互动直播

旁路直播

产品文档

腾讯云

文档目录

旁路直播
直播码模式下旁路直播

最近更新时间：2018-06-15 18:34:24

直播码模式下旁路直播和录制支持项

旁路直播开启方式	旁路直播录制方式	多录制格式	录制回调	开发便利程度	可靠性	资源消耗
自动	自动	FLV/HLS/MP4	☒	※※※※	※※※※	※
手动	自动	FLV/HLS/MP4	☒	※※※	※※※	※※
自动	手动	MP4	☒	※※	※※	※※※
手动	手动	MP4	☒	※	※	※※※※

> **注意：**
> PC 端支持摄像头分享与屏幕分享，移动端仅支持摄像头分享，不支持屏幕分享。

自动旁路直播

开通方式

在 互动直播控制台 的【旁路直播配置】中开启。

自动旁路直播启动的时机

- 用户开启摄像头或者屏幕分享成功后，会触发启动旁路直播的逻辑。这个启动过程一般会在 3s 内完成。
- 如果启动失败，后台会间隔 5s 做重试启动操作。可以通过回调 URL 获得旁路直播开启成功的通知。
- 用户退出房间后会停止旁路直播。

手动旁路直播

手动旁路直播的注意事项

- 在进入音视频房间开启摄像头之后或者启动屏幕分享之后启动旁路直播，在客户端退出房间之前停止旁路直播。
- 考虑到外网接入，网络可能存在丢包，延迟等情况，启动旁路直播需要有重试的机制，但是重试间隔建议是 3s 以上。
- 注意携带的房间 ID，主播 ID 等参数保证对应关系。
- 可以在旁路直播的参数中选择是否开启录制。

开始旁路直播 SDK 接口

Android：

```
ILivePushOption option = new ILivePushOption()
    .encode(ILivePushOption.Encode.HLS)        // 旁路直播协议类型
    .recordFileType(ILivePushOption.RecordFileType.RECORD_HLS_FLV_MP4)     // 录制文件格式
    .channelName("频道 1")                      // 频道名称 直播码模式下无意义
    .channelDesc("我的频道")                    // 频道描述 直播码模式下无意义
    .recordId(123456);                    //手动推流自动录制时，如果需要后台识别特定的录制文件，用户可以通过这个
ILiveRoomManager.getInstance().startPushStream(option, new ILiveCallBack<ILivePushRes>() {
    @Override
    public void onSuccess(ILivePushRes data) {
        iPushId = data.getChnIId();
        Log.d("PUSH_DBG", "startPush->success id: "+data.getChnIId());
        if (null != data.getUrls()){
            // 打印推流类型及地址
            for (ILivePushUrl url : data.getUrls()){
                Log.d("PUSH_DBG", "type:"+url.getEncode()+", "+url.getRateType()+":"+url.getUrl());
            }
        }
    }
    @Override
    public void onError(String module, int errCode, String errMsg) {
        Log.d("PUSH_DBG", "startPush->failed:"+module+"|"+errCode+"|"+errMsg);
    }
});
```

iOS：

```
ILivePushOption *option = [[ILivePushOption alloc] init];
ILiveChannelInfo *info = [[ILiveChannelInfo alloc] init];
info.channelName = @"测试频道"; //直播码模式下无意义
info.channelDesc = @"测试频道描述"; //直播码模式下无意义
option.channelInfo = info;
option.encodeType = ILive_ENCODE_HLS;
option.recrodFileType = ILive_RECORD_FILE_TYPE_HLS;
option.recordId = 123456;//手动推流自动录制时，如果需要后台识别特定的录制文件，用户可以通过这个字段做区分。
[[ILiveRoomManager getInstance] startPushStream:option succ:^(id selfPtr) {
    AVStreamerResp *resp = (AVStreamerResp *)selfPtr;
    NSLog(@"推流成功 %@", [resp urls]);
    NSLog(@"推流获取到的频道 ID：%llu", resp.channelID)
} failed:^(NSString *module, int errId, NSString *errMsg) {
    NSLog(@"推流失败");
}];
```

PC：

```
void OnStartPushStreamSuc( PushStreamRsp &value, void *data )
{
    //开始推流成功
```

```
    m_channelId = value.channelId;   //频道 ID，停止推流时需要使用此 ID;直播码模式下，此值始终未 0;
    m_pushUrls = value.urls.;       //推流地址
}
void OnStartPushStreamErr( int code, const char *desc, void* data )
{
    //开始推流失败
}
PushStreamOption pushOpt;
pushOpt.pushDataType = E_PushCamera;       //推送数据类型
pushOpt.encode = RTMP;                     //推流数据编码方式
pushOpt.recordFileType = RecordFile_NONE;  //推流时自动录制的文件类型
pushOpt.recordId = 123456;                 //手动推流自动录制时，如果需要后台识别特定的录制文件，用户可以通过这
GetILive()->startPushStream( pushOpt, OnStartPushStreamSuc, OnStartPushStreamErr, NULL );
```

停止旁路直播 SDK 接口

Android：

```
ILiveRoomManager.getInstance().stopPushStream(iPushId, new ILiveCallBack() {
    @Override
    public void onSuccess(Object data) {
        Log.d("PUSH_DBG", "stopPush->success");
    }
    @Override
    public void onError(String module, int errCode, String errMsg) {
        Log.d("PUSH_DBG", "stopPush->failed:"+module+"|"+errCode+"|"+errMsg);
    }
});
```

iOS：

```
// pushID 为 startPushStream 中返回的频道 ID
NSArray *chids = @[(pushID)];
[[ILiveRoomManager getInstance] stopPushStreams:chids succ:^{
    NSLog(@"停止推流成功");
} failed:^(NSString *module, int errId, NSString *errMsg) {
    NSLog(@"停止推流失败");
}];
```

PC：

```
void OnStopPushStreamSuc( void* data )
{
    //停止推流成功
}
void OnStopPushStreamErr( int code, const char *desc, void* data )
{
```

```
    //停止推流失败
}
GetILive()->stopPushStream(m_channelId, OnStopPushStreamSuc, OnStopPushStreamErr, NULL);
```

拼接播放地址方式的变化

直播码计算规则

- 直播码= **BIZID_MD5(房间号_用户名_数据类型)** 。
- 字符串传输按 utf-8 编码。摄像头数据类型是 main，屏幕分享的数据类型是 aux。而 BIZID 是一个固定的参数，可以在直播控制台的顶部找到它。
- 假如 BIZID=8888,用户名=14y2l2c，房间号=293710，在进行摄像头分享，则 MD5(293710_14y2l2c_main)=81265058829fd2e50c8ec2ac78d55127 。那么直播码就是 8888_81265058829fd2e50c8ec2ac78d55127 。

拼接播放地址规则

- 播放地址如下：

  传输协议://BIZID.liveplay.myqcloud.com/live/直播码[.格式]

- 以上面为例，三种格式的播放地址依次如下：

  rtmp：

  rtmp://8888.liveplay.myqcloud.com/live/8888_81265058829fd2e50c8ec2ac78d55127

  flv：

  http://8888.liveplay.myqcloud.com/live/8888_81265058829fd2e50c8ec2ac78d55127.flv

  hls：

  http://8888.liveplay.myqcloud.com/live/8888_81265058829fd2e50c8ec2ac78d55127.m3u8

- 我们强烈建议**业务后台拼接播放地址，再下发给客户端。**这样容易应对播放地址拼接规则的变动。

旁路直播事件服务器回调通知

旁路直播会收到两种 event_type。**0** 代表旁路直播中断，**1** 代表旁路直播开启，同时消息体会额外包含如下信息：

字段名称	类型	含义
appname	string	推流路径
App	string	推流域名
appid	integer	用户标识
stream_param	string	推流参数中的 cliRecoId 是透传客户端推流时传入的 recordId，可以通过这个字段过滤特定的录制文件

示例：腾讯云通知直播流（1234_15919131751）发生断流（event_type=0）事件。

```
{
    "app": "1234.livepush.myqcloud.com",
    "appid": 1251659802,
    "appname": "live",
    "channel_id": "1234_15919131751",
    "errorcode": 0,
    "errmsg": "",
    "event_time": 1484033909,
    "event_type": 0,
    "node": "123.126.122.22",
    "sequence": "2480768755672606517",
    "sign": "fef79a097458ed80b5f5574cbc13e1fd",
    "stream_id": "1234_15919131751",
    "stream_param": "txSecret=e48f758df8f3c736ebecf6183640330c2&txTime=5a20f5ca&from=interactive&clie
    "t": 1473126233
    "user_ip": "123.112.131.123"
}
```

附录

回调事件的参数格式

消息组织格式

通知信息是以 JSON 格式进行组织的，然后放在 HTTP POST 协议体里。注意这里的 POST 格式的 ContentType 是 application/json，而不是 multipart/form-data，所以**不要使用 PHP 或者 Java 里读取表单字段的函数**来读取信息。

公共的头信息

如下的字段是每种类型的通知消息都一定会携带的。

字段名称	类型	含义	备注
t	string	有效时间	UNIX 时间戳(十进制)
sign	string	安全签名	MD5(KEY+t)
event_type	int	事件类型	目前可能值为： 0、1、100、200
stream_id	string	直播码	标示事件源于哪一条直播流
channel_id	string	直播码	同 stream_id

- **stream_id | channel_id（直播码）**
  在直播码模式下，stream_id 和 channel_id 两个字段都是同一个值，有两个不同的名字主要是历史原因所致。

- **t（过期时间）**
  来自腾讯云的消息通知的默认过期时间是 10 分钟，如果一条通知消息中的 t 值所指定的时间已经过期，则可以判定这条通知无效，进而可以防止网络重放攻击。t 的格式为十进制 UNIX 时间戳，即从 1970 年 1 月 1 日（UTC/GMT 的午夜）开始所经过的秒数。

- **sign（安全签名）**
  sign = MD5(key + t)：腾讯云把 API 鉴权 key 和 t 进行字符串拼接后，通过 MD5 计算得出 sign 值，并将其放在通知消息里，您的后台服务器在收到通知消息后可以根据同样的算法确认 sign 是否正确，进而确认消息是否确实来自腾讯云后台。

- **event_type（通知类型）**
  目前腾讯云支持三种消息类型的通知：0 — 断流； 1 — 推流；100 — 新的录制文件已生成；200 — 新的截图文件已生成。

回调 URL 的应答方式

- 很多客户会担心消息丢了怎么办，比如客户的服务器宕机了一下会儿，消息会不会丢失呢？

  腾讯云后台目前的消息可靠性保证机制是基于**简单重传**实现的，即：**如果一条通知消息没有成功发送到您指定的回调 URL，腾讯云会反复重试。**

- 那怎么确认消息是已经送达您的服务器了呢？

  这里是需要您的协助的：**当您的服务器成功收到一条 HTTP 事件通知消息时，请回复：200 的状态码**，以代表您已经成功收到了消息，从而避免腾讯云反复重复通知。如果腾讯云没有收到正确回包，会以 1 次/分钟，最多 10 次的重试逻辑继续通知。

> 代表：“嗯，我（客户服务器）已经您的通知了，请（腾讯云）不要再不断地发消息给我”。

频道模式旁路直播

最近更新时间：2018-06-15 18:34:28

概述

旁路直播功能可以把互动直播上行的数据转码成通用格式进行推流分发，以方便用户通过 Web 或流媒体播放器观看。**使用旁路直播功能前，请先在控制台开通腾讯云直播服务，否则将无法使用**，开通操作指引参考 快速入门。

Android SDK 接口

开始旁路直播

设置推流参数：

```
ILivePushOption option = new ILivePushOption();
option.channelName("新随心播推流");
option.encode(TIMAvManager.StreamEncode.RTMP);
```

推流参数 ILivePushOption：

字段名	字段类型	默认值	说明
channelName	String	可选	设置频道名称
channelDesc	String	可选	设置频道描述
channelPassword	String	可选	设置频道播放密码
record	boolean	NO	是否同时开启录制
waterMark	boolean	NO	是否开启水印
waterMarkId	long	可选	水印 id
sdkType	TIMAvManager.SDKType	可选	设置当前 sdk 类型
rateType	TIMAvManager.RateType	原始码率	支持的码率
encode	TIMAvManager.StreamEncode	RTMP	设置推流编码类型

开始旁路推流：

```
ILiveRoomManager.getInstance().startPushStream(option, new ILiveCallBack<TIMAvManager.StreamRes>() {
    @Override
```

```
        public void onSuccess(TIMAvManager.StreamRes data) {
            //旁路推流成功
            List<TIMAvManager.LiveUrl> liveUrls = data.getUrls();
            streamChannelID = data.getChnlId();
        }

        @Override
        public void onError(String module, int errCode, String errMsg) {
            //旁路推流失败
        }
});
```

结束旁路直播

```
ILiveRoomManager.getInstance().stopPushStream(streamChannelIDs, new ILiveCallBack() {
        @Override
        public void onSuccess(Object data) {
            //停止旁路推流成功
        }

        @Override
        public void onError(String module, int errCode, String errMsg) {
            //停止旁路推流失败
        }
});
```

参数名	参数类型	说明
ids	List	要停止推流的频道 ID 数组

Android 旁路直播功能的详细实现参见 新随心播。

iOS SDK 接口

开始旁路直播

设置推流参数：

```
ILivePushOption *option = [[ILivePushOption alloc] init];
ChannelInfo *info = [[ChannelInfo alloc] init];
info.channelName = @"新随心播推流";
info.channelDesc = @"新随心播推流描述测试文本";
option.channelInfo = info;
```

```
option.encodeType = encodeType;
option.sdkType = sdkType;
```

推流参数 ILivePushOption：

字段名	字段类型	默认值	说明
channelInfo	ChannelInfo	必填	旁路直播频道信息
record	BOOL	NO	是否同时开启录制
waterMark	BOOL	NO	是否开启水印
waterMarkId	uint32_t	可选	水印 id
sdkType	AVSDKType	可选	SDK 业务类型
rateType	AVRateType	原始码率	支持的码率
encodeType	AVEncodeType	必填	编码格式

频道参数 ChannelInfo：

字段名	字段类型	默认值	说明
channelName	NSString	必填	直播频道的名称
channelDesc	NSString	可选	直播频道的描述
channelPassword	NSString	可选	为接收方播放器设置的密码

开始旁路推流

```
[[ILiveRoomManager getInstance] startPushStream:option succ:^(id selfPtr) {
    AVStreamerResp *resp = (AVStreamerResp *)selfPtr;
    NSLog(@"频道的 ID=%d,AVLiveUrl 列表=%@",resp.channelID,resp.urls);
} failed:^(NSString *module, int errId, NSString *errMsg) {
    NSLog(@"推流失败");
}];
```

回调结果 AVStreamerResp：

字段名	字段类型	说明
channelID	UInt64	创建频道的 ID
urls	NSArray	AVLiveUrl 列表
recordTaskId	uint32_t	录制任务 id

URL 参数 AVLiveUrl：

字段名	字段类型	说明
type	AVEncodeType	编码格式
playUrl	NSString	播放 url
rateType	AVRateType	码率

结束旁路直播

```
[[ILiveRoomManager getInstance] stopPushStreams:@[@(_channelId)] succ:^{
        NSLog(@"已停止推流");
    } failed:^(NSString *module, int errId, NSString *errMsg) {
        NSLog(@"停止推流失败");
    }];
```

参数名	参数类型	说明
channelIds	NSArray	要停止推流的频道 ID 数组

iOS 旁路直播功能的详细实现见 新随心播。

观看方案

腾讯云 Web 播放器

视频云 Web 播放器是由腾讯视频云自主开发的一款播放器，同时支持 RTMP 和 HLS。单击查看 Web 播放器演示，将启动推流后得到的观看地址输入文本框即可播放。更多详情请参阅 官方文档。

示例：

```
<div id="id_video_container" style="width:100%;height:1px;"></div>
<script src="http://qzonestyle.gtimg.cn/open/qcloud/video/live/h5/live_connect.js" charset="utf-8"></script>
<script>
    (function(){
        var player = new qcVideo.Player(
            "element_id",//页面放置播放位置的元素 ID
            {
                "width": 640, //播放器宽度，单位像素(必选参数)
                "height": 480, //播放器高度，单位像素(必选参数)
                "live_url": "rtmp://http://xxx.liveplay.qcloud.com/live/xxx",//直播地址，支持 hls 和 rtmp 、 flv 三种格式(
                "live_url2": "http://http://xxx.liveplay.qcloud.com/live/xxx.m3u8"//直播地址，同上，（可选参数）
            }
```

```
    );
  })();
</script>
```

腾讯视频云自主开发播放器 TcPlayer

TcPlayer 是由腾讯视频云自主开发的一款播放器，同时支持 RTMP、FLV、HLS。单击查看 TcPlayer 演示，更多详情请参阅 Web 播放器 TCPlayer Lite。

示例：

```
<div id="id_test_video" style="width:100%; height:auto;"></div>
<script src="//restcplayer.qcloud.com/sdk/tcplayer-web-1.0.1.js" charset="utf-8"></script>
<script>
  (function(){
    var player = new TcPlayer('id_test_video', {
      "rtmp": "rtmp://http://xxx.liveplay.qcloud.com/live/xxx", //请替换成实际可用的播放地址
      "flv": "http://http://xxx.liveplay.qcloud.com/live/xxx.flv", //请替换成实际可用的播放地址
      "m3u8": "http://http://xxx.liveplay.qcloud.com/live/xxx.m3u8", //请替换成实际可用的播放地址
      "live": true,
      "coverpic" : "http://www.test.com/myimage.jpg",
      "width" :  '480',//视频的显示宽度，请尽量使用视频分辨率宽度
      "height" : '320'//视频的显示高度，请尽量使用视频分辨率高度
    });
  })();
</script>
```

若需使用 TcPlayer 广告功能，需在页面引入 Google IMA SDK 。

```
<script type="text/javascript" src="//imasdk.googleapis.com/js/sdkloader/ima3.js"></script>
```

通过 adTagUrl 和 auth 参数使用广告功能，帐号及 License 信息可登录 TcPlayer 官网 注册申请，或联系 tcplayer@tencent.com 咨询反馈。

```
var player = new TcPlayer('id_test_video', {
 /* Advertisement-related parameter */
 "adTagUrl": "http://ad_tag_url",    //VAST,VMAP,VAPID 视频广告 Tag
 "auth": {
   "user_id": "your_user_id",       //广告帐户 ID
   "app_id": "your_app_id",         //应用 ID
   "license": "your_license"        //应用 license
 }
});
```

客户端 SDK 播放器（FFmpeg）

FFmpeg 是提供跨平台解决方案的开源音视频处理工具，最新 FFmpeg 已自带 rtmp。单击进入 FFmpeg 下载（注：FFmpeg 采用 LGPL 或 GPL 许可证(依据您选择的组件)，如使用，请遵循许可证的规则）代码可以参考 ffplay.c(如需编译 ffplay，需安装 SDL)。

开发步骤：

```
//初始化
av_register_all();
avformat_network_init();
//打开输入
AVFormatContext *ifmt_ctx = avformat_alloc_context();
avformat_open_input(&ifmt_ctx, in_filename, 0, 0));
//找到视频流和音频流
int videoindex = -1, audioindex = -1, i;
for (i = 0; i<ifmt_ctx->nb_streams; i++) {
   if (ifmt_ctx->streams[i]->codec->codec_type == AVMEDIA_TYPE_VIDEO){
      videoindex = i;
   }
   if (ifmt_ctx->streams[i]->codec->codec_type == AVMEDIA_TYPE_AUDIO){
      audioindex = i;
   }
}
//初始化音视频编解码器
AVCodecContext *pVideoCodecCtx = ifmt_ctx->streams[videoindex]->codec;
AVCodec *pVideoCodec = avcodec_find_decoder(pVideoCodecCtx->codec_id);
AVCodecContext *pAudioCodecCtx = ifmt_ctx->streams[audioindex]->codec;
AVCodec *pAudioCodec = avcodec_find_decoder(pAudioCodecCtx->codec_id);
//开始读取数据并解码
int got_picture, got_frame;
AVPacket *pkt = av_packet_alloc();
AVFrame * pVideoFrame = av_frame_alloc();
AVFrame * pAudioFrame = av_frame_alloc();
av_read_frame(ifmt_ctx, pkt);
avcodec_decode_video2(pVideoCodecCtx, pVideoFrame, &got_picture, pkt);
avcodec_decode_audio4(pAudioCodecCtx, pAudioFrame, &got_frame, pkt);
```

至此，音视频原始数据已经获取到，播放和渲染，依赖使用哪种方案。例如，使用 SDL2，依次调用 SDL_CreateWindow、SDL_CreateRenderer、SDL_CreateTexture 然后拷贝数据并显示：

```
SDL_Texture *txt;
//省略 SDL_CreateTexture 初始化 txt
//注：调用前先确认 pVideoFrame->pCodecCtx->pix_fmt 为 AV_PIX_FMT_YUV420P
//如不是，可先用 sws_scale 转格式
SDL_UpdateYUVTexture(txt, NULL,
   pVideoFrame ->data[0], pVideoFrame ->linesize[0],
   pVideoFrame ->data[1], pVideoFrame ->linesize[1],
   pVideoFrame ->data[2], pVideoFrame ->linesize[2]);
```

或者例如使用 OpenCV，OpenCV 只接受 RBG 数据格式，因此需先转格式，再显示。

```
//转码
AVFrame * pFrameRBG = av_frame_alloc();
int size = av_image_get_buffer_size(AV_PIX_FMT_BGR24,
    pVideoCodecCtx->width, pVideoCodecCtx->height, 1);
uint8_t *out_bufferRGB = new uint8_t[size];
av_image_fill_arrays(pFrameRBG->data, pFrameRBG->linesize,
    out_bufferRGB, AV_PIX_FMT_BGR24,
    pVideoCodecCtx->width, pVideoCodecCtx->height, 8);
struct SwsContext *img_convert_ctx = sws_getContext(
    pVideoCodecCtx->width, pVideoCodecCtx->height,
    pVideoCodecCtx->pix_fmt,
    pVideoCodecCtx->width, pVideoCodecCtx->height, AV_PIX_FMT_BGR24,
    SWS_BICUBIC, NULL, NULL, NULL);
sws_scale(img_convert_ctx,
      (const uint8_t* const*)pVideoFrame->data, pVideoFrame->linesize,
    0, pVideoCodecCtx->height,
    pFrameRBG->data, pFrameRBG->linesize);
//拷贝数据
Mat frame;
if (frame.empty()){
    frame.create(cv::Size(pVideoCodecCtx->width,
        pVideoCodecCtx->height), CV_8UC3);
}
memcpy(frame.data, out_bufferRGB, size);
//显示
imshow("title", frame);
```

> 注：声音播放方式可以用 DirectShow，这里就不展开了。上面仅为简略调用步骤示意，实际使用需自行开发，并且还需很多细化工作，例如使用 jitterbuf 来防抖动，音视频同步等。

常见问题

同一音视频房间的多路流是否支持合并（包括混音）？

目前，一个 QAVSDK 实例的旁路推流只支持推一路语音+一路视频（且必须有视频），虽然同一个互动直播房间可以旁路 4 路不同的音视频流，但是每个流之间各自独立，并不能叠加或混合。

App 退后台对旁路推流有什么影响？

- **Windows** 退后台，与前台无异。
- **iOS** 退后台，可能会被系统挂起导致推流终端，具体详情请参阅 Apple 公司相关技术文档。
- **Android** 退后台，理论上系统是不会自动杀进程的。但各家深度定制的安卓系统行为不尽相同，需要具体情况具体分析。此外，安卓系统自身的一些保护机制也会在某些情况下（如资源不足等）自动结束进程。

直播过程中上行方网络中断对旁路推流有什么影响？如何恢复推流？

如果视频上行方网络中断，则：

- 通过 RTMP 观看的用户，在网络中断的 7 秒后进入，会提示直播已结束。
- 通过 HLS 观看的用户，在后台缓存分片播放完后中断。

视频上行方网络超时断开后，推流后台会将频道保留 1 小时，这段时间内该用户重新推流所返回的 **观看地址不会改变**。如果 1 小时内该用户仍未重新推流，则频道将会回收，即该用户重新推流所获得的观看地址会 **重新生成**。

直播频道的数量限制是与应用相关还是与腾讯云账号相关？

直播频道的数量上限是与腾讯云账号绑定的，一个帐号下多个应用是共用频道资源的。例如，一个账号频道上限 50，该账号下的应用 A 使用了 30 个，则该账号下的其他应用就只有 20 个频道可用。

错误码

错误码	错误说明	处理建议
1001	权限错误	一般是 sdkappid 填写错误导致
1002	账户不存在	请排查接口参数填写的用户数据是否正确
6012	推流超时	请查看上行方的网络状况，App 层面也根据需要引入适当的重试，重试间隔建议为 30 秒。如果依然有问题可以联系我们协助排查
20101	通道数超过上限	推流通道数存在上限，在推流控制台检查并删除无用的通道，或根据实际需要进行扩容
20318	未开通直播资质	请先开通腾讯云直播服务
20406	用户欠费	检查是否已欠费
50002	输入参数检验错误	检查用户 ID 是否填写错误，sdkappid 是否填写错误
50003	后端没有拉取到拉流的 url	反馈腾讯客服
50004	推流请求的推流类型错误	检查推流类型字段填写是否正确
50005	连接后端控制台超时	可能是网络问题，重试处理，重试失败反馈腾讯客服
50006	连接后端控制台超时	可能是网络问题，重试处理，重试失败反馈腾讯客服
50007	后端返回参数为空	反馈腾讯客服
40000000	SDK 请求解析失败	推流请求字段填写是否完整

错误码	错误说明	处理建议
40000001	SDK 请求解析失败-没有推流请求包体	推流请求字段填写是否完整
40000002	SDK 请求解析失败-没有推流请求操作字段	推流请求字段填写是否完整
40000003	SDK 请求解析失败-缺少推流请求的输出编码（HLS/RTMP 等）	推流请求字段填写是否完整
40000004	SDK 请求解析失败-视频源类型错误（摄像头/桌面等）	推流请求字段填写是否完整
40000005	SDK 请求解析失败-请求操作错误（请求推流、停止推流）	推流请求字段填写是否完整
40000006	请求推流的时候检查用户 ID 不正确	推流请求字段填写是否正确
40000007	推流房间 ID 填写成 0	请检查推流的房间 ID 的填写
40000201	请求服务器内部数据打包错误	反馈腾讯客服
40000202	请求服务器内部数据打包错误	反馈腾讯客服
40000203	请求服务器内部数据打包错误	反馈腾讯客服
40000207	请求推流服务器通讯错误-拉取推流服务器地址失败	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000208	请求推流服务器通讯错误-请求推流服务器超时	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000301	解析推流服务器回包错误-数据包解析失败	反馈腾讯客服
40000302	解析推流服务器回包错误-数据包解析失败	反馈腾讯客服
40000303	解析推流服务器回包错误-没有返回 IP	反馈腾讯客服
40000304	解析推流服务器回包错误-没有返回端口	反馈腾讯客服
40000305	解析推流服务器回包错误-没有返回结果	反馈腾讯客服

错误码	错误说明	处理建议
40000306	解析推流服务器回包错误-返回 URL 长度溢出	反馈腾讯客服
40000401	查询房间获取 grocery 服务 IP 错误	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000402	查询房间拉取 grocery 数据错误	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000403	查询房间拉取 grocery 不存在（请求推流的房间不存在）	检查是否成功开房，推流的用户 ID，groupid 是否填写正确
40000404	查询房间流控服务器超时	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000405	查询房间回包错误-数据包解析失败	反馈腾讯客服
40000406	查询房间回包错误-数据包解析失败	反馈腾讯客服
40000407	查询房间回包错误-数据包解析失败	反馈腾讯客服
40000408	查询房间回包错误-没有返回结果	反馈腾讯客服
40000409	查询房间回包错误-数据包解析失败	反馈腾讯客服
40000410	请求推流的房间不存在	检查是否成功开房，推流的用户 ID，groupid 是否填写正确,或者用户是否已经退出房间
40000411	发起推流用户不在房间内	检查是否成功开房，推流的用户 ID，groupid 是否填写正确,或者用户是否已经退出房间
40000412	停止推流重复发送，用户已经停止推流	如果是推流停止操作说明已经停止，重复停止操作，无需处理
40000413	停止推流重复发送，用户已经停止推流	如果是推流停止操作说明已经停止，无需处理
40000414	查询房间-服务器内部操作类型错误	可能是网络问题，重试处理，重试失败反馈腾讯客服
40000415	启动推流重复发送，用户正在推流	如果是推流启动操作说明已经是在推流状态，无需处理

错误码	错误说明	处理建议
40000500	启动推流频率控制	非异常，同一个用户在 3 秒内重复请求推流会返回次错误，重试请求需要在上一次请求发起 3 秒后