直播 SDK

高级功能

产品文档

腾讯云

文档目录

高级功能
设定画面质量

最近更新时间：2024-01-13 15:53:49

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。

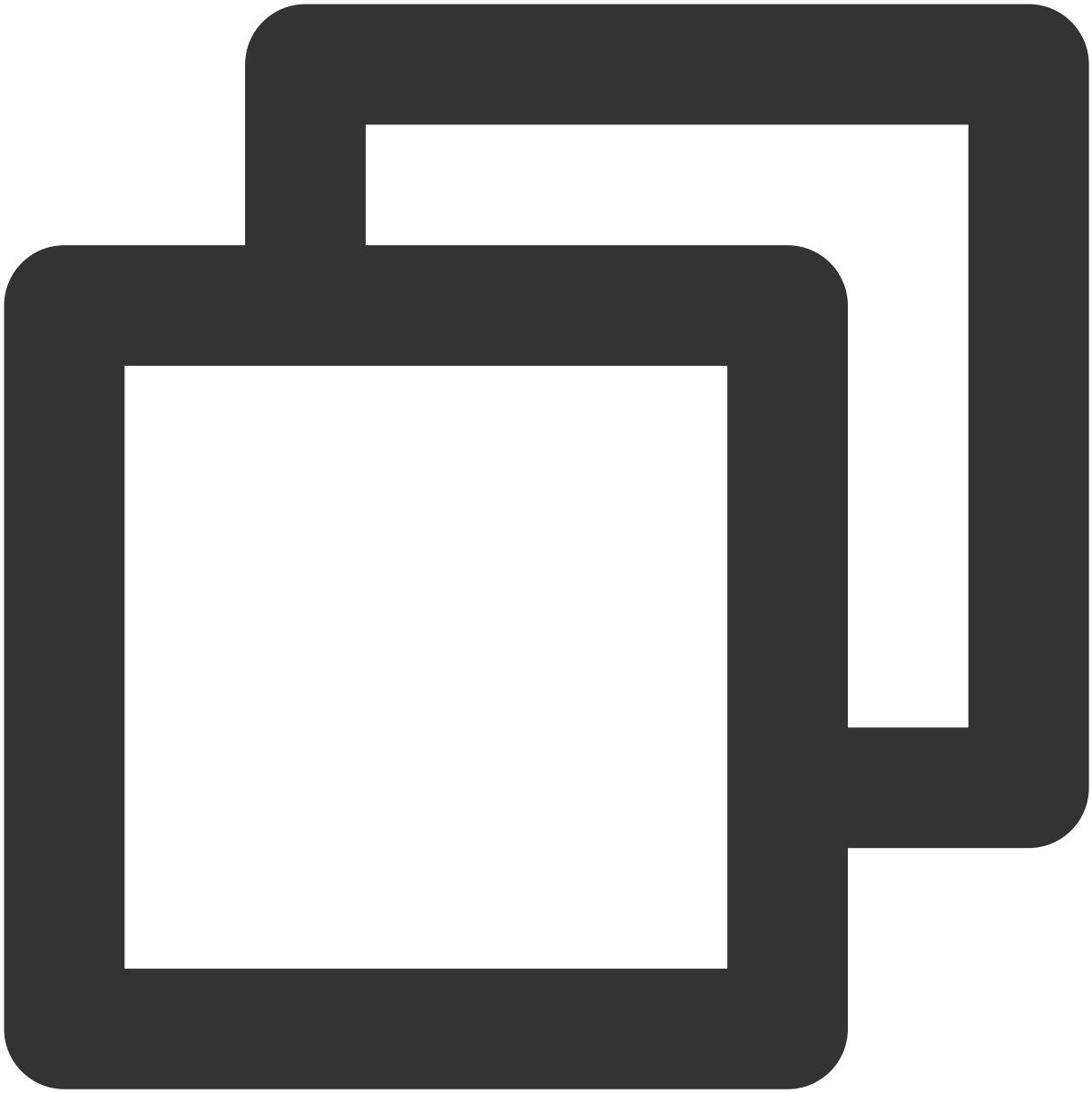
所属平台	GitHub 地址
iOS	Github
Android	Github

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）。



```
public abstract int setVideoQuality(V2TXLiveVideoEncoderParam param);
```

参数

参数	类型	含义
param	V2TXLiveVideoEncoderParam	视频编码参数。

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。

自定义采集和渲染

最近更新时间：2024-01-13 15:53:49

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

定制推流画面

iOS 平台

方案一：修改 OpenGL 纹理

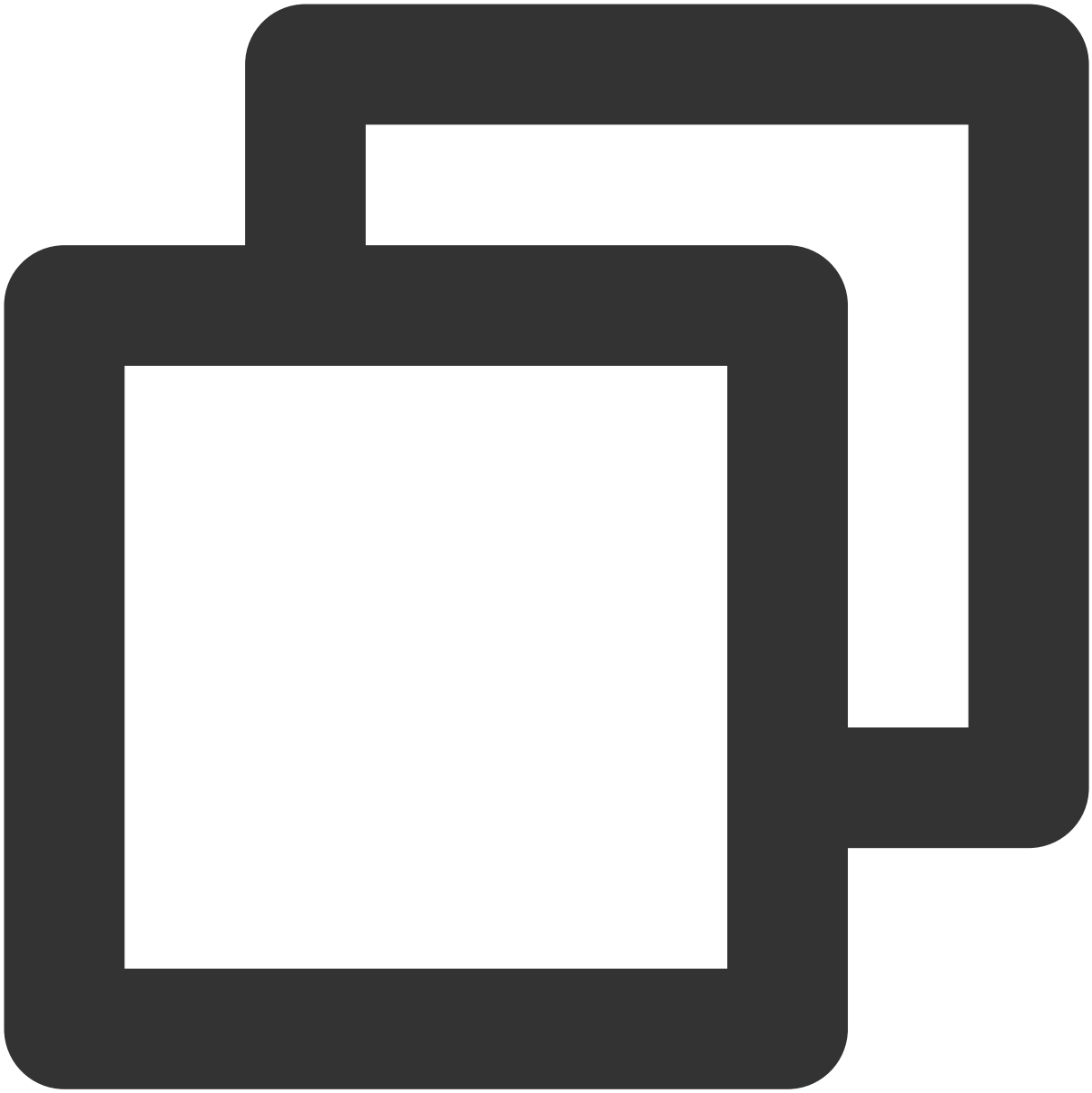
方案二：自己采集数据

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：

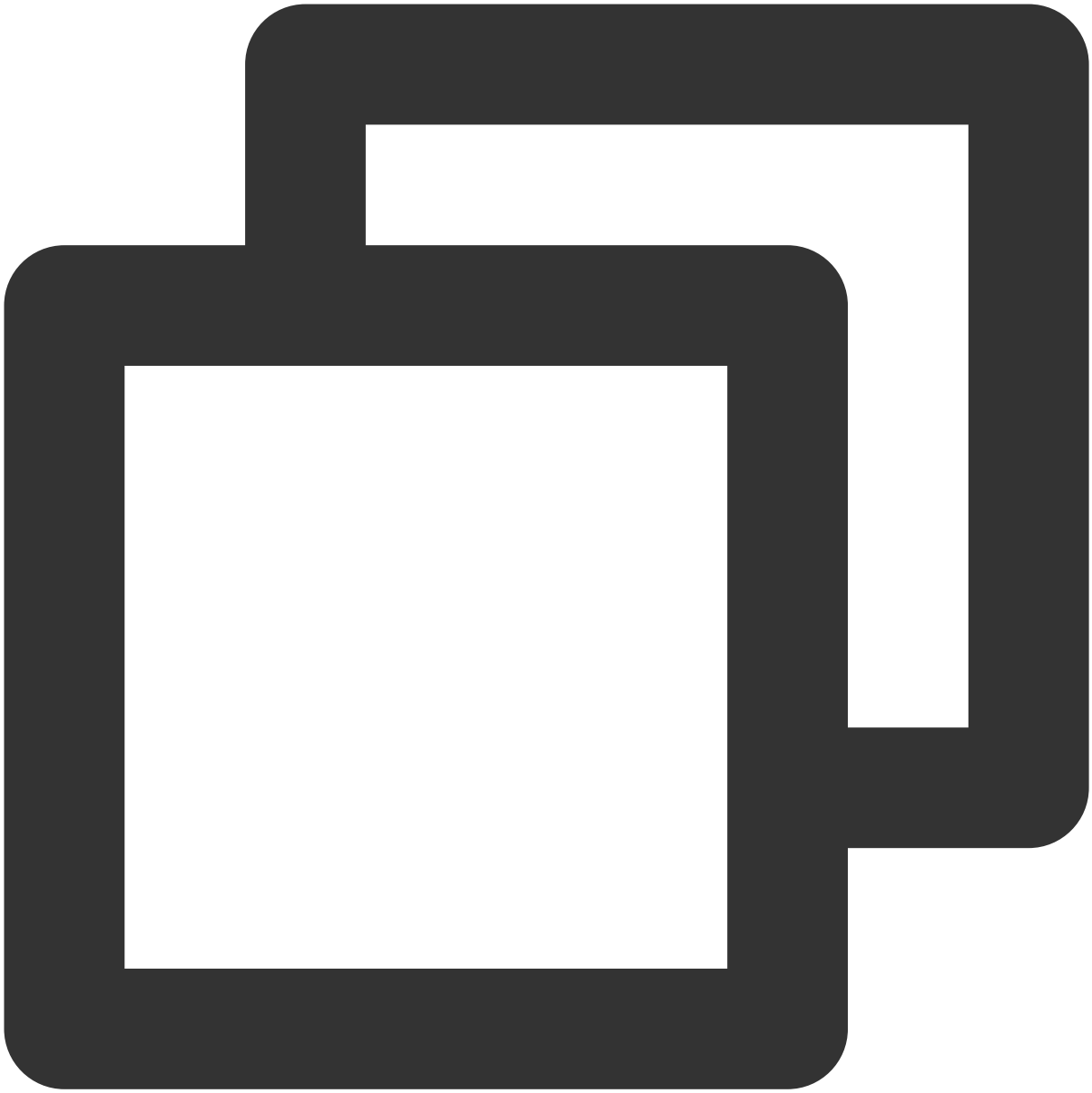
如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函数中将 `dstFrame.textureId` 设置为新纹理的 ID。



```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:(V2TX
{
    GLuint dstTextureId = renderItemWithTexture(srcFrame.textureId, srcFrame.widt
    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：

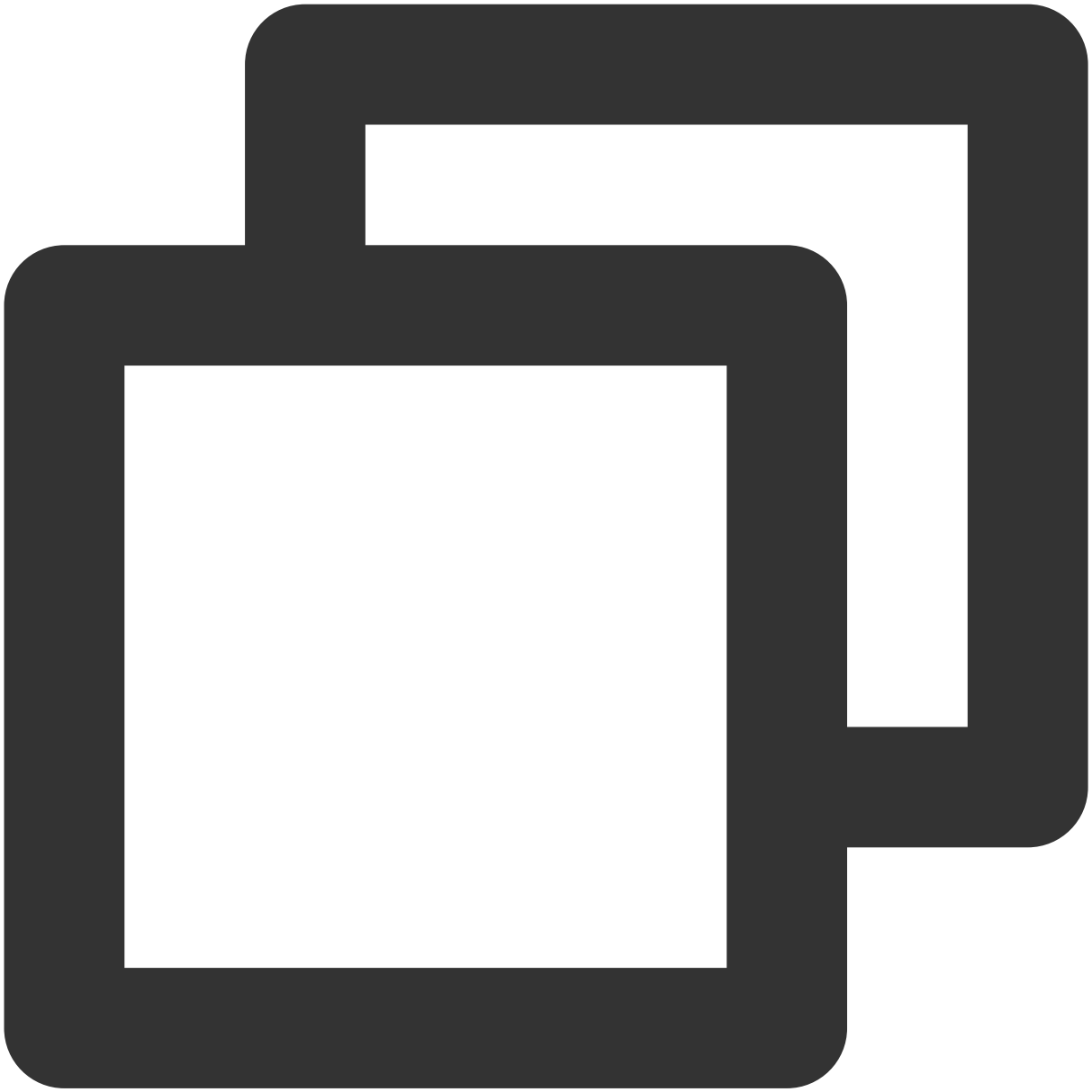


```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:(V2TXLi
    {
        thirdparty_process(srcFrame.textureId, srcFrame.width, srcFrame.height, dstF
    }
```

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。

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
 *
 * @note  需要在 [startPush](@ref V2TXLivePusher#startPush:) 之前调用 [enableCustomVi
 *
```

```
 * @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
 */
- (V2TXLiveCode)sendCustomVideoFrame:(V2TXLiveVideoFrame *)videoFrame;
```

Android 平台

方案一：修改 OpenGL 纹理

方案二：自己采集数据

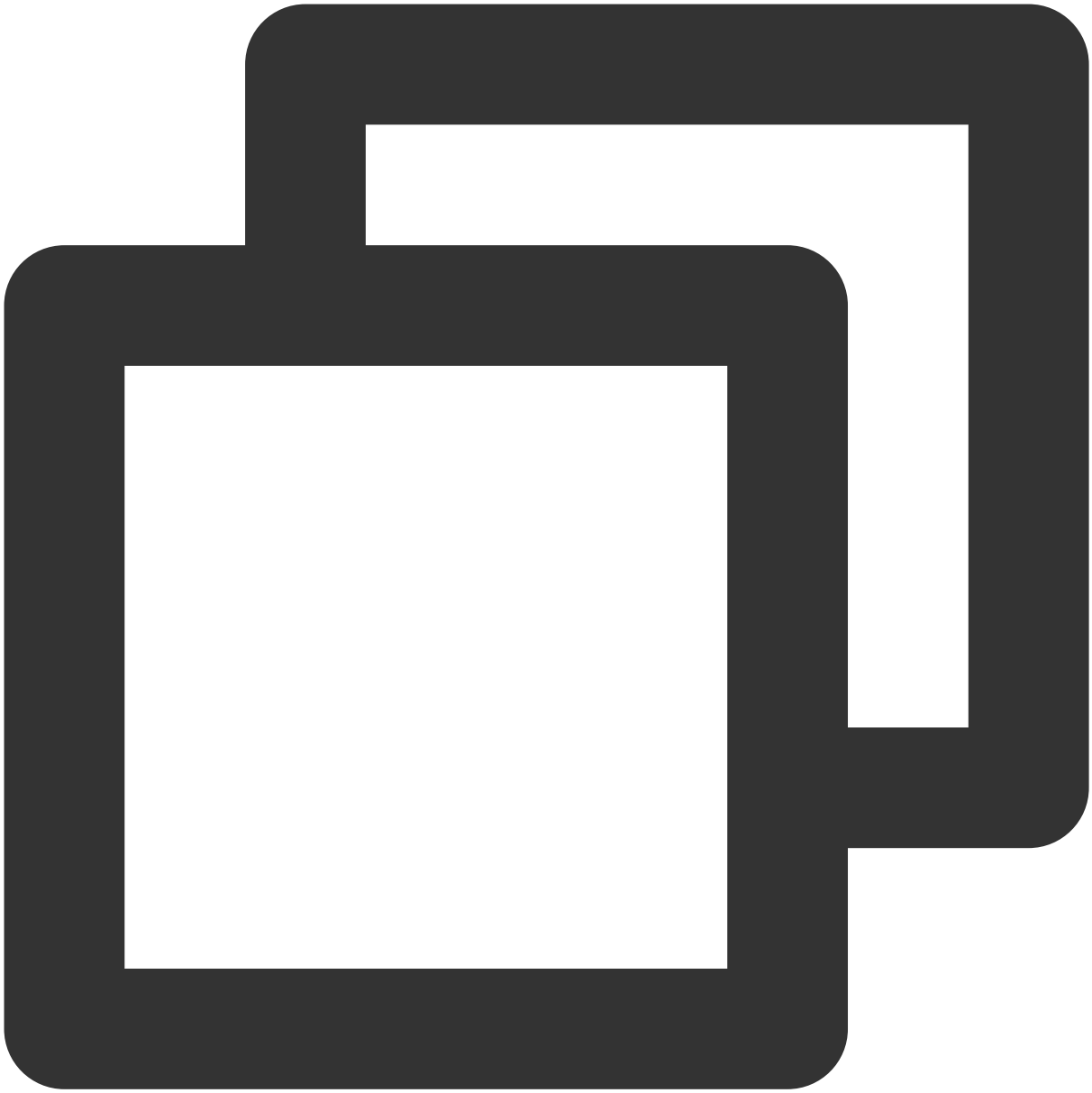
如果您有自定义图像处理的需求（例如堆加字幕），同时又希望复用 LiteAV SDK 的整体流程，您可以按照如下攻略进行定制。

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。

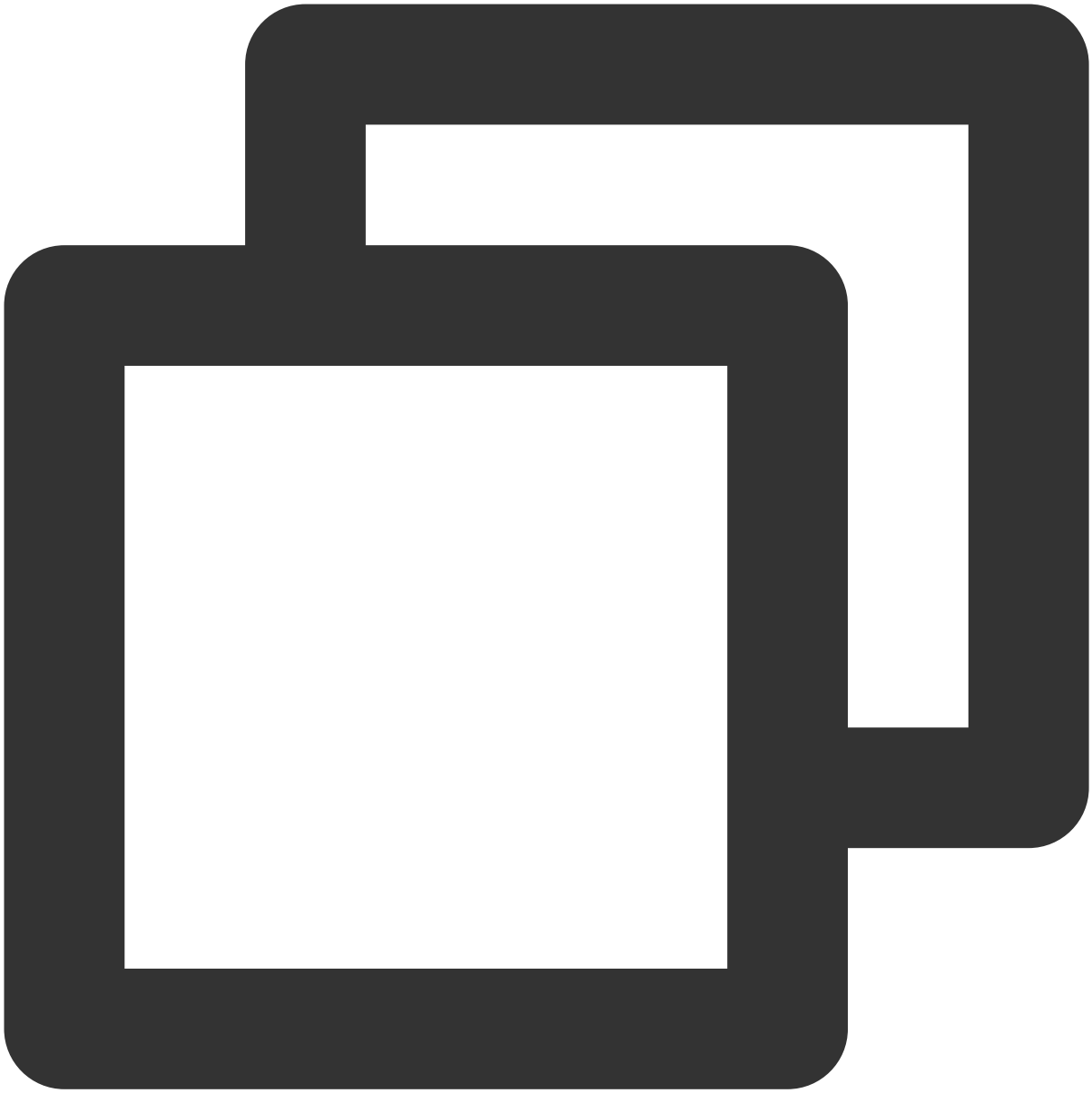


```
private class MyPusherObserver extends V2TXLivePusherObserver {
 @Override
 public void onGLContextCreated() {
 mFURenderer.onSurfaceCreated();
 mFURenderer.setUseTexAsync(true);
 }

 @Override
 public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFrame dst
 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：

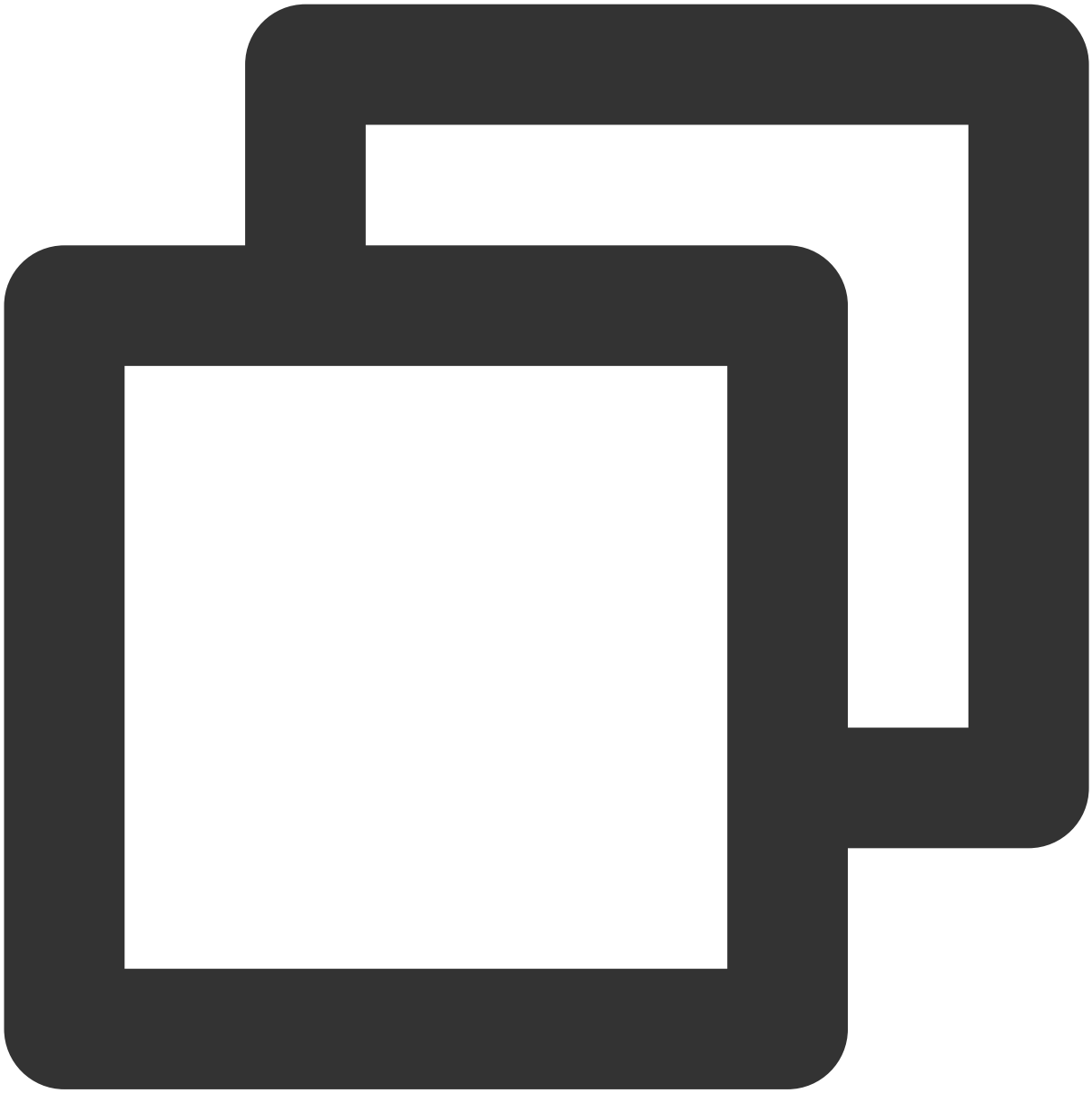


```
@Override
public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFrame dst
thirdparty_process(srcFrame.texture.textureId, srcFrame.width, srcFrame.height, ds
return 0;
}
```

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。

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。



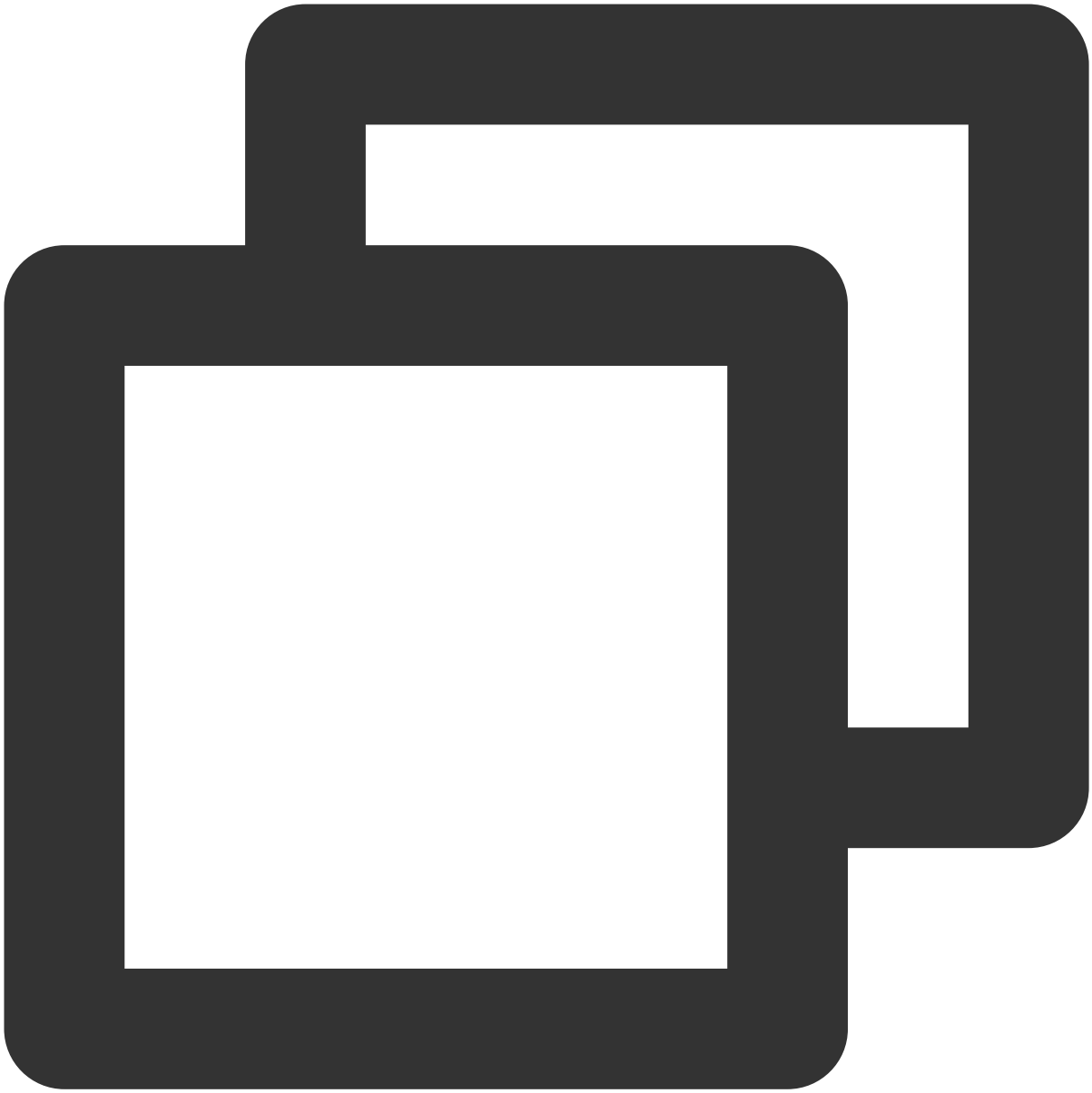
```
/**
 * @brief 在自定义视频采集模式下，将采集的视频数据发送到SDK。 <br/>
 *        在自定义视频采集模式下，SDK不再采集摄像头数据，仅保留编码和发送功能。
 *
 * @note  需要在 {@link V2TXLivePusher#startPush(String)} 之前调用 {@link V2TXLivePusl
 *
 * @param videoFrame 向 SDK 发送的 视频帧数据 {@link V2TXLiveVideoFrame}
```

```
     *
     * @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
     */
    public abstract int sendCustomVideoFrame(V2TXLiveVideoFrame videoFrame);
```

定制播放数据

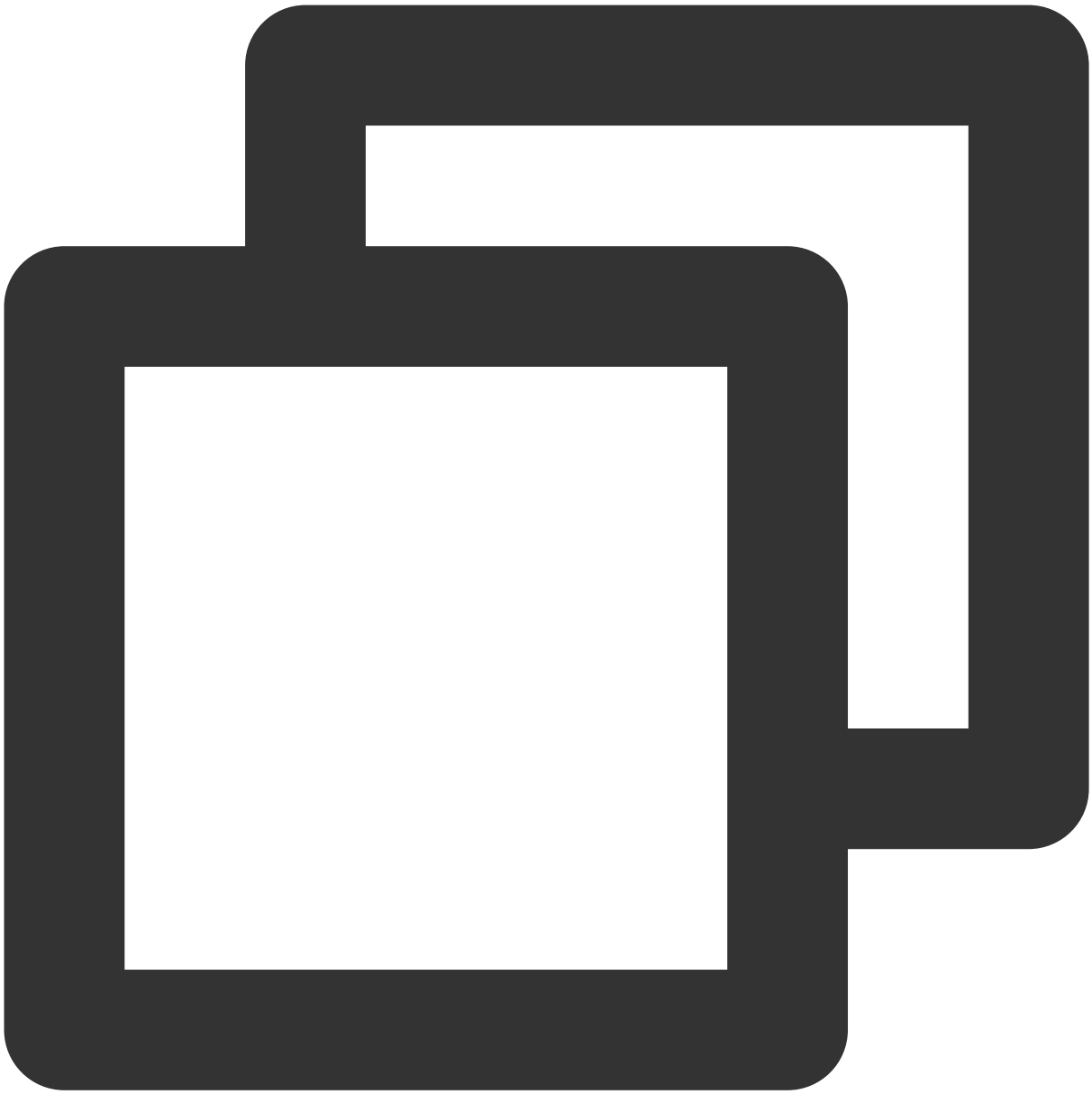
iOS 平台

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。

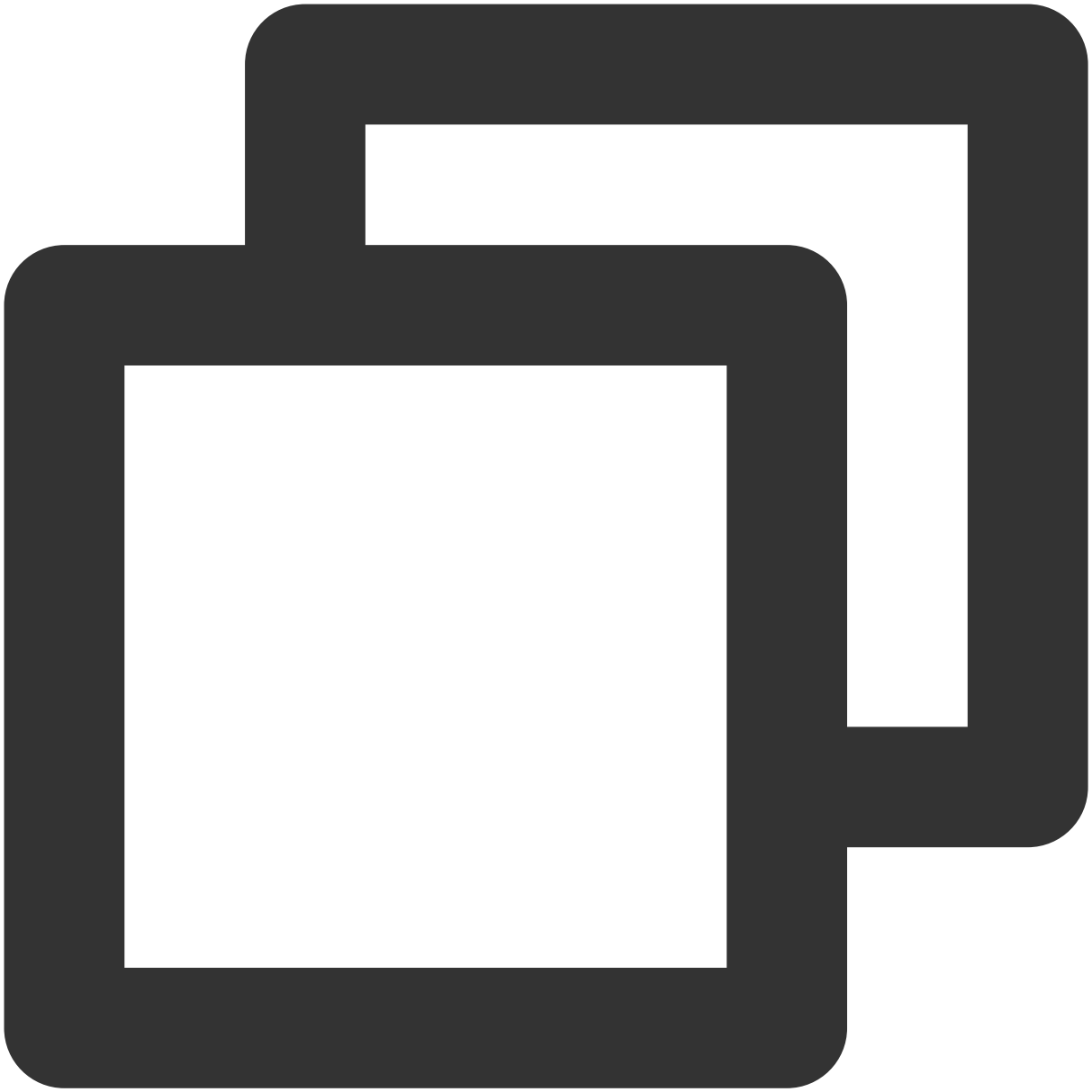


```
/**
 * @brief 视频帧信息。
 *        V2TXLiveVideoFrame 用来描述一帧视频画面的裸数据，它可以是一帧编码前的画面，也可以是-
 * @note  自定义采集和自定义渲染时使用。自定义采集时，需要使用 V2TXLiveVideoFrame 来包装待发送
 */
@interface V2TXLiveVideoFrame : NSObject

///【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:pix
 *
 * @param videoFrame  视频帧数据 {@link V2TXLiveVideoFrame}
 */
- (void)onRenderVideoFrame:(id<V2TXLivePlayer>)player
                     frame:(V2TXLiveVideoFrame *)videoFrame;
                 @end
```

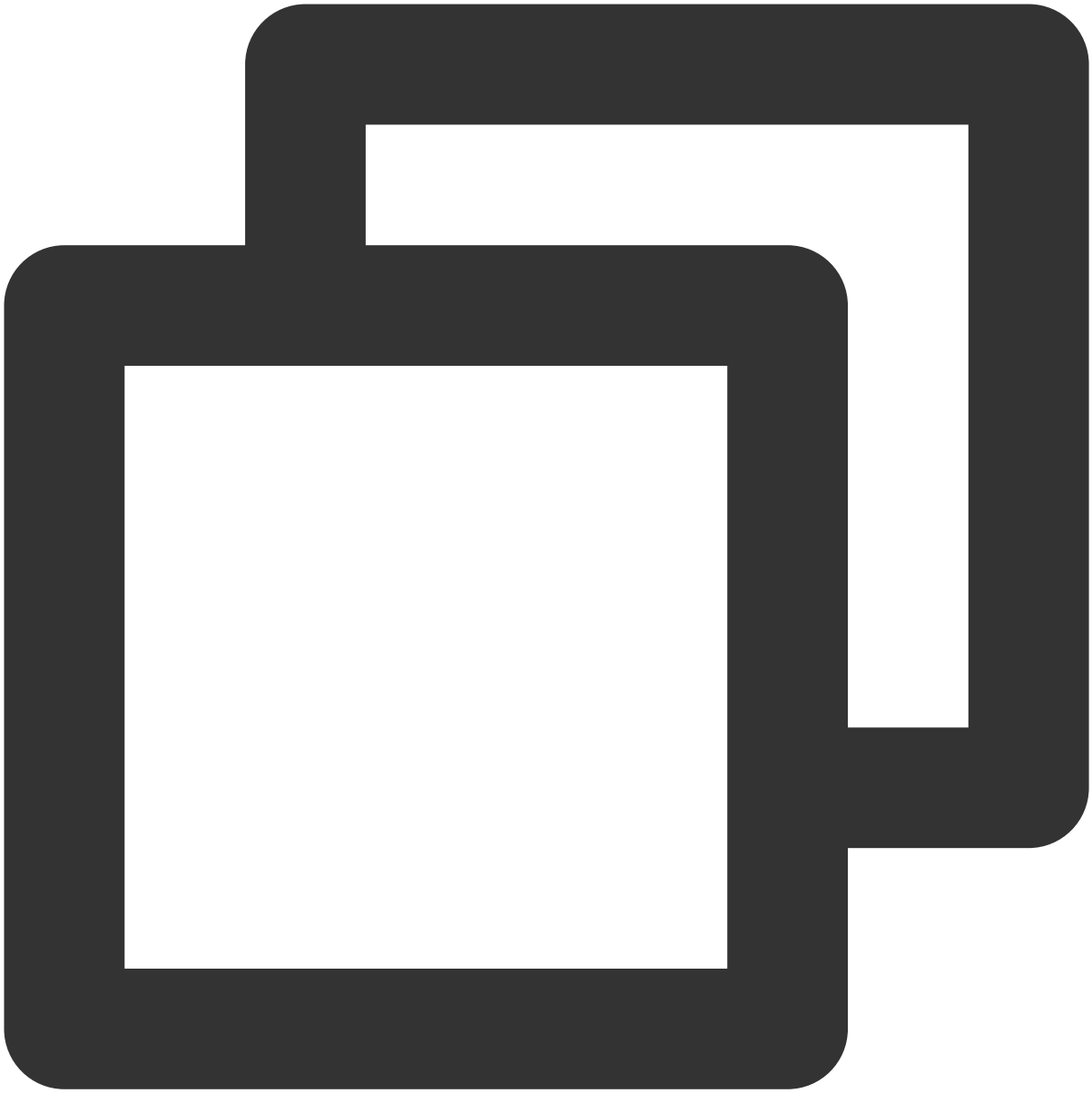
Android 平台

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。



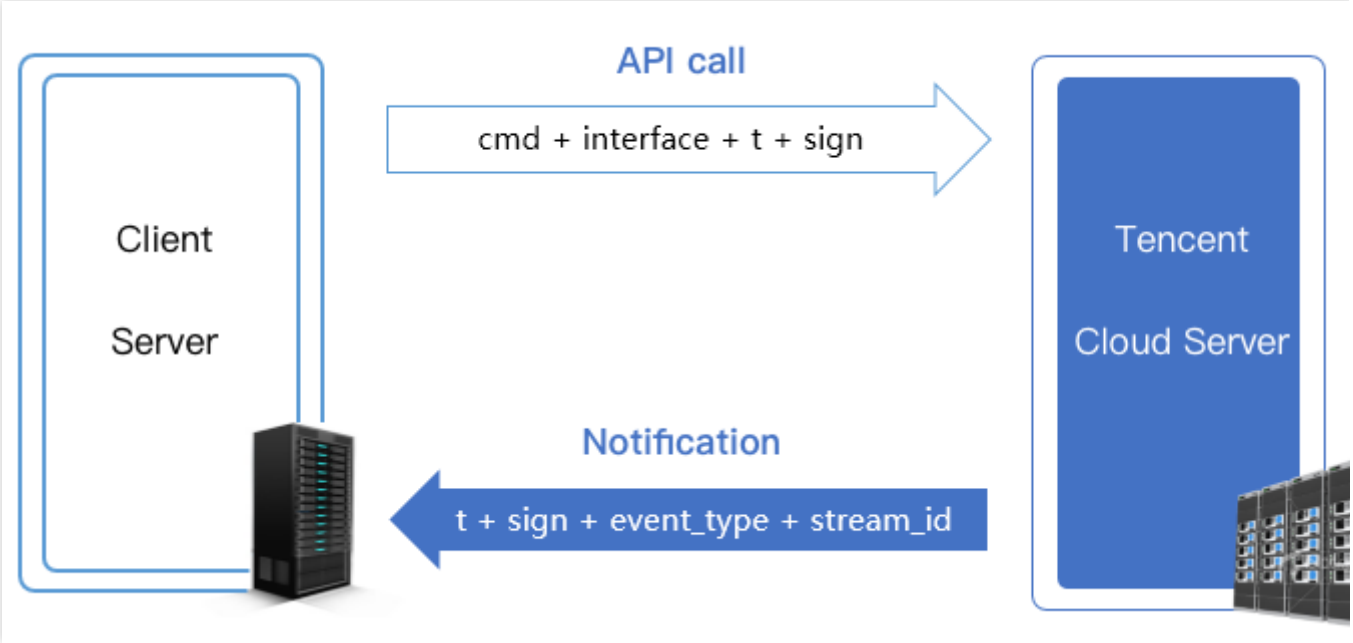
```
public final static class V2TXLiveVideoFrame
{
    /// 视频像素格式
    public V2TXLivePixelFormat pixelFormat = V2TXLivePixelFormat.V2TXLivePixelForma
    /// 视频数据包装格式
    public V2TXLiveBufferType bufferType   = V2TXLiveBufferType.V2TXLiveBufferTypeU
    /// 视频纹理包装类
    public V2TXLiveTexture texture;
    /// 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：2024-01-13 15:53:49

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：

1.1 进入 域名管理 单击推流域名后面的**管理**。

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

说明：

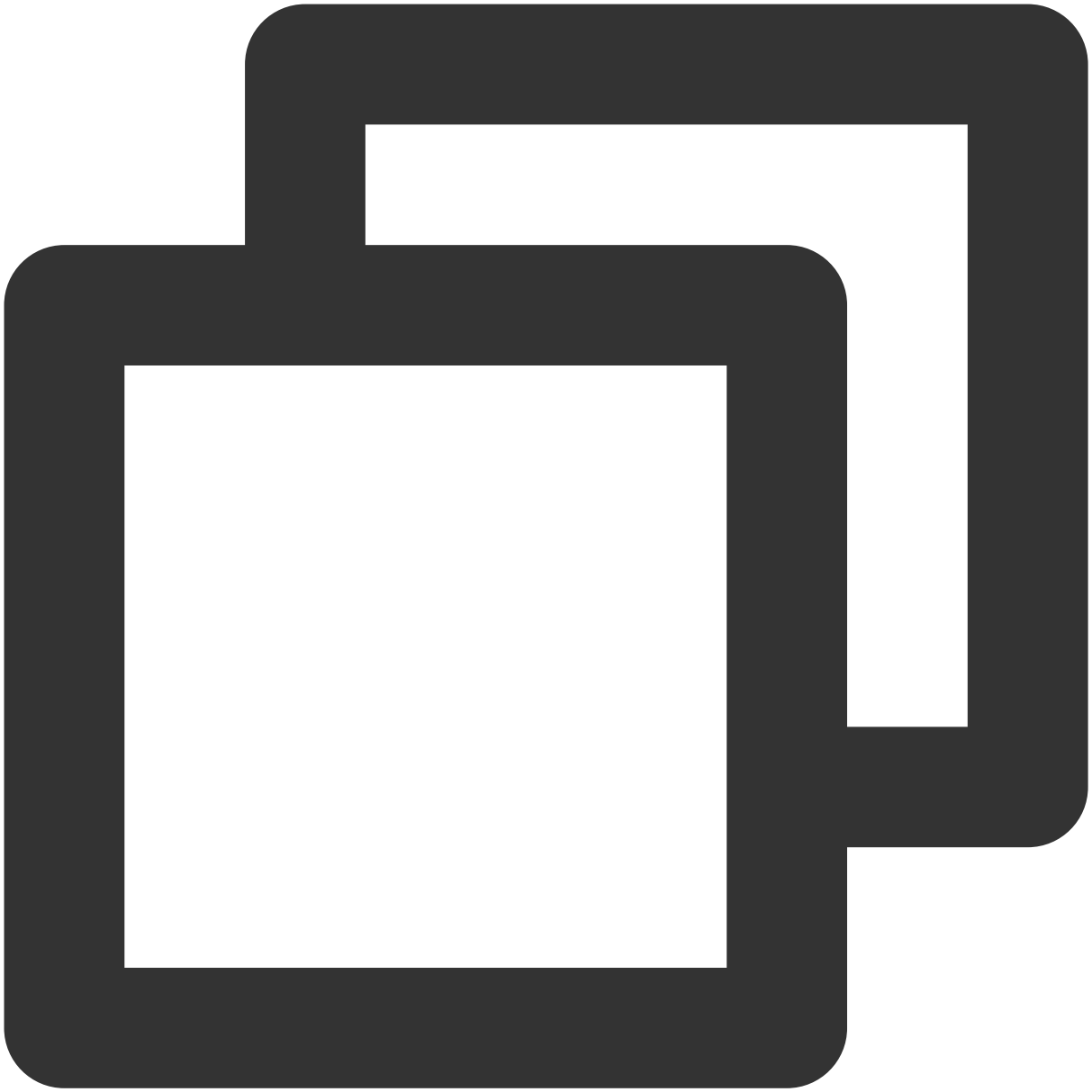
若您需要对**播放域名**进行鉴权配置，您可以在 域名管理 选择播放域名或单击后面的**管理**，进入播放域名详情页，选择**访问控制**，查看**鉴权配置**标签，单击**编辑**，进行配置。

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。

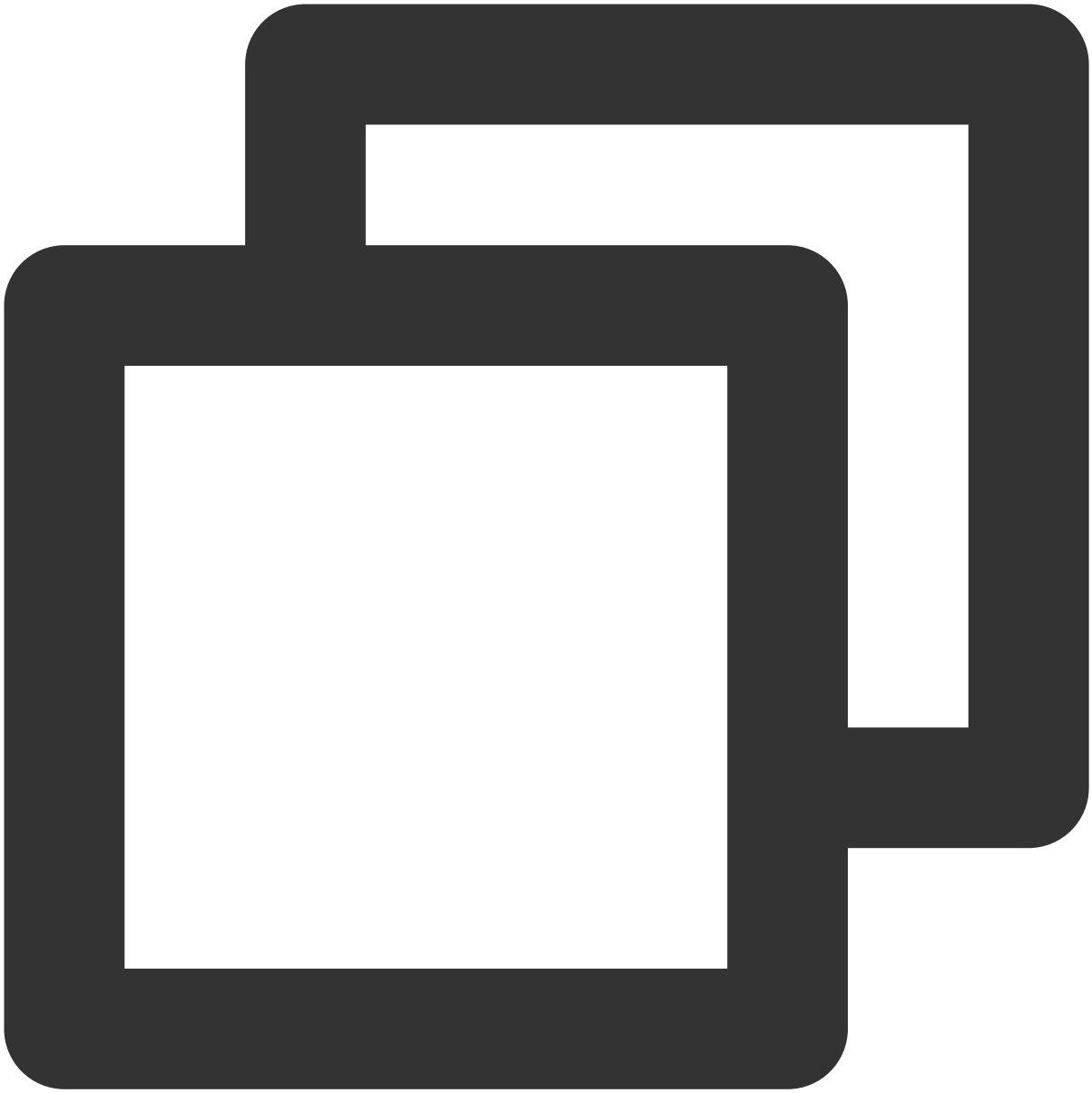
计算示例

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) = 5ee8ca6c28cbe415b40352969c
```

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

录制和回看

最近更新时间：2024-01-13 15:53:49

功能介绍

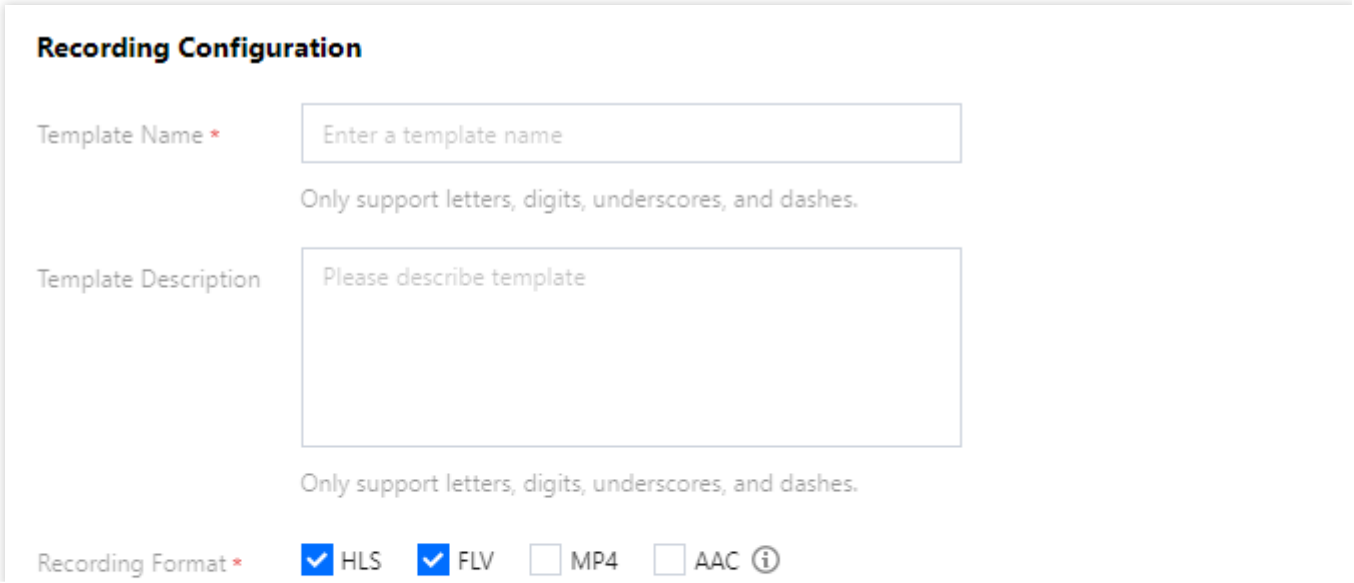
录制回看是指您可以把用户整个直播过程录制下来，然后作为点播视频用于回看。
在 App 上线的初期阶段，由于主播数量比较少，所以在直播列表中加入录制回看，能够在一定程度上丰富 App 在观众端的信息量。即使到 App 成长起来主播数量形成规模以后，好的直播内容的沉淀依然是必不可少的一个部分，每个主播的个人介绍里除了有名字、照片和个人信息，历史直播的视频回看更是不可或缺的一个重要组成部分。

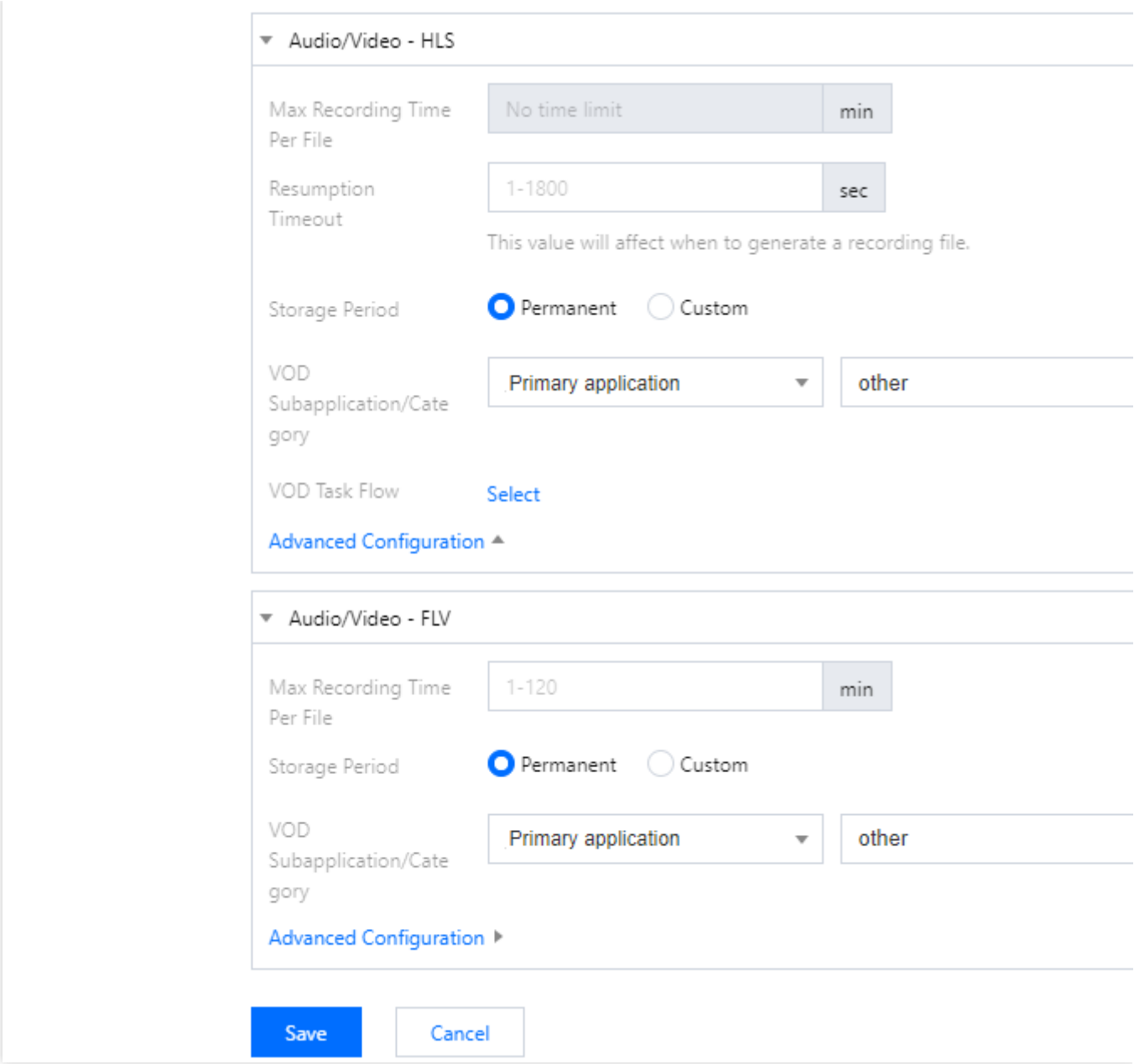
开启录制

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。



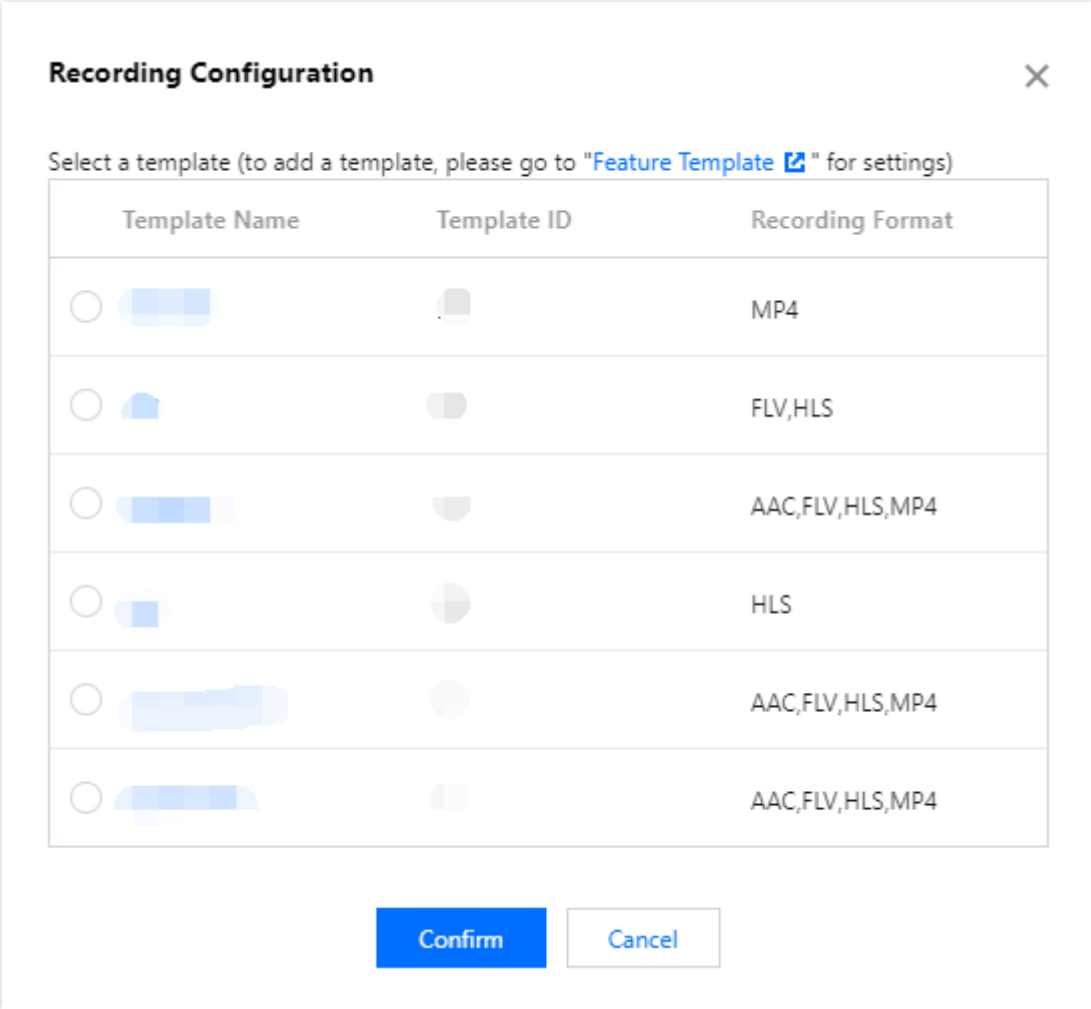


配置项		配置描述
单个录制文件时长（分钟）		录制 HLS 格式最长单个文件时长无限制，如果超出续录等待时长则新建文件继续录制。录制 MP4、FLV 或 AAC 格式单个文件时长限制为1分钟 - 120分钟。
续录等待时长（秒）		仅 HLS 格式支持文件推流中断续录，续录等待时长可设置为0s - 1800s。
保存时长（天）		单个录制文件保存最大时长均为1500天，文件保存时长0为永久。可选择永久存储或指定时间。
指定点播应用/分类		支持录制至云点播 VOD 指定 子应用 的点播分类中，默认录制至账号主应用，仅支持写入状态为开启的子应用。
高级配置	视频存储策略	支持对录制至云点播上的媒资进行降冷操作，若录制文件不需要频繁访问，可以使用降冷功能来实现低频访问长期存储。若录制视频为正常业务回放需要，

		标准存储即可满足需求，默认为标准存储。 选择 **标准存储** 时，若目前选中的应用已开启降冷策略，录制文件会先生成标准存储文件后再根据降冷策略进行降冷，可 查看策略。 选择 **低频存储** 时，若目前选中的应用/分类已开启降冷策略，录制文件会先直接生成低频存储文件后再判断是否执行点播降冷策略。
	点播任务流处理	单击 **选择绑定的任务流** 可选择绑定点播子应用下已建立的任务流，或从当前点播任务流选择界面点击任务流名称前往点播控制台新增/修改任务流配置。绑定成功后，在生成录制文件后执行点播任务流模板，产生对应的 云点播费用。

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。

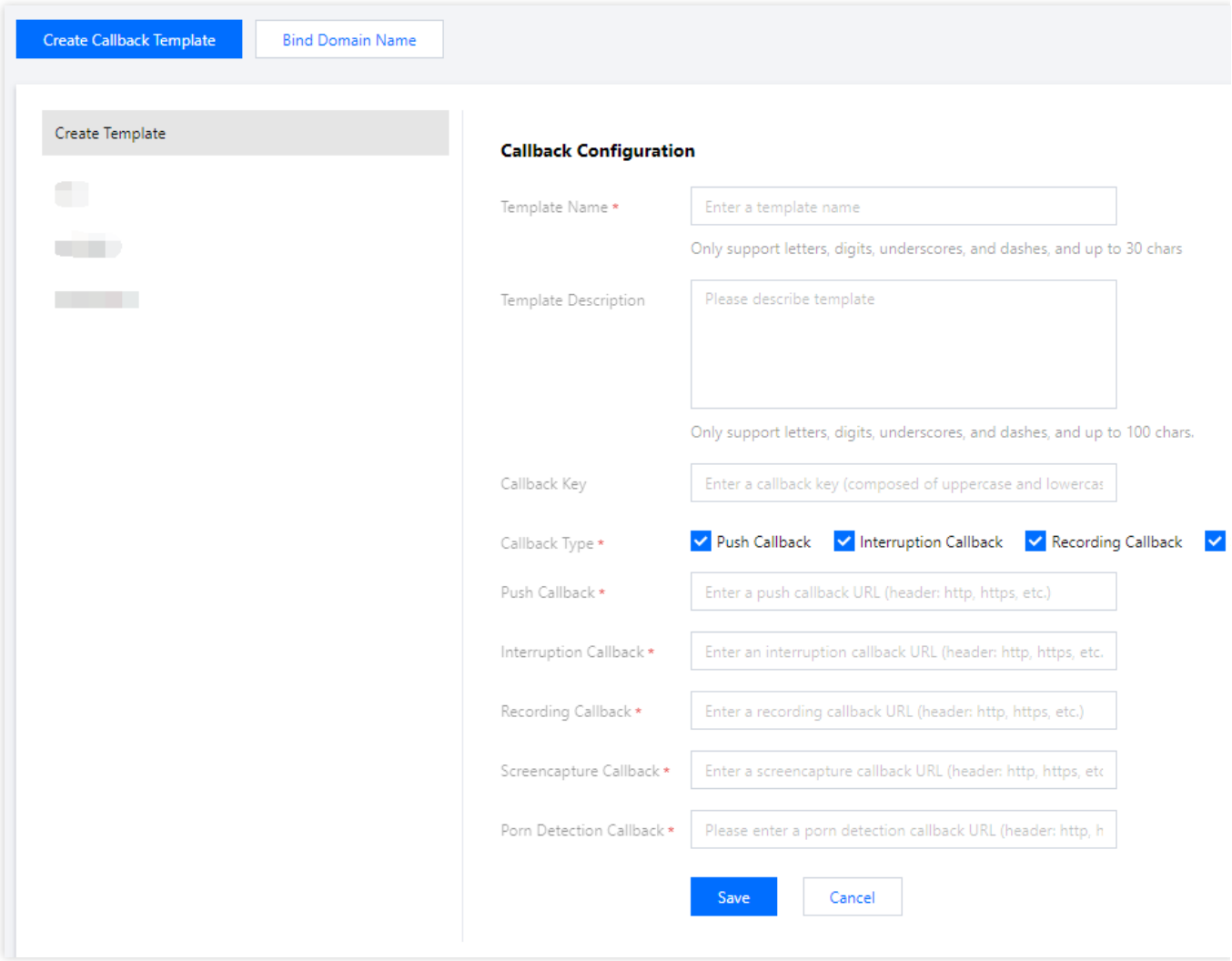


获取文件

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：

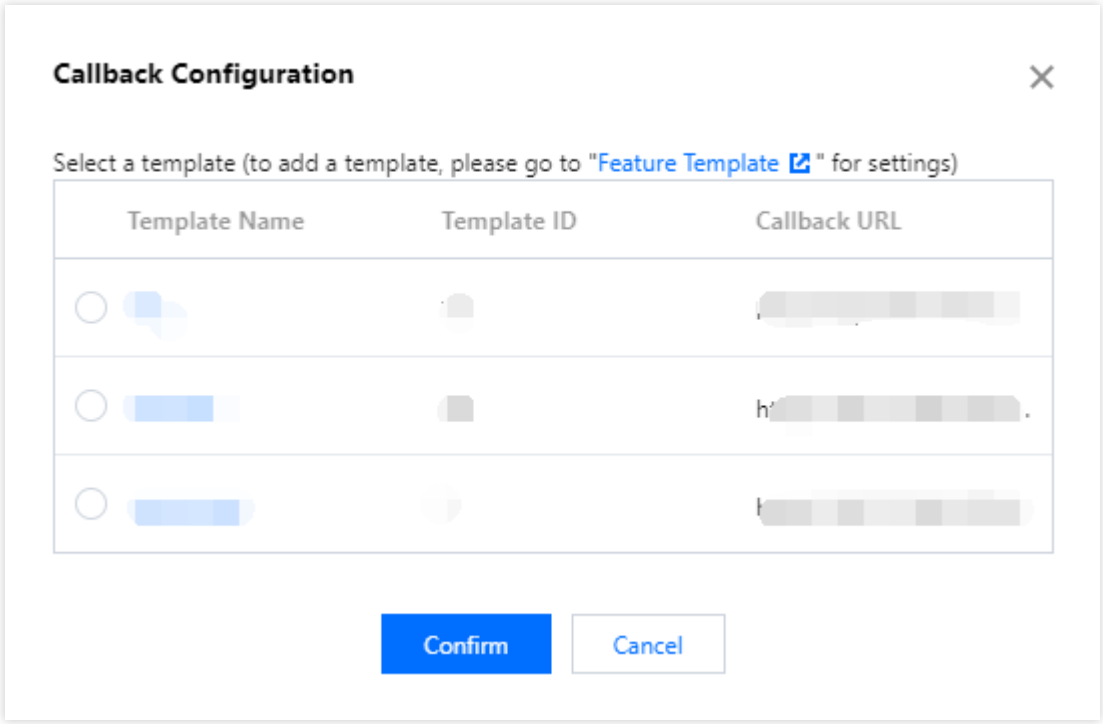
方案1：被动监听通知

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。

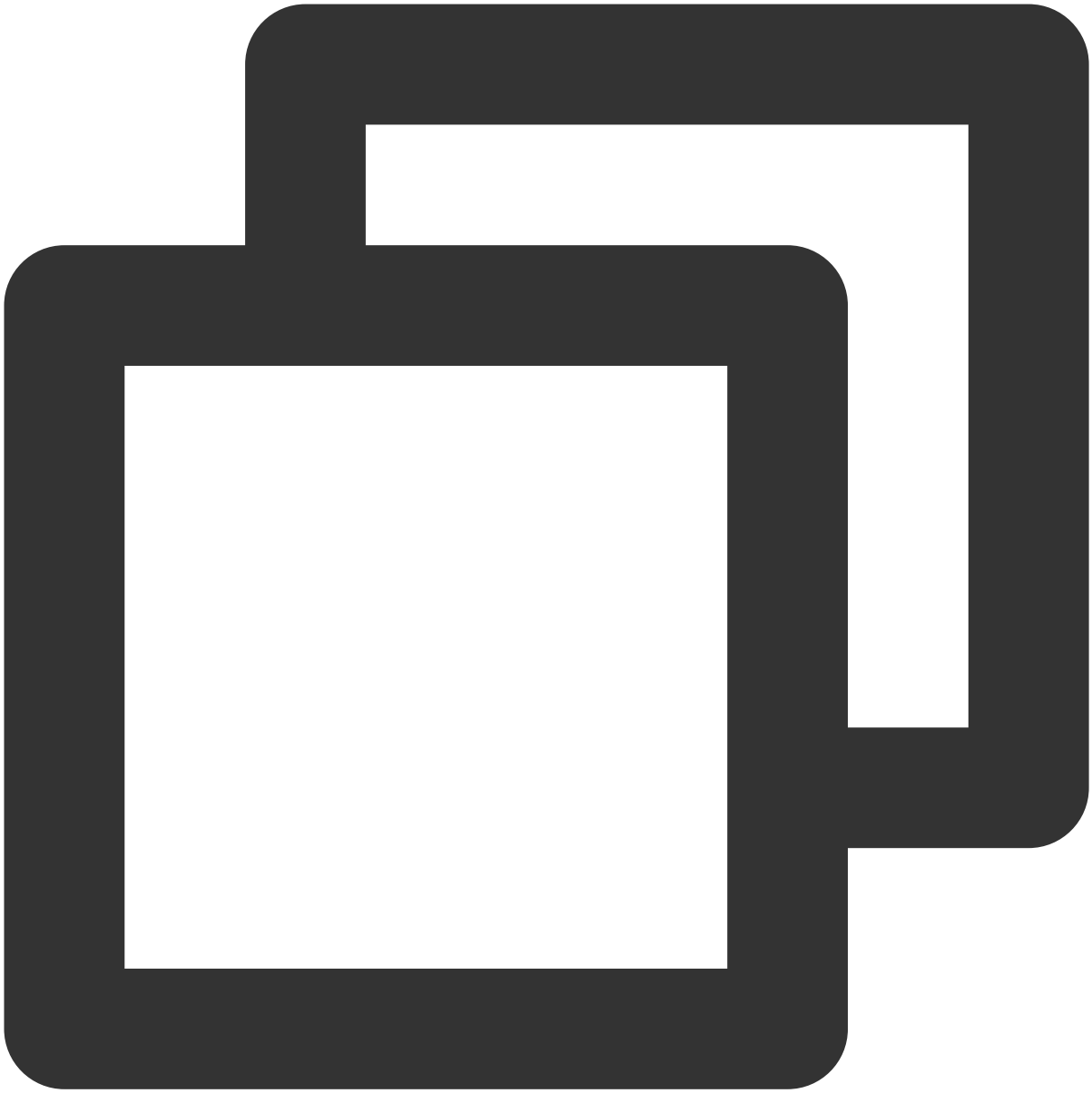


关联域名

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,
```

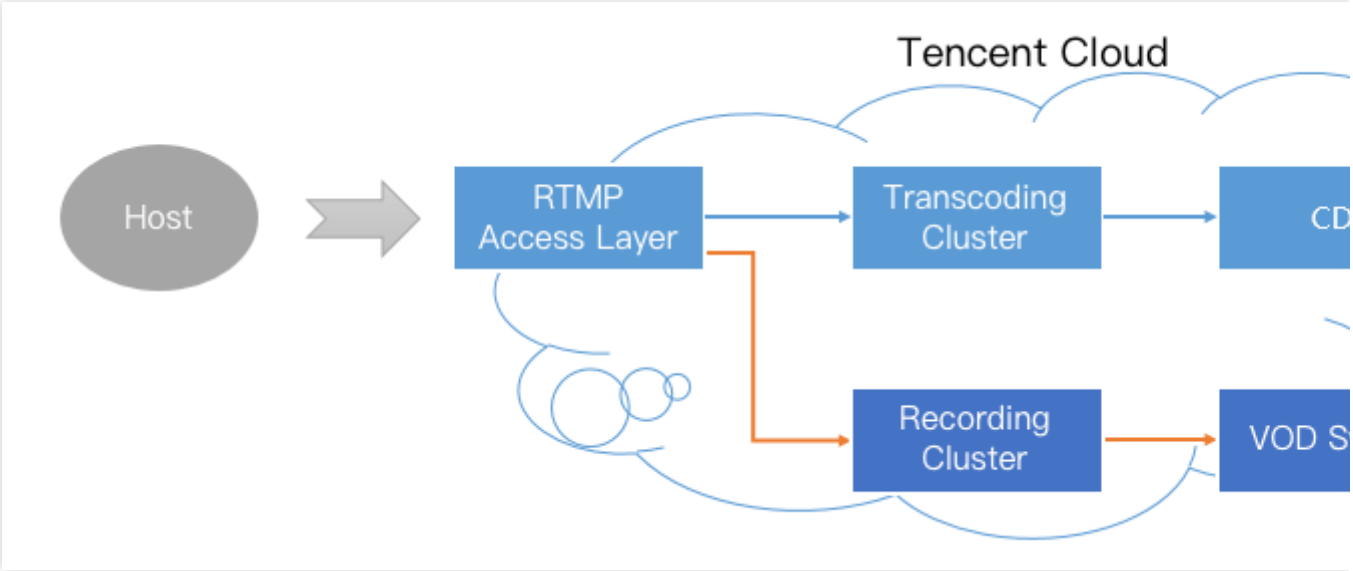
```
    "video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
    "video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9
}
```

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？



对于一条直播流，一旦开启录制，音视频数据就会被旁路到录制系统。主播的手机推上来的每一帧数据，都会被录制系统追加写入到录制文件中。
一旦直播流中断，接入层会立刻通知录制服务器将正在写入的文件落地，将其转存到点播系统中，并为其生成索引，这样您在点播系统中就会看到这个新生成的录制文件了。同时，如果您配置了录制事件通知，录制系统会将该文件的**索引 ID** 和**在线播放地址**等信息通知给您之前配置的服务器上。
但是，如果一个文件过大，在云端的转出和处理过程中就很容易出错，所以为了确保成功率，我们的单个录制文件最长不会超过120分钟，您可以通过 RecordInterval 参数指定更短的分片。

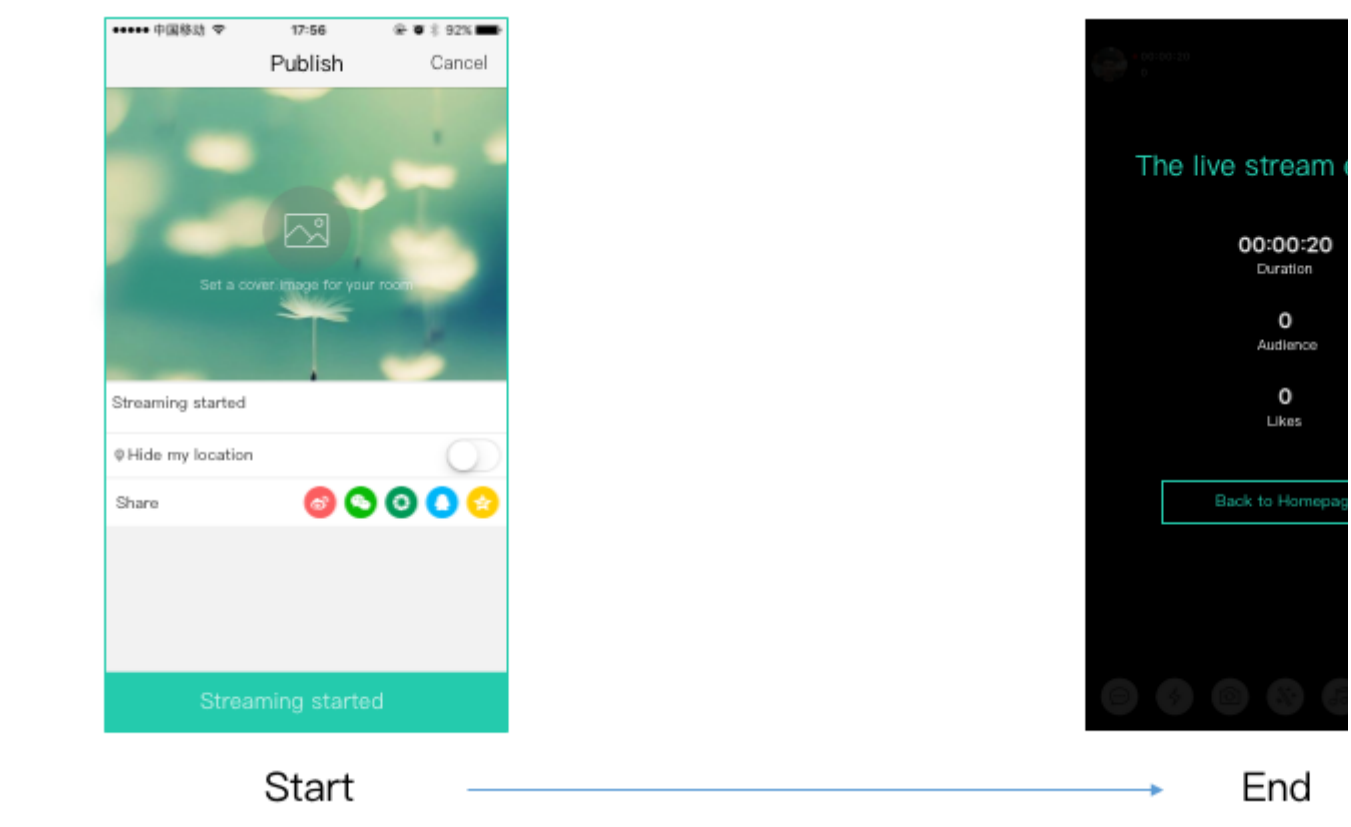
2. 一次直播会有几个录制文件？

录制 MP4、FLV 或 AAC 格式：单个文件时长限制为1分钟 - 120分钟。您可以通过 创建录制模板 接口中的 RecordInterval 参数指定更短的分片。

如果一次直播过程非常短暂，录制模块未启动就结束推流，那么系统会无法生成录制文件。
如果一次直播时间不算长（小于 RecordInterval），且中途没有推流中断的事情发生，那么通常只有一个文件。
如果一次直播时间很长（超过 RecordInterval ），那么会按照 RecordInterval 指定的时间长度进行分片，分片的原因是避免过长的文件在分布式系统中流转时间的不确定性。
如果一次直播过程中发生推流中断（之后 SDK 会尝试重新推流），那么每次中断均会产生一个新的分片。
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。续录超时时长可设置为0s - 1800s。

3. 如何知道哪些文件属于某一次直播？

准确来说，作为 PAAS 的腾讯云并不清楚您的一次直播是怎么定义的，如果您的一次直播持续了20分钟，但中间有一次因为网络切换导致的断流，以及一次手动的停止和重启，那么这算是一次直播还是三次呢？
对于普通的移动直播场景，我们一般定义如下的界面之间的这段时间为一次直播：



所以来自 App 客户端的时间信息很重要，如果您希望定义这段时间内的录制文件都属于这次直播，那么只需要用直播码和时间信息检索收到的录制通知即可（每一条录制通知事件都会携带 StreamID、开始时间和结束时间等信息）。

4. 如何把碎片拼接起来？

目前腾讯云支持使用云端 API 接口拼接视频分片 。