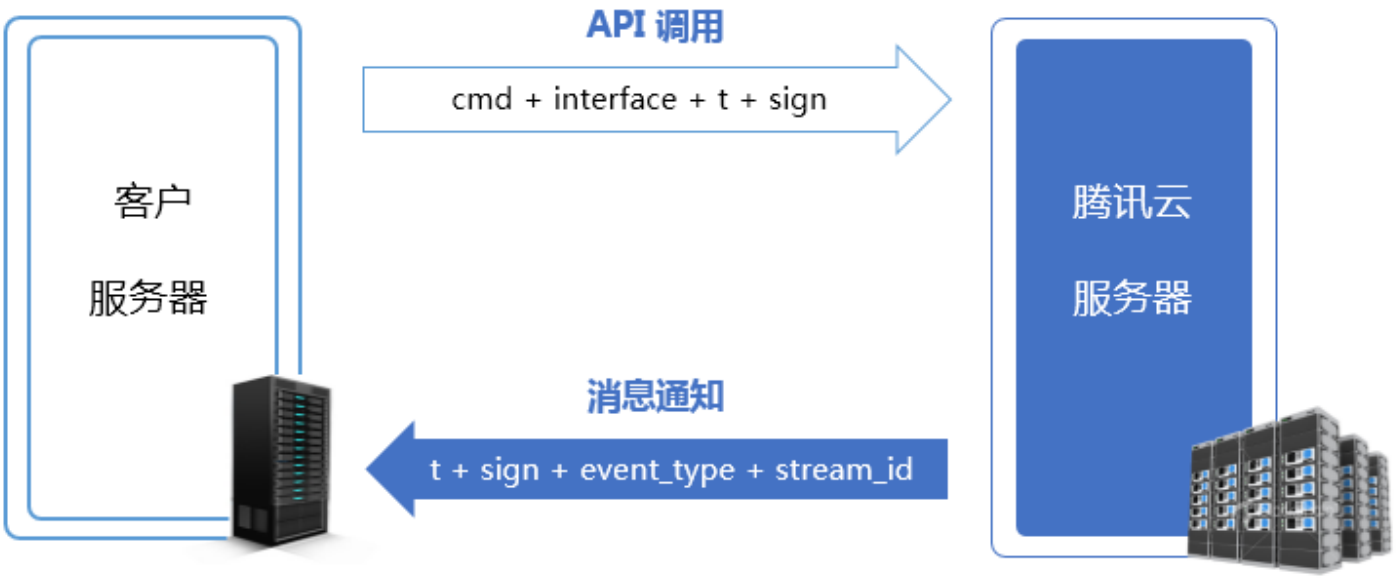
版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

与腾讯云后台通讯



您的服务器与腾讯云服务器的信息同步可以通过两种方式组合实现：

- **API 调用**：腾讯云提供了直播相关的 API 接口，包括状态查询和状态管理等功能，供您的后台服务器调用。
- **消息通知**：腾讯云在直播流状态变更、录制文件生成等一系列事件发生时，能够以事件消息（JSON）的形式主动通知您的后台服务器，只需要您在腾讯云注册接收事件通知的回调 URL 即可实现。

API 调用

腾讯云提供了直播相关的 API 接口，包括状态查询和状态管理等功能，供您的后台服务器调用，详情请参见 云直播 API 概览 文档。

调用方法

在您的**服务端**采用 HTTP 协议的 GET 请求方式（即调用参数直接拼接在 URL 中）进行调用即可，详细的调用方法在每个 API 的说明文档中都有示例参考。

安全机制

由于对 API 的调用采用的是普通的 HTTP 协议（出于性能考虑），这就需要一套行之有效的办法来确保您的服务器与腾讯云后台之间的通讯安全。

所有直播码相关的云端 API 都采用了同一种安全检查机制，**t + sign 校验**：

- **t（过期时间）**：如果一个 API 请求或者通知中的 t 值所规定的时间已经过期，则可以判定这个请求或者通知为无效的，这样做可以防止网络重放攻击。t 的格式为 UNIX 时间戳，即从1970年01月01日（UTC/GMT 的午夜）开始所经过的秒数。
- **sign（安全签名）**：sign = MD5(key + t)，即把加密 key 和 t 进行字符串拼接后，计算一下 md5 值。这里的 key 即 CGI 调用 key，您在腾讯云直播管理控制台 域名管理 中可以进行设置，以推流配置为例步骤如下：
  i. 进入 域名管理 单击推流域名后面的 **管理** 。

  ii. 选择 **推流配置**，查看 **鉴权配置** 标签，单击 **编辑** 进行配置。

> 说明：
> 若您需要对**播放域名**进行鉴权配置，您可以在 域名管理 选择播放域名或单击后面的 **管理**，进入播放域名详情页，选择**访问控制**，查看 **鉴权配置** 标签，单击 **编辑**，进行配置。

- **安全原理**
  由于 MD5 是不可逆的 HASH 算法，所以只要确保 KEY 不泄露，即使攻击者拿到很多对 t 和 sign 也无法反算出 KEY 值，进而无法进行伪装攻击。

- **计算示例**
  例如，我们现在的时间是2021-07-21 11:46:00，我们希望有效期是1分钟，也就是2021-07-21 11:47:00以后再收到携带这个 t 的请求或者通知即判定为非法的：

```
t = "2021-07-21 11:47:00" = 1626839220
```

  假设我们的 key 是 **5d41402abc4b2a76b9719d911017c592**，那么我们计算的签名结果就是：

```
sign = MD5(5d41402abc4b2a76b9719d911017c5921626839220 ) = 5ee8ca6c28cbe415b403
52969cdf8249
```

错误码

HTTP 错误

错误码	含义	备注
403	Forbidden	接口为了安全考虑开启了校验，若使用浏览器验证发现该错误，可检查下 cookie 里是否含有 skey。
404	Not Found	查看请求时是否带上 host。

接口通用返回错误

错误信息	含义
appid is invalid	appid 不合法，表示未开通该功能。

接口前端接入返回错误信息

错误信息	含义

错误信息	含义
cmd is invalid	cmd 不合法，表示未开通该功能。
sign invalid	鉴权计算错误，请参见 安全机制。
time expired	鉴权成功，但是超过了 URL 有效期，请参见 安全机制。

接口后端查询返回错误码

错误码	错误信息	含义
0	query data successfully	本次查询成功，并返回结果数据。
1000	user is not registered for statapi	用户没有注册 statapi，请提工单到后台开通。
1001	user service for statapi was stopped	用户 statapi 访问服务已经被终止。
1201	internel/system error	内部系统错误，属于系统异常，建议通过 工单 反馈到服务商。
1202	invalid request/request frequency exceeds limit	无效的请求，一般是超过了频控次数，如果频率不能满足业务需要，可申请增加次数。
1204	invalid input param	输入参数错误，请检查下输入的参数是否符合接口规范。
1301	has not live stream	没有活跃的流，在调用实时接口时会返回改错误码。
10003	query data is empty	后端查询数据成功，但是返回数据为空。例如，某时间段没有播放，此时调用接口 Get_LivePlayStatHistory 就会返回10003。

> ⚠ 注意：
> 以上错误码针对本文 API 列表中的 API，不包括 消息事件通知。

消息通知

详情请参见腾讯云事件 消息通知 服务。

变声和混响
iOS

最近更新时间：2021-08-12 19:54:49

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	–	✓	✓	✓	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

录制变声混响

```
//获取 recorder 对象
recorder = [TXUGCRecord shareInstance];

// 设置混响
// TXRecordCommon.VIDOE_REVERB_TYPE_0 关闭混响
// TXRecordCommon.VIDOE_REVERB_TYPE_1 KTV
// TXRecordCommon.VIDOE_REVERB_TYPE_2 小房间
// TXRecordCommon.VIDOE_REVERB_TYPE_3 大会堂
// TXRecordCommon.VIDOE_REVERB_TYPE_4 低沉
// TXRecordCommon.VIDOE_REVERB_TYPE_5 洪亮
// TXRecordCommon.VIDOE_REVERB_TYPE_6 金属声
// TXRecordCommon.VIDOE_REVERB_TYPE_7 磁性
[recorder setReverbType:VIDOE_REVERB_TYPE_1];

// 设置变声
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_0 关闭变声
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_1 熊孩子
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_2 萝莉
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_3 大叔
```

版权所有：腾讯云计算（北京）有限责任公司

```
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_4 重金属
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_6 外国人
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_7 困兽
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_8 死肥仔
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_9 强电流
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_10 重机械
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_11 空灵
[record setVoiceChangerType:VIDOE_VOICECHANGER_TYPE_1];
```

说明：

变声混响只针对录制人声有效，针对 BGM 无效。

Android

最近更新时间：2021-08-12 19:54:54

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	–	✓	✓	✓	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

录制变声混响

```
// 设置混响
// TXRecordCommon.VIDOE_REVERB_TYPE_0 关闭混响
// TXRecordCommon.VIDOE_REVERB_TYPE_1 KTV
// TXRecordCommon.VIDOE_REVERB_TYPE_2 小房间
// TXRecordCommon.VIDOE_REVERB_TYPE_3 大会堂
// TXRecordCommon.VIDOE_REVERB_TYPE_4 低沉
// TXRecordCommon.VIDOE_REVERB_TYPE_5 洪亮
// TXRecordCommon.VIDOE_REVERB_TYPE_6 金属声
// TXRecordCommon.VIDOE_REVERB_TYPE_7 磁性
mTXCameraRecord.setReverb(TXRecordCommon.VIDOE_REVERB_TYPE_1);

// 设置变声
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_0 关闭变声
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_1 熊孩子
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_2 萝莉
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_3 大叔
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_4 重金属
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_6 外国人
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_7 困兽
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_8 死肥仔
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_9 强电流
```

版权所有：腾讯云计算（北京）有限责任公司

```
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_10 重机械
// TXRecordCommon.VIDOE_VOICECHANGER_TYPE_11 空灵
mTXCameraRecord.setVoiceChangerType(TXRecordCommon.VIDOE_VOICECHANGER_TYPE_1);
```

说明：

变声混响只针对录制人声有效，针对 BGM 无效。

设定画面质量

最近更新时间：2022-03-04 11:06:16

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

示例代码

针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

功能介绍

LiteAVSDK 通过 V2TXLivePusher 提供的 setVideoQuality 接口来设定画面质量：

接口定义

可以通过 setVideoQuality 设置推流视频分辨率，以及宽高比模式（横屏 / 竖屏）。

```
int setVideoQuality(V2TXLiveVideoResolution resolution, V2TXLiveVideoResolutionMode resolutionMode);
```

参数

参数	类型	含义

参数	类型	含义
resolution	V2TXLiveVideoResolution	视频分辨率
resolutionMode	V2TXLiveVideoResolutionMode	视频宽高比模式（横屏或者竖屏模式）

- *V2TXLiveVideoResolution 枚举值：*

取值	含义
V2TXLiveVideoResolution160x160	分辨率 160*160，码率范围：100Kbps ~ 150Kbps，帧率: 15fps
V2TXLiveVideoResolution270x270	分辨率 270*270，码率范围：200Kbps ~ 300Kbps，帧率：15fps
V2TXLiveVideoResolution480x480	分辨率 480*480，码率范围：350Kbps ~ 525Kbps，帧率：15fps
V2TXLiveVideoResolution320x240	分辨率 320*240，码率范围：250Kbps ~ 375Kbps，帧率：15fps。
V2TXLiveVideoResolution480x360	分辨率 480*360，码率范围：400Kbps ~ 600Kbps，帧率：15fps
V2TXLiveVideoResolution640x480	分辨率 640*480，码率范围：600Kbps ~ 900Kbps，帧率：15fps
V2TXLiveVideoResolution320x180	分辨率 320*180，码率范围：250Kbps ~ 400Kbps，帧率：15fps
V2TXLiveVideoResolution480x270	分辨率 480*270，码率范围：350Kbps ~ 550Kbps，帧率：15fps
V2TXLiveVideoResolution640x360	分辨率 640*360，码率范围：500Kbps ~ 900Kbps，帧率：15fps
V2TXLiveVideoResolution960x540	分辨率 960*540，码率范围：800Kbps ~ 1500Kbps，帧率：15fps
V2TXLiveVideoResolution1280x720	分辨率 1280*720，码率范围：1000Kbps ~ 1800Kbps，帧率：15fps
V2TXLiveVideoResolution1920x1080	分辨率 1920*1080，码率范围：2500Kbps ~ 3000Kbps，帧率：15fps

- *V2TXLiveVideoResolutionMode 枚举值：*

取值	含义
V2TXLiveVideoResolutionModeLandscape	横屏模式下的分辨率：V2TXLiveVideoResolution640_360 + V2TXLiveVideoResolutionModeLandscape = 640x360
V2TXLiveVideoResolutionModePortrait	竖屏模式下的分辨率：V2TXLiveVideoResolution640_360 + V2TXLiveVideoResolutionModePortrait = 360x640

参数设定建议

应用场景	resolution	resolutionMode
秀场直播	• V2TXLiveVideoResolution960x540 • V2TXLiveVideoResolution1280x720	横屏或者竖屏
手游直播	V2TXLiveVideoResolution1280x720	横屏或者竖屏
连麦（主画面）	V2TXLiveVideoResolution640x360	横屏或者竖屏
连麦（小画面）	V2TXLiveVideoResolution480x360	横屏或者竖屏
蓝光直播	V2TXLiveVideoResolution1920x1080	横屏或者竖屏

注意事项

为了连麦更流畅，进入连麦状态后请调用 setVideoQuality() 将 quality 挡位设置为 V2TXLiveVideoResolution640x360（主播）或 V2TXLiveVideoResolution480x360（连麦观众），结束连麦状态后可以调用 setVideoQuality() 将 quality 挡位恢复为连麦前的值。

常见问题

1. 为什么观众端看到的画面没有主播端清晰?

主播端看到的画面，是从摄像头采集的原始画面，经过前处理（美颜、镜像、裁剪等操作）后直接渲染给主播观看，所以清晰度是最高的。而观众端看到的是经过编码器压缩再解码的画面，由于编码本身会降低压缩质量（视频码率设置的越低，压缩程度越严重），所以观众端看到的画面会比主播端清晰度低。

2. 为什么 V2TXLivePusher 推出来的流会有368 × 640或者544 × 960这样的分辨率?

在开启硬件加速后，您可能会发现诸如368 × 640或者544 × 960此类“不完美”分辨率，这是由于部分硬编码器要求像素能被16整除所致，属于正常现象，您可以通过设置播放端的填充模式解决“小黑边”问题。

版权所有：腾讯云计算（北京）有限责任公司

SDK 指标监控

最近更新时间：2021-12-23 17:01:14

版本支持

本页文档所描述功能，在腾讯云视立方中支持情况如下：

版本名称	基础直播 Smart	互动直播 Live	短视频 UGSV	音视频通话 TRTC	播放器 Player	全功能
支持情况	✓	✓	–	–	–	✓
SDK 下载	下载	下载	下载	下载	下载	下载

不同版本 SDK 包含的更多能力，具体请参见 SDK 下载。

概述

SDK 的各项监控指标可以从 V2TXLivePusherObserver 和 V2TXLivePlayerObserver 的回调中获取。

V2TXLivePusherObserver

获取推流的状态数据

V2TXLivePusherObserver 的 onStatisticsUpdate 回调，会每隔2秒会将 SDK 内部的状态指标同步出来，其中如下指标比较有意义：

推流状态	含义说明
appCpu	当前进程的 CPU 使用率。
systemCpu	本机总体的 CPU 使用率。
width	当前视频的宽度，单位：像素值。
height	当前视频的高度，单位：像素值。
fps	当前视频帧率，也就是视频编码器每条生产了多少帧画面。
videoBitrate	当前视频编码器输出的比特率，也就是编码器每秒生产了多少视频数据，单位：kbps。
audioBitrate	当前音频编码器输出的比特率，也就是编码器每秒生产了多少音频数据，单位：kbps。

有参考价值的状态指标

状态指标	说明
systemCpu	• 如果**系统 CPU 使用率**超过80%，音视频编码的稳定性会受到影响，可能导致画面和声音的随机卡顿。 • 如果**系统 CPU 使用率**经常100%，会导致视频编码帧率不足，音频编码跟不上，必然导致画面和声音的严重卡顿。
fps	常来说每秒15帧以上的视频流才能保证观看的流畅度，常规推流如果 FPS 在10帧以下，观众就会明显的感到画面卡顿。

> 说明：
> 很多客户会遇到的一个问题： App 在线下测试时性能表现极佳，但在 App 外发上线后，前排房间里的互动消息的滚屏和刷新会产生极大的 CPU 消耗导致直播画面卡顿严重。

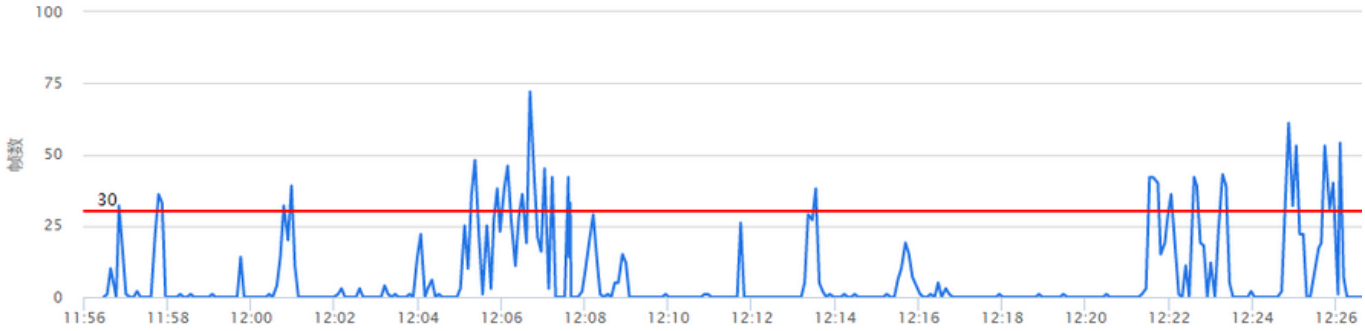
看懂腾讯云推流图表

在 **云直播控制台** > **质量监控** 您可以看到您所属账户里的直播间情况，以及每个直播间的推流质量数据：

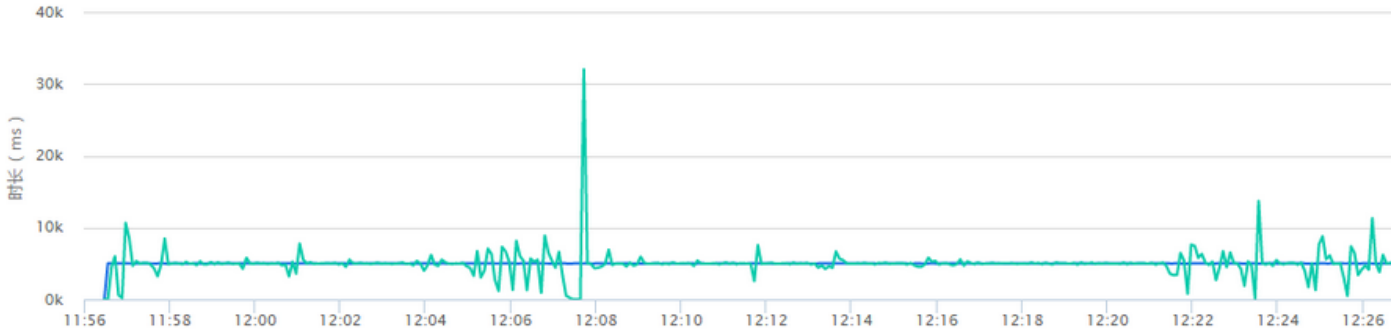
- **主播端-应发速率-实发速率曲线图**
  蓝色曲线代表 BITRATE 的统计曲线，即 SDK 产生的音视频数据，绿色曲线代表实际网络发出去多少。两条线重合度越高表示推流质量越好。

- **主播端-音视频数据堆积情况**
  - 如果曲线始终贴着0刻度线走，说明整个推流过程非常顺畅，一点都没有堆积。
  - 如果出现大于0的部分，说明当时有网络波动导致数据积压，有可能在播放端产生轻微卡顿和音画不同步的现象。

- 如果堆积超出红色警戒线，说明已经产生了丢包，必然会在播放端产生卡顿和音画不同步的现象。



- **云端-应收视频时长-实收视频时长曲线**
  这里是腾讯云服务端的统计图表，如果您不是使用腾讯云 SDK 推流，那么您将只能看到这个图表，前面两个（数据源来自 SDK）是看不到的。蓝绿两条线重合度越高，说明推流质量越好。



V2TXLivePlayerObserver

获取播放的状态数据

V2TXLivePlayerObserver 的 onStatisticsUpdate 回调，会每隔2秒会将 SDK 内部的状态指标同步出来，其中如下指标比较有意义：

播放状态	含义说明
appCpu	当前进程的 CPU 使用率。
systemCpu	本机总体的 CPU 使用率。
width	当前视频的宽度，单位：像素值。
height	当前视频的高度，单位：像素值。
fps	当前流媒体的视频帧率。
videoBitrate	当前流媒体的视频码率，单位：kbps。
audioBitrate	当前流媒体的音频码率，单位：kbps。

有参考价值的状态指标

状态指标	说明
systemCpu	• 如果**系统 CPU 使用率**超过80%，音视频编码的稳定性会受到影响，可能导致画面和声音的随机卡顿。 • 如果**系统 CPU 使用率**经常100%，会导致视频编码帧率不足，音频编码跟不上，必然导致画面和声音的严重卡顿。
fps	常来说每秒15帧以上的视频流才能保证观看的流畅度，常规推流如果 FPS 在10帧以下，观众就会明显的感到画面卡顿。