实时音视频

服务端功能

腾讯云

【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或95716。

文档目录

服务端功能
实现云端录制与回放

最近更新时间：2025-11-26 12:22:51

场景说明

在远程教育、秀场直播、视频会议、远程定损、金融双录、在线医疗等应用场景中，考虑取证、质检、审核、存档和回放等需求，常需要将整个视频通话或互动直播过程录制和存储下来的情况。

> **说明：**
> 下文将针对实时音视频最新推出的云端录制能力进行使用说明。自**2022年08月01日**起新创建的应用（SdkAppId）录制功能类型为新版云端录制。若您当前的 TRTC 应用（sdkappid）使用的是旧版云端录制，详情请参见 旧版云端录制 。判断当前应用的云端录制的类型和旧版云端录制能力切换为新版的方式，详情请参见 控制台 > 云端录制说明 。

功能说明

通过 TRTC 的云端录制功能，您可以将房间中的每一个用户的音视频流都录制成独立的文件（单流录制），或者把同一个房间的音视频媒体流合流录制成一个文件（合流录制）。

- **订阅流：** 我们支持通过制定订阅用户的黑白名单的方式来指定您需要订阅的用户媒体流（**仅支持 API 录制**）。
- **转码参数：**合流的场景下，我们支持通过设置编解码的参数来指定录制的视频文件的质量。
- **合流参数：**合流的场景下，我们支持多种灵活可变的自动多画面布局模板和自定义布局模板。
- **文件存储：**支持存储到云点播 VOD 或对象存储 COS。
- **回调通知：**我们支持回调通知的能力，通过配置您的回调域名，云端录制的事件状态会通知到您的回调服务器。

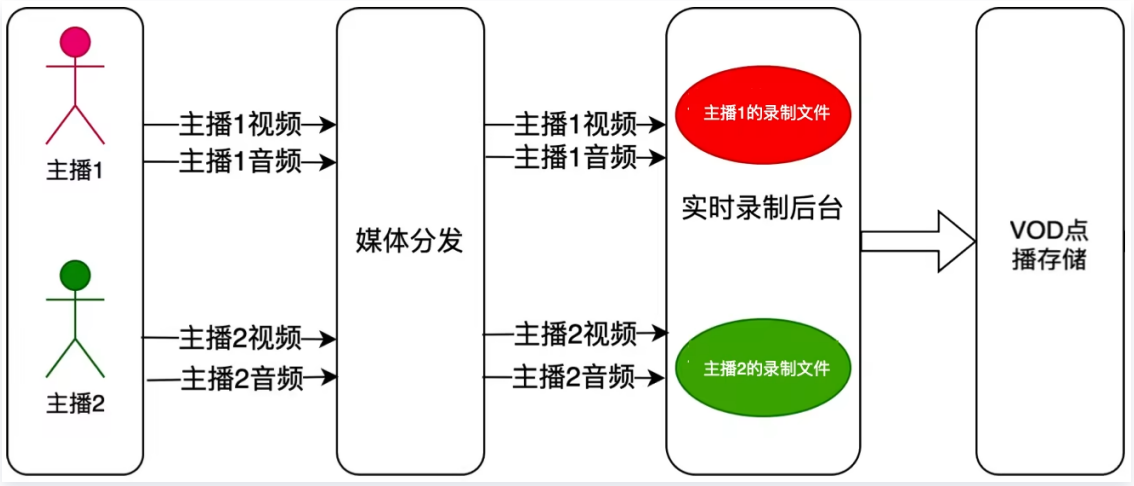
单流录制和合流录制说明

单流录制

如下图所示为单流录制的场景，房间1234里面主播1和主播2都上行了音视频流，假设您订阅了主播1和主播2的音视频流，并设置录制模式为单流录制，录制后台会分别拉取主播1和主播2的音视频流，并把他们录制成独立的媒体文件包含：

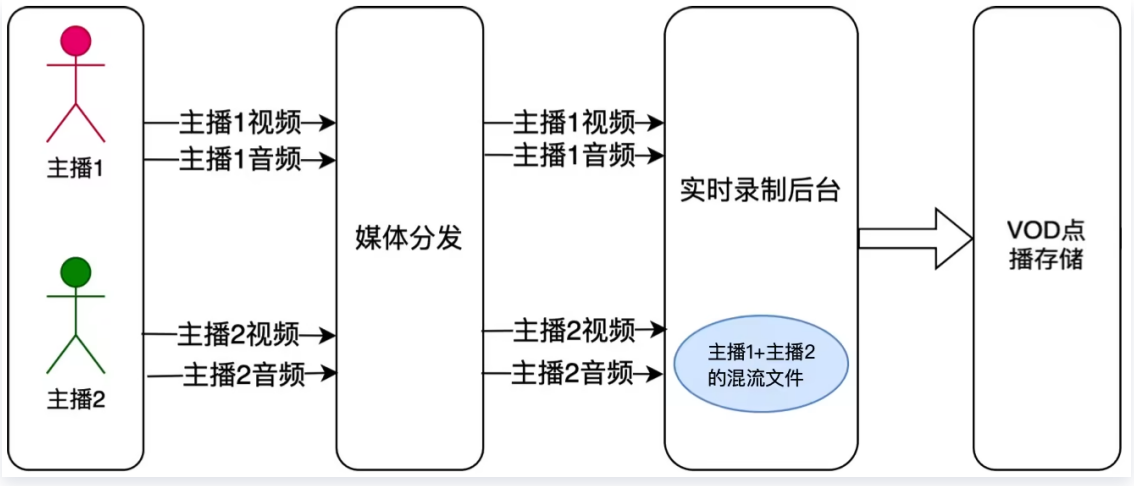
- 主播1的一个音视频录制文件。
- 主播2的一个音视频录制文件。

录制后台会把这些文件上传到您指定的云存储平台（云点播 VOD 或对象存储 COS）。具体录制流程如下：



合流录制

如下图所示为合流录制的场景，房间1234里面有主播1和主播2都上行了音视频流，假设您订阅了主播1和主播2的音视频流，设置录制模式为合流录制，录制后台会分别拉取主播1和主播2的音视频流，并把他们的视频流按照您配置多画面模板进行合流，音频流进行混音，最后把媒体流混合成一路媒体文件。包含：合流后的一个音视频录制文件，具体发起方式请见 API手动录制 。
录制后台会把这些文件上传到您指定的云存储平台。具体录制流程如下：



录制文件命名和文件切分说明

录制 MP4/AAC 文件名命名规则

- **单流录制** MP4/AAC 文件名规则：`<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Index>.mp4/aac`
- **合流录制** MP4/AAC 文件名规则：`<SdkAppId>_<RoomId>_<Index>.mp4/aac`

录制 HLS 文件名命名规则

- **单流录制 HLS 文件名规则：** `<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Type>.m3u8`
- **合流录制 HLS 文件名规则：** `<SdkAppId>_<RoomId>.m3u8`
- **字段含义说明：**

字段	含义
<SdkAppId>	录制任务的 SdkAppId
<RoomId>	录制的房间号，如果这里 RoomId 如果是字符串房间号，我们会对房间号先做 base64 操作，再把 base64 后的字符串中符号'/'替换成 '-' (中划线)，符号'='替换成 '.'
<UserId>	录制的用户 ID，UserId 会先做 base64 操作，再把base64后的字符串中符号'/'替换成 '-' (中划线)，符号'='替换成 '.'
<MediaId>	主辅流标识，main 代表主流（摄像头），aux 代表辅流（屏幕分享）
<Index>	如果没有触发切片逻辑（大小超过2GB或超过设置的切片时长）则无该字段，否则为切片的索引号，从1开始递增
<Type>	录制文件流类型，audio/video/audiovideo

> **说明：**
> 自定义设置文件名称前缀：使用 API 录制 存储至云点播 VOD 时，可通过 TencentVod 中的 UserDefineRecordId 参数自定义文件名称前缀，前缀与默认录制文件名之间用__UserDefine_u_分隔。

录制文件切分说明

- **录制 MP4/AAC 文件切分的条件：**
  - 录制切分时长可设置范围1-1440分钟，默认1440分钟。
  - 单个 MP4/AAC 文件大小达到 2GB。
- **录制 HLS 文件切分的条件：**
  - 录制任务持续时间超过14天时，HLS 文件将会被切分。

录制上传云存储说明

录制后台会在录制结束后将录制的文件通过您指定的方式上传到云存储平台（云点播 VOD 或对象存储 COS），并通过回调的形式把播放地址发送给您。如果录制模式为单流录制模式，每一个订阅的主播都会有一个对应的播放地址；如果录制模式为合流录制模式，只有一个合流后媒体的播放地址。

1. 通过 API 发起录制：在存储参数 StorageParams 中必须指定 CloudVod（存储至云点播 VOD）或 CloudStorage（存储至对象存储 COS 或第三方云存储）的参数，请确保已经开通对应的云存储服务且未欠费。
2. 上传任务的过程中使用 DescribeCloudRecording 只能查询到录制任务进行状态，不会携带录制文件的信息。

> ⚠ **注意：**
> 文件录制后会上传至您指定云存储平台（云点播 VOD 或对象存储 COS），为确保录制文件成功，请确保您指定的云点播 VOD 或对象存储 COS 服务可用。

API 接口和录制并发限制

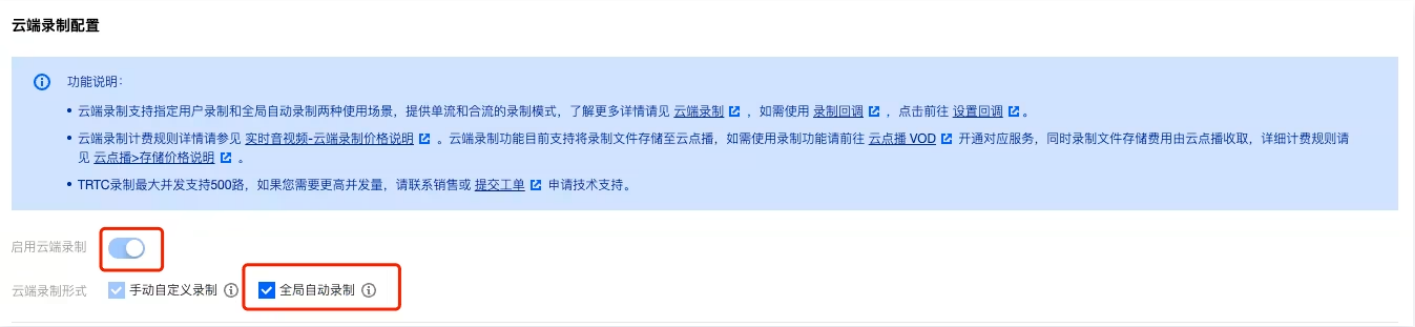
- 录制接口的调用频率限制为20qps（如需提高 QPS 请 提交工单）。
- 单个接口超时时间为6秒。
- 单个应用下**默认并发录制支持500路**（全局自动录制和 API 录制任务的总和），超过并发限制的任务会失败，如需更多并发路数，请 提交工单 联系我们。
- 单次录制任务最大支持同时订阅的房间内主播数为25个，主播只上行音频也会单独占据一路。

录制控制方案

TRTC 提供了两种云端录制方案，分别是 全局自动录制 和 API 手动录制，这两种方案并不冲突，可以同时使用两种录制方案，但会产生两份录制文件和费用。API 录制相比全局自动录制的优点是录制灵活、功能完备，客户可以指定录制订阅房间内的主播，自定义合流布局，录制中途更新布局和订阅等。全局自动录制的优点是录制不需客户启动和停止，由 TRTC 后台管理录制任务。

方案一：全局自动录制

TRTC 提供了一种无需手动发起并管理录制任务的自动录制方式，要使用该种录制方案，请前往**实时音视频控制台** > **应用管理** > **录制管理**中开启云端录制功能，完成全局自动录制模板配置并开启全局自动录制。
生效后（生效等待5-10分钟）TRTC 房间中的主播上行音视频后将触发启动录制任务，房间内主播都退房且超过设置的等待续录时间后将触发停止录制任务。



开启全局自动录制功能前请配置全局自动录制模板，全局自动录制支持 单流录制（即每个主播单个录制一个文件），开启后只对新创建的房间有效，对开启自动录制功能之前已经创建的房间不生效。

全局单流录制

全局单流录制录制格式支持音视频录制、纯音频录制和纯视频录制，录制文件支持 MP4、HLS 和 AAC（纯音频录制格式下），录制文件切片策略请参见 录制文件切分说明 。

配置项	说明
录制模式	• 单流录制：房间中的每个主播的视频画面都会单独保存成一份文件 如需录制多个主播混合后的画面，请使用 API手动合流录制
录制格式	• 音视频格式：录制房间内的音频和视频流，适用于视频通话、互动直播场景 • 纯音频格式：只录制房间内的音频流
文件格式	支持 MP4 、HLS 和 AAC（纯音频格式下）
单个录制文件时长	可用于指定录制文件切片时长，设置范围1-1440分钟，默认1440分钟
续录等待时长	设置续录超时时长，当打断间隔不超过设定的续录超时时长时，一次通话（或直播）只会生成一个文件，但需要等待续录时间超时后才能收到录制文件，单位：秒，**取值范围1 - 86400（默认30s）**。 **注意：在续录等待期内，单流录制会按照音频时长收取录制费用，请合理设置。**
录制文件存储	支持存储至 云点播 VOD 或 对象存储 COS 。 云点播：需支持指定云点播应用、录制文件在云点播的存储时间以及绑定点播任务流。 对象存储：存储至 COS 的服务由云点播联合提供，如需存储至 VOD，您需授权给 VOD 相关 COS 桶的读写权限，并需要完成您的存储桶 bucket 和云点播应用的绑定，绑定后云点播会为您创建一个应用，通过选择这个绑定的应用，可完成指定的存储桶设置。
回调地址与回调密钥	新版云端录制提供了详尽的录制事件功能，您可以配置可用的服务端url用于接收录制回调事件，同时支持配置回调密钥用于校验回调事件的安全性，更多请见 。

说明：

- 单流录制模式下房间内的音视频流将按照推流参数进行每一路单独录制，无需设置转码。
- 续录等待时长未到期内录制机器人会在房间内继续等待主播上行进而完成录制，并不会因为主播退房后就立即结束，请合理设置。
- 单流录制最多录制一个房间内的25个主播，如果超过25个主播将会按照进房时间由先到后排序，录制前25位主播（如需单流录制超过25位主播，请参见 API 录制 ）。

方案二：API 手动录制

启动录制

通过您的后台服务调用 REST API（CreateCloudRecording）来启动云端的录制，需要重点关注参数— **任务 ID（TaskId）**；这个参数是本次录制任务的唯一标识，您需要保存下这个任务 ID 作为后续针对这个录制任务接口操作的输入参数。

> **说明：**
> 1. 发起云端录制任务的接口 CreateCloudRecording 中需要您指定分配录制机器人的进房参数 UserId 和 UserSig（如何获取 UserSig），请不要与您房间内的正常主播或观众使用的 UserId 重复且不可与正在录制中的房间内指定的录制机器人 UserId 一致，否则会导致录制任务失败。
> 2. **手动录制下，您可以前往控制台配置回调地址，以接受录制回调事件**，请见 录制回调说明 。

录制的模式（RecordMode）

- **单流录制**：实时录制房间内每个主播的音频视频单录制为一个音视频文件上传到云存储平台（云点播 VOD 或对象存储 COS）。
- **合流录制**：将房间内您所订阅所有主播的音视频流混录成一个音视频文件上传到云存储平台（云点播 VOD 或对象存储 COS）。

指定录制用户（SubscribeStreamUserIds）

默认情况下，云端录制会录制房间内所有的媒体流（最多25路），超过25个用户，默认录制最先进房的25位主播。您也可以通过参数 SubscribeStreamUserIds 指定想要录制或者不想录制的主播用户的黑白名单信息，当然我们也支持在录制的过程中进行更新操作。单流录制场景，如果房间内主播超过25人，可以通过设置订阅名单发起多次录制任务实现。

指定存储位置和录制格式（StorageParams）

存储位置：支持存储至云点播 VOD 或对象存储 COS，请通过在 StorageParams 中 CloudVod 参数进行指定您的指定存储参数。
录制格式：默认录制格式是 MP4, 如果需要录制成 HLS 格式, 可通过 CloudVod 下 TencentVod 内的 MediaType 设置为1来指定格式为 HLS；如果需要录制AAC 格式文件， 可通过 CloudVod 下 TencentVod 内的 MediaType 设置为2来指定格式为 AAC（仅在 StreamType =1纯音频录制时有效）

录制开始的时间的获取

通过订阅回调，监听录制回调事件。在事件类型311中的 StartTimeStamp 字段您可以获取到录制文件对应的录制起始时间戳，EndTimeStamp 字段可以获取到对应的录制结束时间戳。

```
{
    "EventGroupId": 3,
    "EventType": 311,
    "CallbackTs": 1622186289148,
    "EventInfo": {
        "RoomId": "xx",
        "EventTs": "1622186289",
        "UserId": "xx",
        "TaskId": "xx",
```

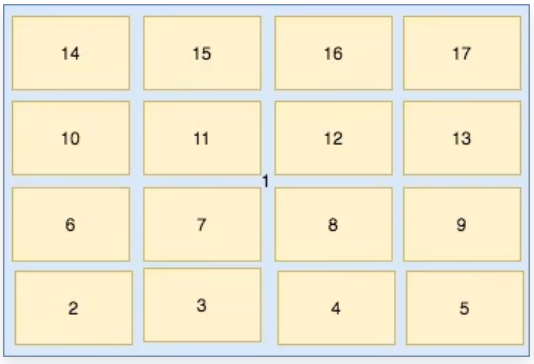
```
        "Payload": {
            "Status": 0,
            "TencentVod": {
              "UserId": "anchor1",
              "TrackType": "video",
              "MediaId": "main",
              "FileId": "xxxxx",
              "VideoUrl": "http://xxxxx",
              "CacheFile": "xxxxxx",
              "StartTimeStamp": 1622186279,
              "EndTimeStamp": 1622186811
        }
      }
    }
}
```

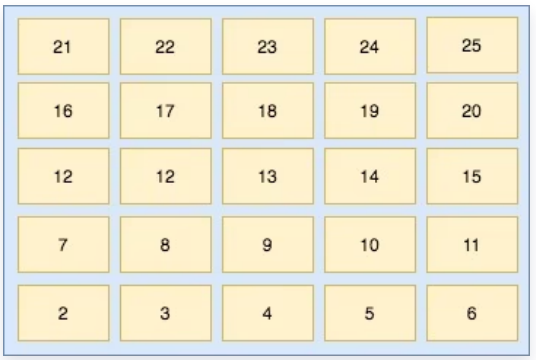
合流录制的布局模式参数（MixLayoutMode）

支持 悬浮布局 、屏幕分享布局 、九宫格布局（默认）和 自定义布局 四种布局：

- **悬浮布局**

  默认第一个进入房间的主播（也可以指定一个主播）的视频画面会铺满整个屏幕。其他主播的视频画面从左下角开始依次按照进房顺序水平排列，显示为小画面，小画面悬浮于大画面之上。当画面数量小于等于17个时，每行4个（4 × 4排列）。当画面数量大于17个时，重新布局小画面为每行5个（5 × 5）排列。最多支持25个画面，如果用户只发送音频，仍然会占用画面位置。
  悬浮布局随着订阅的子画面增加按照下图进行变化：

子画面小于等于17个时	• 每个小画面的宽和高分别为整个画布宽和高的 0.235 • 相邻小画面的左右和上下间距分别为整个画布宽和高的 0.012 • 小画面距离画布的水平和垂直边距也分别为整个画布宽和高的 0.012	

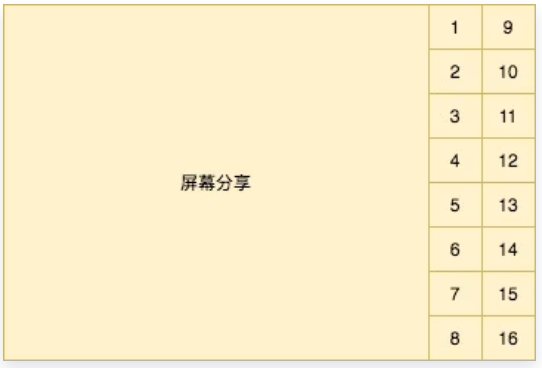
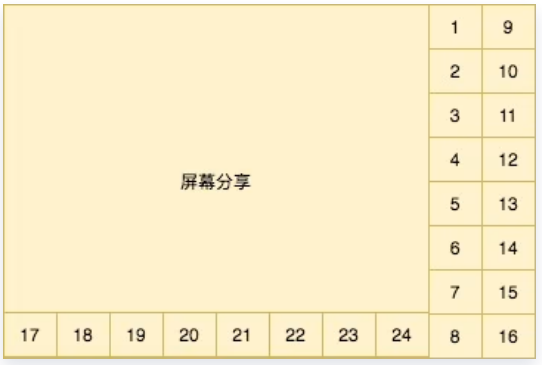
子画面大于17个时	• 每个小画面的宽和高分别为整个画布宽和高的 0.188 • 相邻小画面的左右和上下间距分别为整个画布宽和高的 0.01 • 小画面距离画布的水平和垂直边距也分别为整个画布宽和高的 0.01	

- **屏幕分享布局：**

指定一个主播在屏幕左侧的大画面位置（如果不指定，那么大画面位置为背景色），其他主播自上而下依次垂直排列于右侧。当画面数量少于17个的时候，右侧每列最多8人，最多占据两列。当画面数量多于17个的时候，超过17个画面的主播从左下角开始依次水平排列。最多支持24个画面，如果主播只发送音频，仍然会占用画面位置。

屏幕分享布局随着订阅的子画面增加按照下图进行变化：

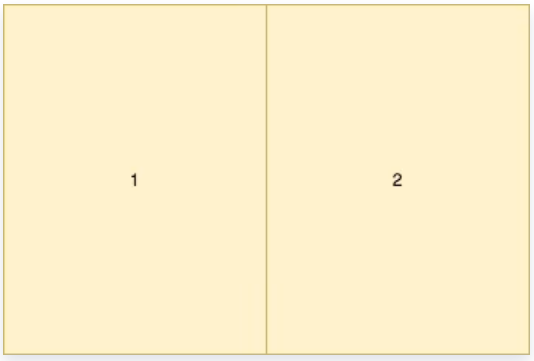
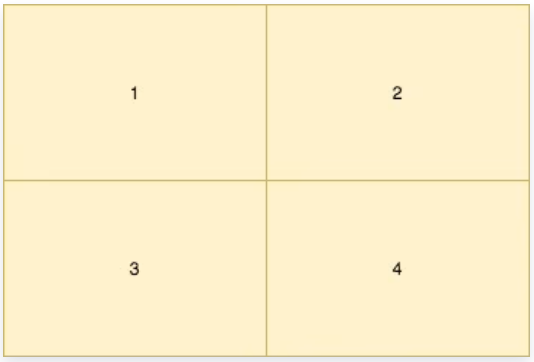
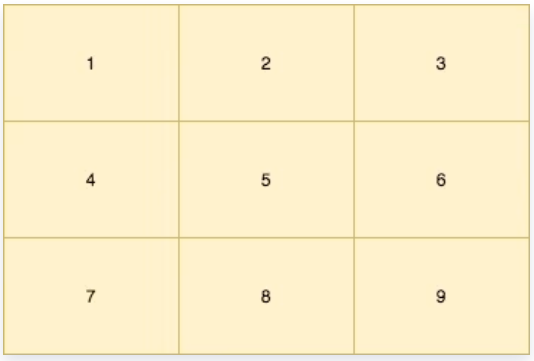
子画面小于等于5个时	• 右侧小画面的宽为整个画布宽的1/5，右侧小画面的高为整个画布高的1/4 • 左侧大画面的宽为整个画布宽的4/5，左侧大画面的高为整个画布高	
子画面大于5且小于等于7个时	• 右侧小画面的宽为整个画布宽的1/7，右侧小画面的高为整个画布高的1/6 • 左侧大画面的宽为整个画布宽的6/7，左侧大画面的高为整个画布高	

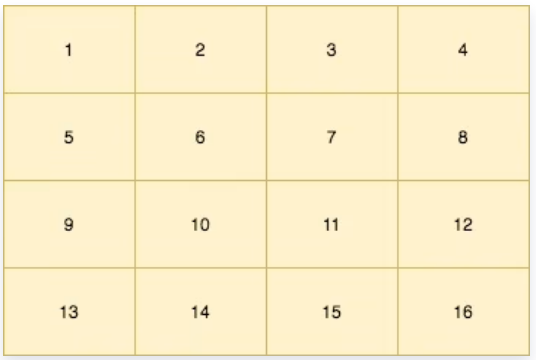
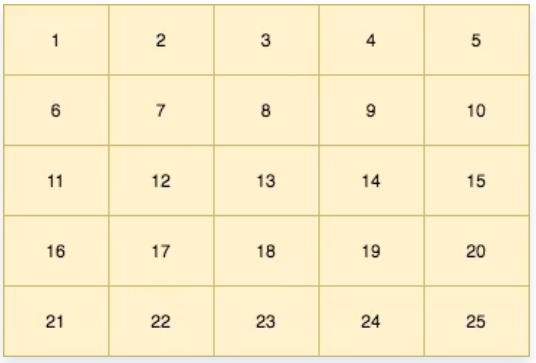
子画面大于7且小于等于9个时	• 右侧小画面的宽为整个画布宽的1/9，右侧小画面的高为整个画布高的1/8 • 左侧大画面的宽为整个画布宽的8/9，左侧大画面的高为整个画布高	
子画面大于9小于等于17个时	• 右侧小画面的宽为整个画布宽的1/10，右侧小画面的高为整个画布高的1/8 • 左侧大画面的宽为整个画布宽的4/5，左侧大画面的高为整个画布高	
子画面大于17个时	• 右（下）侧小画面的宽为整个画布宽的1/10，右（下）侧小画面的高为整个画布高的1/8 • 左侧大画面的宽为整个画布宽的4/5，左侧大画面的高为整个画布高的7/8	

- **九宫格布局**

  根据主播的数量自动调整每个画面的大小，每个主播的画面大小一致，最多支持25个画面。

  九宫格布局随着订阅的子画面增加按照下图进行变化：

子画面为1个时	每个小画面的宽和高分别为整个画布宽和高	
子画面为2个时	• 每个小画面的宽为整个画布宽的 1/2 • 每个小画面的高为整个画布高	
子画面小于等于4个时	每个小画面的宽和高分别为整个画布宽和高的 1/2	
子画面小于等于9个时	每个小画面的宽和高分别为整个画布宽和高的 1/3	

子画面小于等于16个时	每个小画面的宽和高分别为整个画布宽和高的 1/4	
子画面大于16个时	每个小画面的宽和高分别为整个画布宽和高的1/5	

- **自定义布局**

  根据您的业务需要在 MixLayoutList 内自己定制每个主播画面的布局信息。

合流录制的水印参数 (MixWatermark)

我们支持在合流录制中添加图片水印，最大支持个数为25个，可以在画布任意位置添加水印。

字段名	解释
Top	水印相对左上角的垂直位移
Left	水印相对左上角的水平位移
Width	水印显示的宽度
Height	水印显示的高度
url	水印文件的存储 URL

查询录制（DescribeCloudRecording）

如果需要，您可以调用该接口查询录制服务的状态。

> **注意：**
> - 只有录制任务存在的时候才能查询到信息，如果录制任务已经结束会返回错误。

- 如果是上传云点播 VOD 任务，该接口返回的 StorageFile 为空。

更新录制（ModifyCloudRecording）

如果需要，您可以调用该接口修改录制服务的参数，如订阅黑白名单 SubscribeStreamUserIds（单流和合流录制有效），录制的模板参数 MixLayoutParams（合流录制有效）。

> ⚠ **注意：**
> 更新操作是全量覆盖的操作，并不是增量更新的操作，您每次更新都需要携带全量的信息，包括模板参数 MixLayoutParams 和黑白名单 SubscribeStreamUserIds，因此您需要保存之前的启动录制的参数或者重新计算完整的录制相关参数。

停止录制（DeleteCloudRecording）

在录制结束之后需要调用停止录制（DeleteCloudRecording）的接口来结束录制任务，否则录制任务会等待到达预设的超时时间 MaxIdleTime 后自动结束。

> ⚠ **注意：**
> MaxIdleTime 的定义是房间内持续没有主播的状态超过 MaxIdleTime 的时长，这里如果房间是存在有主播，但是主播没有上行数据是不会进入超时的计时状态的，此时后台录制会持续工作。建议业务在录制结束的时候调用此接口结束录制任务。

录制回调事件

我们针对云端录制功能提供了多种的回调事件，帮助您及时了解录制任务的处理和完成情况，录制回调地址配置和事件说明请见 云端录制回调 。

录制文件管理

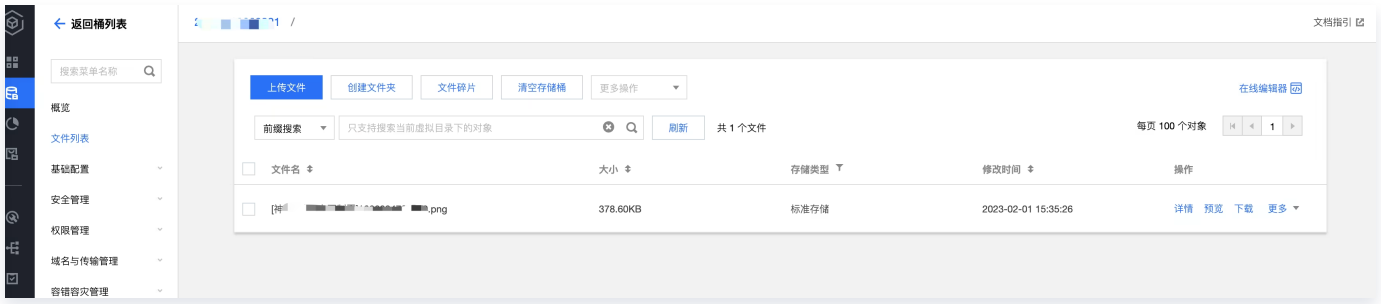
查找录制文件

结束房间完成录制任务后，TRTC 录制系统中录制下来的文件上传至您指定的云存储平台（云点播 VOD 或对象存储 COS）。您可以直接前往 云点播控制台 或 对象存储 COS 控制台 查找，也可以由您的后台服务器使用 REST API 进行定时筛选：

方式一：在点播控制台手动查找

1. 登录 云点播控制台，在左侧导航栏选择**媒资管理**。
2. 单击列表上方的**前缀搜索**，选择**前缀搜索**，在搜索框输入关键词，按照 录制文件命名规则 填入，例如合流录制下填入：`1400000123_1001`，单击 🔍，将展示视频名称前缀相匹配的视频文件。

3. 登录 对象存储 COS 控制台，选择您指定的存储桶 Bucket 进行查找：



另外您也可以前往 实时音视频控制台 > 录制管理 > 录制文件管理中筛选对应点播应用查看对应的录制文件。

存储至腾讯云 COS 的目标录制文件，回调事件内本身不提供播放链接，但可以通过固定规则拼接。拼接规则如下：`https://<bucket>.cos.<region>.myqcloud.com/<FileNamePrefix>/<Taskid>/<录制文件名>`

举个例子，客户发起录制对应的存储是：`trtc-test-125**39`，存储桶地域 `ap-shanghai`，录制文件前缀设置为 `prefix1/prefix2`，录制文件唯一任务Taskid 为 `m9-bVV*****zQtdRzgE.`，录制文件名 `140**311_68888.m3u8`，那么拼接出来的播放链接为：

`https://trtc-test-125**39.cos.ap-shanghai.myqcloud.com/prefix1/prefix2/m9-bVV*****zQtdRzgE./140**311_68888.m3u8`

> ⚠ **注意：**

> 如果 Taskid 如果有' + '字符，需要转义成' %2B '。COS 存储桶需要设置公有访问，才可以正常播放录制文件。

方式二：通过点播 REST API 查找

腾讯云点播系统提供了一系列 REST API 来管理其上的音视频文件，您可以通过 搜索媒体信息 这个 REST API 来查询您在点播系统上的文件。您可以通过 TrtcSdkAppIds（对应发起录制的应用 SdkAppid） 或 TrtcRoomIds（对应发起录制的房间号 Roomid） 参数进行匹配查找。
REST API请求示例：

```
https://vod.tencentcloudapi.com/?Action=SearchMedia
&TrtcSdkAppIds=1400xxxx123
&TrtcRoomIds=1234
&Sort.Field=CreateTime
&Sort.Order=Desc
&<公共请求参数>
```

> **说明：**
> 如需下载录制文件，请前往 点播控制台-媒资管理 下，找到对应的录制文件，在“操作”中进行下载。

接收录制文件

除了 查找录制文件，您还可以通过在控制台 配置回调地址，让腾讯云主动把新录制文件的消息推送给您的服务器。
房间里的最后一路音视频流退出后，该过程大约默认需要30秒至数分钟（具体时间根据您所录制的文件大小而定，若您设置了续录时间为300秒，则等待时间将在默认基础上叠加300秒）。转存完成后，腾讯云会通过您在 设置录制回调 中设置的回调地址（HTTP/HTTPS）向您的服务器发送通知。
腾讯云会将录制和录制相关的事件都通过您设置的回调地址推送给您的服务器，您可以通过接收**事件类型为311**的上传成功回调来获取录制文件的播放地址 VideoUrl，具体回调信息见下方：

```
{
  "EventGroupId": 3,
  "EventType": 311,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "UserId": "xx",
    "TaskId": "xx",
```

```
    "Payload": {
      "Status": 0,
      "TencentVod": {
        "UserId": "xx",
        "TrackType": "audio_video",
        "MediaId": "main",
        "FileId": "xxxx",
        "VideoUrl": "http://xxxx",
        "CacheFile": "xxxx.mp4",
        "StartTimeStamp": xxxx,
        "EndTimeStamp": xxxx
      }
    }
  }
}
```

删除录制文件

腾讯云点播系统提供了一系列 REST API 来管理其上的音视频文件，您可以通过 删除媒体 API 删除某个指定的文件。
REST 请求示例：

```
https://vod.tencentcloudapi.com/?Action=DeleteMedia
&FileId=52858907988664150587
&<公共请求参数>
```

回放录制文件

在线教育等场景中，通常需要在直播结束后多次回放录制文件，以便充分利用教学资源。

获取点播地址（VideoUrl）

在 接收录制文件 时，可以获取回调消息中 VideoUrl 字段，该字段为当前录制文件在腾讯云的点播地址。

对接点播播放器

根据使用平台对接点播播放器，具体操作参考如下：

- iOS 平台
- Android 平台
- Web 浏览器

> **注意：**
> 建议使用 专业版 TRTC SDK，专业版集合了 播放器（Player+）、直播 SDK 等功能，由于底层模块的高度复用，集成专业版的体积增量要小于同时集成两个独立的 SDK，并且可以避免符号冲突（symbol duplicate）的困扰。

相关费用

云端录制与回放功能使用到的功能包括：云端录制服务、云点播 VOD 或对象存储 COS 的回放文件存储与处理、云点播 VOD 或对象存储 COS 的播放服务，以及终端 SDK 播放点播视频的能力。可能会根据实际需求产生以下费用。

云端录制费用

云端录制费用取决于您所录制的时长和画面分辨率，同时根据录制模式的不同（单流或者合流）进行区分定价，录制费用计算公式如下：
云端录制费用 = 录制音频费用 + 录制视频费用 = 录制音频输入时长用量 × 单路或多路对应的音频单价 + 录制视频各分辨率档位输入时长用量 × 单路或多路对应的相应视频分辨率档位单价

> **说明：**
> 更多详细云端录制费用说明和计费示例，请参见 云端录制计费说明 。

文件存储费用

录制出的视频文件存放于云点播 VOD 或对象存储 COS 服务，由于存储本身会产生磁盘资源的消耗，因此需要按照存储的资源占用进行收费。存储的时间越久费用也就越高，因此如无特殊需要，您可以将文件的存储时间设置的短一些来节省费用，或者将文件存放在自己的服务器上。云点播VOD存储费用可以选择 视频存储（日结）价格 进行日结计算，也可以购买 存储资源包；对象存储COS存储费用说明请见 按量计费（后付费）。

> **注意：**
> 特别说明：若您选择存储至对象存储 COS 将会收取录制文件投递至 COS 的费用，详见 投递费用说明 ，存储至 VOD 将不收取此项费用。

文件播放费用

如果您录制的视频文件要被用于回看播放，会使用云点播或对象存储的 CDN 播放功能。由于观看本身会产生 CDN 流量消耗，因此需要按照云点播或对象存储的价格进行计费，默认按流量收费。观看的人数越多费用越高，云点播播放费用可以选择 按量计费 进行日结或月结计算，也可以购买 流量套餐包 。对象存储 COS 播放费用请见 流量费用说明 。

SDK 播放授权

音视频通话（TRTC）全功能版本 SDK 提供了功能全面性能强大的视频播放能力，可轻松配合云点播 VOD 或对象存储 COS 实现视频播放功能。移动端 SDK 在10.1及以上的版本可通过获取指定 License 以解锁视频播放能力。

> ⚠ **注意：**
> TRTC 的音视频通话、直播的播放能力无需 License 授权。

您可直接 购买播放器 License，或通过 购买的云点播流量包 免费获赠播放器 License 或 短视频 License，两种 License 均可用于解锁 SDK 的视频播放功能。并且点播资源包可以抵扣云点播的播放产生的日结流量，详细说明请参见 云点播预付费资源包 。
License 计费说明参见 腾讯云视立方 License，License 购买完成后可参考 License 操作指引 进行新增和续期等操作。

实践教程

为了保障录制的高可用，建议客户在集成 RESTful API 的同时注意以下几点。

- 调用 CreateCloudRecording 请求后，请关注 HTTP response，如果请求失败，那么需要根据具体的状态码采取相应的重试策略。错误码是由“一级错误码”和“二级错误码”组合而成，例如：`InvalidParameter.SdkAppId` 。
  - 如果返回的 Code 是 `InvalidParameter.xxxxx`，说明输入的参数有误，请根据提示检查参数。
  - 如果返回的 Code 是 `InternalError.xxxxx`，说明遇到了服务端错误，可以使用相同的参数重试多次，直到返回正常，拿到 taskid 为止。建议使用退避重试策略，如第一次3s重试，第二次6s重试，第三次12s重试，以此类推。
  - 如果返回的 Code 是 `FailedOperation.RestrictedConcurrency`，说明客户的并发录制任务数，超过了后台预留的资源（默认是500路），请联系腾讯云技术支持来调整最高并发路数限制。
- 如果您有订阅录制回调，当收到 EVENT_TYPE_CLOUD_RECORDING_RECORDER_STOP 回调事件，LeaveCode 为500时，说明录制与主播数据长时间断开连接，请再次发起录制任务保证录制的可用性。
- 调用 CreateCloudRecording 接口时，指定的 UserId/UserSig 是录制作为单独的机器人用户加入房间的 ID，请不要和 TRTC 房间内的其他用户重复。同时，TRTC 客户端加入的房间类型必须和录制接口指定的房间类型保持一致，比如 SDK 创建房间用的是字符串房间号，那么云端录制的房间类型也需要相应设置成字符串房间号。
- 录制状态查询，客户可以通过以下几种方式来得到录制相应的文件信息：
  - 成功发起 CreateCloudRecording 任务后15s左右，调用 DescribeCloudRecording 接口查询录制文件对应的信息，如果查询到状态为 idle 说明录制没有拉到上行的音视频流，请检查房间内是否有主播上行。
  - 成功发起 CreateCloudRecording 后，在确保房间有上行音视频的情况下，可以按照录制文件名的生成规则来拼接录制文件名称。具体文件名规则请参见 录制文件名命名规则 。
  - 录制文件的状态会通过回调发送到客户的服务器，如果订阅了相关回调，将会收到录制文件的状态信息。具体回调信息请参见 回调接口 。

- 录制用户（UserId）的 UserSig 过期时间应该设置成比录制任务生命周期更长的时间，防止录制任务机器断网，在内部高可用生效的时候，恢复录制因为 UserSig 过期而失败。

TRTC 云端录制，API 请求频率限制实践教程

腾讯云 API 服务对每个用户的请求频率设置了上限，以保障系统稳定性和资源公平分配。当用户请求频率超过预设阈值时，系统会返回频率限制错误。默认录制接口的 qps 为20次/秒。可通过 提交工单 报备，来联系提升接口 qps。一般情况下，qps 的设置值与在线报备的最高并发的比值是1:20，例如2000路并发在线的录制任务，可以提升 qps 到100，实际需要请根据业务实现方式进行评估，也可以选择报备腾讯相关人员进行评估。
如遇到限频错误，短期方案，可以按照以下方法，尝试快速调整：

- 降低请求频率至限制范围内。
- 业务实现请求队列。
- 添加适当的请求间隔时间。

长期方案可以按照以下方法进行调整：

- 实现指数退避重试机制，如第一次3s重试，第二次6s重试，第三次12s重试，以此类推直到重试成功。
- 优化业务逻辑，录制提前进房等待，减少并发 API 调用。

具体限频报错的示例，如下：

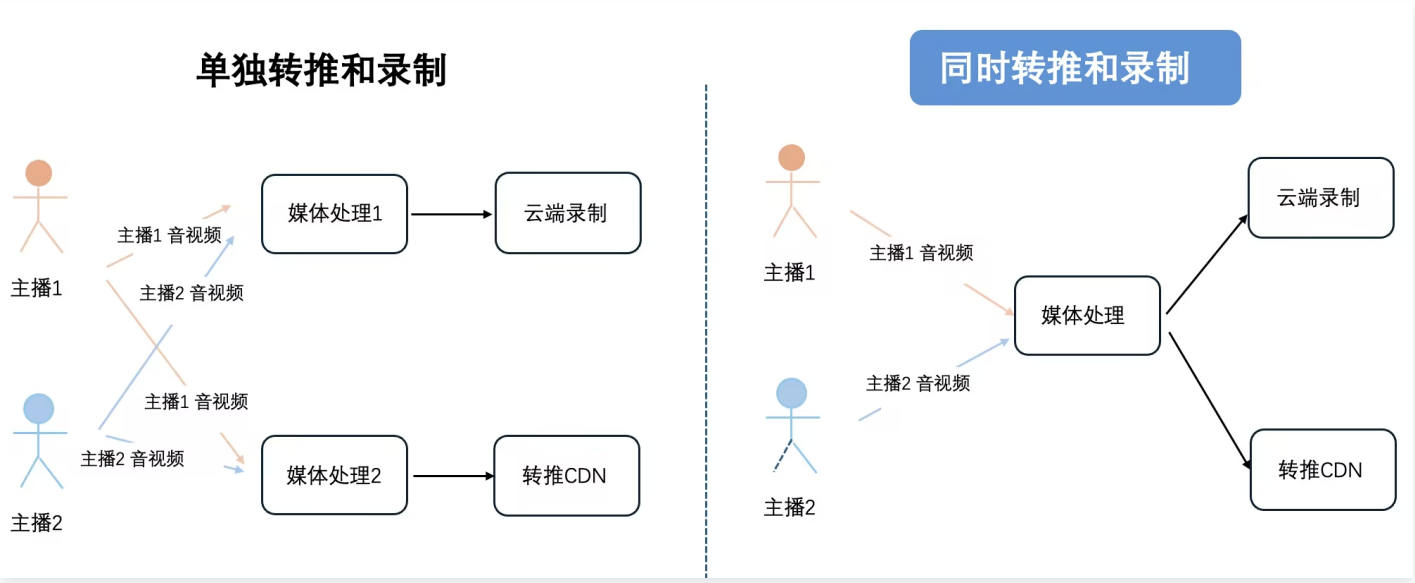
```
{
    "Response": {
        "Error": {
            "Code": "RequestLimitExceeded",
            "Message": "您当前每秒请求21次，超过每秒频率上限20，请稍后重试。"
        },
        "RequestId": "79c5cdbc-8a97-4d8b-be91-5c5e9143ad93"
    }
}
```

同时发起云端录制与转推

最近更新时间：2024-11-12 11:58:52

功能说明

针对在同一房间内同时需要录制和转推主播的音视频流的情况，TRTC 推出了一项全新解决方案。该方案支持通过一次接口调用，将房间内主播的音视频流同时进行录制存储和转推 CDN。另外对于需要混流录制和混流转推 CDN 的场景，用户只需完成一次混流任务，与分别发起混流录制和混流转推功能相比，只需一次接口调用并可节省一次混流的成本。



录制文件命名和文件切分说明

录制 MP4/AAC 文件名命名规则

- **单流录制 MP4/AAC 文件名规则：** `<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Index>.mp4/aac`
- **混流录制 MP4/AAC 文件名规则：** `<SdkAppId>_<RoomId>_<Index>.mp4/aac`

录制 HLS 文件名命名规则

- **单流录制 HLS 文件名规则：** `<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Type>.m3u8`
- **混流录制 HLS 文件名规则：** `<SdkAppId>_<RoomId>.m3u8`
- **字段含义说明：**

字段	含义

<SdkAppId>	录制任务的 SdkAppId
<RoomId>	录制的房间号，如果这里 RoomId 如果是字符串房间号，我们会对房间号先做 base64 操作，再把 base64 后的字符串中符号'/'替换成 '-'（中划线），符号'='替换成 '.'
<UserId>	录制的用户 ID，UserId 会先做 base64 操作，再把base64后的字符串中符号'/'替换成 '-'（中划线），符号'='替换成 '.'
<MediaId>	主辅流标识，main 代表主流（摄像头），aux 代表辅流（屏幕分享）
<Index>	如果没有触发切片逻辑（大小超过2GB或超过设置的切片时长）则无该字段，否则为切片的索引号，从1开始递增
<Type>	录制文件流类型，audio/video/audiovideo

说明：
自定义设置文件名称前缀：使用 API 录制 存储至云点播 VOD 时，可通过 TencentVod 中的 UserDefineRecordId 参数自定义文件名称前缀，前缀与默认录制文件名之间用 "__UserDefine_u_" 分隔。

录制文件切分说明

- **录制 MP4/ACC 文件切分的条件：**
  - 录制切分时长可设置范围1 – 1440分钟，默认1440分钟。
  - 单个 MP4/AAC 文件大小达到2GB。
- **录制 HLS 文件切分的条件：**
  - 录制任务持续时间超过14天时，HLS 文件将会被切分。

录制上传云存储说明

录制后台会在录制结束后将录制的文件通过您指定的方式上传到云存储平台（云点播 VOD 或对象存储 COS），并通过回调的形式把播放地址发送给您。如果录制模式为单流录制模式，每一个订阅的主播都会有一个对应的播放地址；如果发起混流录制，只有一个混合后媒体的播放地址。
通过 API 发起录制：在 McuRecordParams 中必须指定 McuStorageParams 的参数（云点播 VOD 或对象存储 COS），请确保已经开通对应的云存储服务且未欠费。

注意：
文件录制后会上传至您指定云存储平台（云点播 VOD 或对象存储 COS），为确保录制文件成功，请确保您指定的云点播 VOD 或对象存储 COS 服务可用。

API 接口和录制并发限制

- 接口的调用频率限制为20qps（如需提高 QPS 请 提交工单 ）。
- 单个接口超时时间为6秒。
- 单个应用下**默认并发录制支持500路（与 云端录制 共用）**，超过并发限制的任务会失败，如需更多并发路数，请 提交工单 联系我们。
- 单次录制任务最大支持同时订阅的房间内主播数为25个，主播只上行音频也会单独占据一路。

使用说明

目前支持通过 Restful API 接口：**启动转推任务** StartPublishCdnStream 同时设置录制相关参数（RecordParams）和转推相关参数（PublishCdnParams.N）即可同时发起录制与转推功能。此接口也可单独发起录制或转推功能。
针对录制参数的设置，提供以下两种设置方式：

录制参数设置方式	说明
Restful API 参数指定	如需灵活调整录制参数，可通过 RecordParams 来设置录制参数，此时 **RecordParams -> UniRecord 需设置为3。**
读取控制台录制模板	如录制参数相对固定，可通过在 控制台 > 应用管理 > 录制管理 下的配置录制模板，根据模板中的录制参数来发起录制，此时 **RecordParams -> UniRecord 需设置为2。** **⚠ 注意：** 通过控制台获取的录制参数时，仅使用您配置的单流录制参数，控制台录制模板中的混流参数不生效，如果发起混流，请通过 StartPublishCdnStream ->AudioParams 和 VideoParams 进行指定。

说明：

- 发起录制任务接口中需要您指定分配录制机器人的进房参数 UserId 和 UserSig（如何获取 UserSig），请不要与您房间内的正常主播或观众使用的 UserId 重复且不可与正在录制中的房间内指定的录制机器人 UserId 一致，否则会导致录制任务失败。
- **手动录制下，您可以前往控制台配置回调地址，以接受录制回调事件**，请见 录制回调说明 。

事件回调

我们针对云端录制和转推功能功能提供了多种的回调事件，帮助您及时了解任务的处理和完成情况，录制回调地址配置和事件说明请见 云端录制回调 和 旁路转推回调 。

录制文件管理

结束录制任务后，TRTC 录制系统中录制下来的文件上传至您指定的云存储平台（云点播 VOD 或对象存储 COS），详细说明请见 录制文件管理 。

相关费用

同时发起转推和录制功能，会产生 TRTC 云端录制费用 、TRTC 转推费用 、根据您指定存储云存储不同（云点播 VOD 或对象存储 COS）会由存储方收取回放文件存储与处理、播放费用。

实践教程

录制场景

根据录制需求场景的不同，我们提供以下实践说明：
场景一：在转推、媒体处理（混流、转码，加水印等）同时将对应结果同时进行录制下来。

场景二：续录多条流，即将不同时刻的多个主播画面录制到同一个文件中，实现续录。

场景一：录制 + 转推

当您需要对主播流进行转推，或者需要对上行流进行处理，同时有录制的需求时，可以使用转推录制接口，利用转推接口的流处理能力，对流进行灵活处理后进行录制。这里以云 API 为例介绍的转推录制使用方式（后续会支持终端 SDK 接口发起方式）。

1. “录制&转推&混流&水印”任务同时发起。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StartPublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
    "RoomId": "350",
    "RoomIdType": 0,
    "AgentParams": {
        "UserId": "trtc_partner_test_1",
        "UserSig": "********",
        "MaxIdleTime": 10
    },
    "WithTranscoding": 1,
    "AudioParams": {
        "AudioEncode": {
            "Codec": 0,
            "SampleRate": 48000,
            "Channel": 2,
            "BitRate": 64
        },
        "SubscribeAudioList": [
            {
                "UserInfo": {
                    "RoomId": "350",
                    "RoomIdType": 0,
                    "UserId": "Trtc_User_1"
                }
```

```
            },
            {
                "UserInfo": {
                    "RoomId": "350",
                    "RoomIdType": 0,
                    "UserId": "Trtc_User_0"
                }
            }
        ]
    },
    "VideoParams": {
        "VideoEncode": {
            "Width": 1280,
            "Height": 720,
            "Fps": 15,
            "BitRate": 512,
            "Gop": 2
        },
        "LayoutParams": {
            "PureAudioHoldPlaceMode": 0,
            "MixLayoutMode": 1
        },
        "BackGroundColor": "0xFF0000",
        "WaterMarkList": [
            {
                "WaterMarkType": 0,
                "WaterMarkImage": {
                    "LocationX": 64,
                    "LocationY": 64,
                    "WaterMarkHeight": 64,
                    "WaterMarkWidth": 128,
                    "WaterMarkUrl": "https://xxxxxxxx.png",
                    "ZOrder": 3
                }
            }
        ]
    },
    "RecordParams": {
        "UniRecord": 3,
        "RecordFormat": ["mp4"],
```

```
        "StreamType": 0,
        "McuStorageParams": {
            "McuCloudVod": {
                "McuTencentVod": {
                    "ExpireTime": 86400
                }
            }
        }
    },
    "PublishCdnParams": [
        {
            "IsTencentCdn": 1,
            "PublishCdnUrl": "rtmp://xxxxxx"
        }
    ]
}
```

2. 使用 UpdatePublishCdnStream 将其切换为单流录制 + 转推任务。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: UpdatePublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
    "TaskId":
"D0xMnLdRsmIO5+E09Y5wpUh+Q556USkfaMtHzbFuHc19muIw84b4EN1Bc0stJtznupVpR
Keyjc-0nDjD8V+HfU8A",
    "SequenceNumber": 5,
    "WithTranscoding": 0,
    "SingleSubscribeParams": {
        "UserMediaStream": {
            "UserInfo": {
                "RoomId": "350",
                "RoomIdType": 0,
                "UserId": "Trtc_User_0"
```

```
            },
            "StreamType": 0
        }
    }
}
```

3. 结束转推录制任务。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StopPublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
    "TaskId":
"D0xMnLdRsmIO5+E09Y5wpUh+Q556USkfaMtHzbFuHc19muIw84b4EN1Bc0stJtznupVpR
Keyjc-0nDjD8V+HfU8A"
}
```

场景二：续录多条流

为了实现多个任务的续录，这些任务发起时需要填写相同的 recordkey，并且前一个任务结束后，后一个任务要在指定的续录时间（通过控制台录制模板和Restful API 参数 RecordWaitTime 指定）内发起。

> **注意：**
> 若不需实现续录多条流，请勿将在发起任务时填写相同的 recordkey。

1. 首先发起录制任务 Task1。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StartPublishCdnStream
<公共请求参数>
```

```
{
    "SdkAppId": ********,
    "RoomId": "350",
    "RoomIdType": 0,
    "AgentParams": {
        "UserId": "trtc_partner_test_1",
        "UserSig": "********",
        "MaxIdleTime": 10
    },
    "WithTranscoding": 1,
    "AudioParams": {
        "AudioEncode": {
            "Codec": 0,
            "SampleRate": 48000,
            "Channel": 2,
            "BitRate": 64
        },
        "SubscribeAudioList": [
            {
                "UserInfo": {
                    "RoomId": "350",
                    "RoomIdType": 0,
                    "UserId": "Trtc_User_1"
                }
            },
            {
                "UserInfo": {
                    "RoomId": "350",
                    "RoomIdType": 0,
                    "UserId": "Trtc_User_0"
                }
            }
        ]
    },
    "VideoParams": {
        "VideoEncode": {
            "Width": 1280,
            "Height": 720,
            "Fps": 15,
            "BitRate": 512,
```

```
            "Gop": 2
        },
        "LayoutParams": {
            "PureAudioHoldPlaceMode": 0,
            "MixLayoutMode": 1
        }
    },
    "RecordParams": {
        "UniRecord": 3,
        "RecordFormat": ["mp4"],
        "StreamType": 0,
        "RecordKey": "recordkey2403122301",
        "McuStorageParams": {
            "McuCloudVod": {
                "McuTencentVod": {
                    "ExpireTime": 86400
                }
            }
        }
    },
    "PublishCdnParams": [
        {
            "IsTencentCdn": 1,
            "PublishCdnUrl": "rtmp:xxxxxxx"
        }
    ]
}
```

2. 结束录制任务 Task1，结束时不指定 recordkey。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StopPublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
```

```
    "TaskId":
"D0xMgKlRssqyrNUNqIPkpuwPuC1GLuIfSZN4zbFuHc19muIw84b4EN1Bc0stJtznufPhu
idu6JHPSpuljN"
}
```

3. 使用同样的 recordkey 发起续录任务 Task2。

> **说明：**
> RecordParams 中的参数以第一次任务为准，后续续录任务指定的 RecordParams 参数不生效，所以录制文件的前缀还是"prefix_test"，而不是“prefix_changed”

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StartPublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
    "RoomId": "350",
    "RoomIdType": 0,
    "AgentParams": {
        "UserId": "trtc_partner_test_1",
        "UserSig": "********",
        "MaxIdleTime": 10
    },
    "WithTranscoding": 0,
    "AudioParams": {
        "AudioEncode": {
            "Codec": 0,
            "SampleRate": 48000,
            "Channel": 2,
            "BitRate": 64
        }
    },
    "VideoParams": {
        "VideoEncode": {
            "Width": 1280,
```

```
            "Height": 720,
            "Fps": 15,
            "BitRate": 512,
            "Gop": 2
        }
    },
    "SingleSubscribeParams": {
        "UserMediaStream": {
            "UserInfo": {
                "RoomId": "350",
                "RoomIdType": 0,
                "UserId": "Trtc_User_0"
            },
            "StreamType": 0
        }
    },
    "RecordParams": {
        "RecordKey": "recordkey2403122301",
        "UniRecord": 3,
        "RecordFormat": ["mp4"],
        "StreamType": 0,
        "McuStorageParams": {
            "McuCloudVod": {
                "McuTencentVod": {
                    "ExpireTime": 86400
                }
            }
        }
    },
    "PublishCdnParams": [
        {
            "IsTencentCdn": 1,
            "PublishCdnUrl": "rtmp://xxxxxxx"
        }
    ]
}
```

4. 指定 recordkey，停止录制任务 Task2。

```
POST / HTTP/1.1
Host: trtc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: StopPublishCdnStream
<公共请求参数>

{
    "SdkAppId": ********,
    "RecordKey": "recordkey2403122301",
    "TaskId":
"D0zNfGlRsohMM0sTZo5hlGZD4gkBQp4fCZh4zbFuHc19muIw84b4EN1Bc0stJtznuZDTg
fDCsYnCozJw"
}
```

说明：

1. 带 recordkey A，发起任务 A；带 recordkey A，发起任务 B，录制会从录任务 A 切换到录任务 B，虽然任务 A 没退出，但不会再录制。
2. 转推录制的参数以第一次任务发起为准，后续续录任务指定的录制参数（RecordParams 中的参数）不生效。
3. 如果转推录制发起时指定了 recordkey，结束转推录制分几种情况：

   3.1 结束转推录制指定同样的 recordkey：

   - 如果此时 recordkey 对应的录制任务正在录制当前流，则转推和录制都立即结束，续录时间不生效。
   - 如果此时 recordkey 对应的录制任务没有录制当前流，则转推立即结束，录制任务不受影响继续进行。如果想要结束录制，需要通过结束录制当前正在录制的转推任务实现。

   补充说明：例如使用 recordkey A 先发起转推任务 1，然后用同样的 recordkey A 发起了转推任务 2，此时 recordkey A 对应的录制任务会切换到转推任务 2进行录制；当停止转推任务 1 时即使填写了 recordkey A，但是因为当前 recordkey A 已经不再录制 转推任务 1，因此无法将录制任务 A 结束；只有在结束 转推任务 2 时才能调用结束任务接口来结束录制任务 A。

   3.2 结束转推录制不指定 recordkey：则当前转推结束，录制任务等待续录时间后结束；续录时间内使用同样的 recordkey 发起转推录制任务，则两次转推续录到一个文件中。

监听服务端事件回调
房间与媒体回调

最近更新时间：2025-01-06 17:22:42

事件回调服务支持将实时音视频业务下的事件，以 HTTP/HTTPS 请求的形式通知到您的服务器。事件回调服务已集成房间事件组（Room Event）和媒体事件组（Media Event），您可以向腾讯云提供相关的配置信息来开通该服务。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

注意：

您需要提前准备以下信息：

- **必要项**：接收回调通知的 HTTP/HTTPS 服务器地址。
- **可选项**：计算签名的密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：

- **字符编码格式**：UTF-8。
- **请求**：body 格式为 JSON。
- **应答**：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。

> ⚠ **注意：**
> - 您的 HTTP 应答包长（header+body）需要控制在2000个字节以内。
> - POST 请求的 JSON 包体不会删减已有字段，但会根据业务需求添加新字段。在集成回调时，您需要适应这些新增字段的情况。

参数说明

回调消息参数

- 事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	sdk application id

- 事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组 ID
EventType	Number	回调通知的事件类型
CallbackTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒
EventInfo	JSON Object	事件信息

事件组 ID

字段名	值	含义

EVENT_GROUP_ROOM	1	房间事件组
EVENT_GROUP_MEDIA	2	媒体事件组

说明：
录制事件组相关说明请参见 实现云端录制与回放 。

事件类型

字段名	值	含义
EVENT_TYPE_CREATE_ROOM	101	创建房间
EVENT_TYPE_DISMISS_ROOM	102	解散房间
EVENT_TYPE_ENTER_ROOM	103	进入房间
EVENT_TYPE_EXIT_ROOM	104	退出房间
EVENT_TYPE_CHANGE_ROLE	105	切换角色
EVENT_TYPE_START_VIDEO	201	开始推送视频数据
EVENT_TYPE_STOP_VIDEO	202	停止推送视频数据
EVENT_TYPE_START_AUDIO	203	开始推送音频数据
EVENT_TYPE_STOP_AUDIO	204	停止推送音频数据
EVENT_TYPE_START_ASSIT	205	开始推送辅路数据
EVENT_TYPE_STOP_ASSIT	206	停止推送辅路数据

> **注意：**
> 退出房间只会回调104事件，不会回调202跟204事件。104事件相当于包含了202和204事件。手动关闭视频/音频，才会回调202/204事件。

事件回调示例

101

```
{    "EventGroupId": 1,
     "EventType":    101,
     "CallbackTs":   1687770730166,
     "EventInfo":    {
         "RoomId":   12345,
         "EventTs":  1687770730,
         "EventMsTs":    1687770730160,
         "UserId":   "test"
         }
}
```

102

```
{
     "EventGroupId": 1,
     "EventType":    102,
     "CallbackTs":   1687771618531,
     "EventInfo":    {
         "RoomId":   "12345",
         "EventTs":  1687771618,
         "EventMsTs":    1687771618457
      }
}
```

103

```
{
     "EventGroupId": 1,
     "EventType":    103,
```

```
    "CallbackTs":   1687770731932,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687770731,
        "EventMsTs":    1687770731831,
        "UserId":   "test",
        "Role": 21,
        "TerminalType": 2,
        "UserType": 3,
        "Reason":   1
    }
}
```

104

```
{
    "EventGroupId": 1,
    "EventType":    104,
    "CallbackTs":   1687770731922,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687770731,
        "EventMsTs":    1687770731898,
        "UserId":   "test",
        "Role": 20,
        "Reason":   1
    }
}
```

105

```
{
    "EventGroupId": 1,
    "EventType":    105,
    "CallbackTs":   1687772245596,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687772245,
```

```
        "EventMsTs":    1687772245537,
        "UserId":   "test",
        "Role": 21
    }
}
```

201

```
{
    "EventGroupId": 2,
    "EventType":    201,
    "CallbackTs":   1687771803198,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687771803,
        "EventMsTs":    1687771803192,
        "UserId":   "test"
    }
}
```

202

```
{
    "EventGroupId": 2,
    "EventType":    202,
    "CallbackTs":   1687771919458,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687771919,
        "EventMsTs":    1687771919447,
        "UserId":   "test",
        "Reason":   0
    }
}
```

203

```
{
    "EventGroupId": 2,
    "EventType":    203,
    "CallbackTs":   1687771869377,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687771869,
        "EventMsTs":    1687771869365,
        "UserId":   "test"
    }
}
```


204


```
{
    "EventGroupId": 2,
    "EventType":    204,
    "CallbackTs":   1687770732498,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687770732,
        "EventMsTs":    1687770732383,
        "UserId":   "test",
        "Reason":   0
    }
}
```


205


```
{
    "EventGroupId": 2,
    "EventType":    205,
    "CallbackTs":   1687772013823,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687772013,
        "EventMsTs":    1687772013753,
        "UserId":   "test"
```

```
    }
}
```

206

```
{
    "EventGroupId": 2,
    "EventType":    206,
    "CallbackTs":   1687772015054,
    "EventInfo":    {
        "RoomId":   12345,
        "EventTs":  1687772015,
        "EventMsTs":    1687772015032,
        "UserId":   "test",
        "Reason":   0
    }
}
```

事件信息

字段名	类型	含义
RoomId	String/Number	房间名（类型与客户端房间号类型一致）
EventTs	Number	事件发生的 Unix 时间戳，单位为秒（兼容保留）
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒
UserId	String	用户 ID
UniqueId	Number	唯一标识符（option：房间事件组携带） 当客户端发生了一些特殊行为，例如切换网络、进程异常退出及重进等，此时您的回调服务器可能会收到同一个用户多次进房和退房回调，UniqueId 可用于标识用户的同一次进退房
Role	Number	角色类型（option：进退房时携带）
TerminalType	Number	终端类型（option：进房时携带）
UserType	Number	用户类型（option：进房时携带）
Reason	Number	具体原因（option：进退房、停止媒体流时携带）

ClientIpv4	String	客户端 Ipv4 地址（option：使用Ipv4进房时，103事件携带）
ClientIpv6	String	客户端 Ipv6 地址（option：使用Ipv6进房时，103事件携带）

注意：
我们已发布“过滤客户端特殊行为导致的重复回调”策略。如果您是2021年07月30日之后接入回调服务，默认走新策略，房间事件组不再携带 UniqueId（唯一标识符）。

角色类型

字段名	值	含义
MEMBER_TRTC_ANCHOR	20	主播
MEMBER_TRTC_VIEWER	21	观众

终端类型

字段名	值	含义
TERMINAL_TYPE_WINDOWS	1	Windows 端
TERMINAL_TYPE_ANDROID	2	Android 端
TERMINAL_TYPE_IOS	3	iOS 端
TERMINAL_TYPE_LINUX	4	Linux 端
TERMINAL_TYPE_OTHER	100	其他

用户类型

字段名	值	含义
USER_TYPE_WEBRTC	1	webrtc

USER_TYPE_APPLET	2	小程序
USER_TYPE_NATIVE_SDK	3	Native SDK

具体原因

字段名	含义
进房	1：正常进房 2：切换网络 3：超时重试 4：跨房连麦进房
退房	1：正常退房 2：超时离开 3：房间用户被移出 4：取消连麦退房 5：强杀 **注意：Android 系统无法捕捉进程被强制终止，只能等待后台超时离开，此时回调 reason 为 2**
停止媒体流	0：正常停止 1：超时停止 **注意：后台连续30秒(默认)没有收到媒体流，将回调停止媒体流事件，并携带 Reason 为1，表示超时停止。**

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

注意：

body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t103,\n\t\"CallbackTs
\":\t1615554923704,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t12345,\
n\t\t\"EventTs\":\t1608441737,\n\t\t\"UserId\":\t\"test\",\n\t\t\"
```

```
UniqueId\":\t1615554922656,\n\t\t\"Role\":\t20,\n\t\t\"Reason\":\t
1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#   key：控制台配置的密钥key
//#   body：腾讯云回调返回的body体
//#   sign：腾讯云回调返回的签名值sign
//# 返回值：
//#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#   Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
```

```
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
```

```
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
    }

    public function genEventSig() {
```

```
        return $this->__hmacsha256();
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
```

```
}
```

旁路转推回调

最近更新时间：2023-07-31 17:32:42

服务端转推回调支持将您使用旁路转推 REST API 产生转推 CDN 的事件，以 HTTP/HTTPS 请求的形式通知到您的服务器。您可以向腾讯云提供相关的配置信息来开通该服务。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

> **注意：**
> 您需要提前准备以下信息：
> - **必要项**：接收回调通知的 HTTP/HTTPS 服务器地址。
> - **可选项**：计算签名的 密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：

- **字符编码格式**：UTF-8。
- **请求**：body 格式为 JSON。
- **应答**：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。
- **包体示例**：下述为“转推时间组-CDN 推流正在进行”事件的包体示例。

```
{
    "EventGroupId": 4,
    "EventType": 401,
    "CallbackTs": 1622186275913,
    "EventInfo": {
        "RoomId": "xx",
        "RoomType": 1,
        "EventTsMs": 1622186275913,
        "UserId": "xx",
```

```
        "TaskId": "xx",
        "Payload": {
            "Url": "rtmp://tencent-url/xxxx"
            "Status": 2                          /表示该转推任务正在向腾讯云CDN推流/
        }
    }
}
```

参数说明

回调消息参数

- 事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	sdk application id

- 事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组ID，混流转推事件固定为4
EventType	Number	回调通知的事件类型
CallbackTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒
EventInfo	JSON Object	事件信息

事件组 ID

字段名	值	含义
EVENT_GROUP_CLOUD_PUBLISH	4	转推事件组

事件类型

字段名	值	含义
EVENT_TYPE_CLOUD_PUBLISH_CDN_STATUS	401	云端转推 CDN 状态回调

事件信息

字段名	类型	含义
RoomId	String	房间名（类型与客户端房间号类型一致）
RoomType	Number	0表示数字房间号，1表示字符串房间号
EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
UserId	String	发起任务时指定的伴生机器人的用户 ID（AgentParams.UserId）
TaskId	String	任务 ID
Payload	JSON Object	事件的详细信息

Payload（详细信息）

字段名	值	含义
Url	String	推流的目的 URL 地址
Status	Number	转推状态
ErrorCode	Number	错误码
ErrorMsg	String	错误信息

转推状态

字段名	值	含义	回调频率
PUBLISH_CDN_STREAM_STATE_IDLE	0	推流未开始或已结束	仅回调1次

PUBLISH_CDN_STREAM_STATE_CONNECTING	1	正在连接 TRTC 服务器和 CDN 服务器	每5秒回调1次，60秒超时后不再回调
PUBLISH_CDN_STREAM_STATE_RUNNING	2	CDN 推流正在进行	仅回调1次
PUBLISH_CDN_STREAM_STATE_RECOVERING	3	TRTC 服务器和 CDN 服务器推流中断，正在恢复	每5秒回调1次，60秒超时后不再回调
PUBLISH_CDN_STREAM_STATE_FAILURE	4	TRTC 服务器和 CDN 服务器推流中断，且恢复或连接超时	仅回调1次
PUBLISH_CDN_STREAM_STATE_DISCONNECTING	5	正在断开 TRTC 服务器和 CDN 服务器	仅回调1次

转推状态推荐处理

状态	处理方法
PUBLISH_CDN_STREAM_STATE_IDLE	表示 URL 移除成功，无需处理。
PUBLISH_CDN_STREAM_STATE_CONNECTING	• 表示 URL 正在连接中，每隔5s回调一次，直到连接成功回调 `PUBLISH_CDN_STREAM_STATE_RUNNING`，或者60s后回调 `PUBLISH_CDN_STREAM_STATE_FAILURE`。您可以在收到 `PUBLISH_CDN_STREAM_STATE_FAILURE` 的时候替换有问题的 URL，调用 `UpdatePublishCdnStream` 更新 Publish 参数。 • 如果您的业务对时间比较敏感，可以在收到2个或以上的 `PUBLISH_CDN_STREAM_STATE_CONNECTING` 回调后，替换有问题的 URL，调用 `UpdatePublishCdnStream` 更新 Publish 参数。
PUBLISH_CDN_STREAM_STATE_RUNNING	表示 URL 推流成功，无需处理。
PUBLISH_CDN_STREAM_STATE_RECOVERING	• 表示推流过程发生了中断，正在重连中，每隔5s回调一次，直到重连成功回调 `PUBLISH_CDN_STREAM_STATE_RUNNING`，或者60s后回调 `PUBLISH_CDN_STREAM_STATE_FAILURE`。通常为网络抖动，无需处理。

	• 如果 PUBLISH_CDN_STREAM_STATE_RECOVERING 和 PUBLISH_CDN_STREAM_STATE_RUNNING 短时间内交替出现，您需要检查下是否存在多任务使用相同的推流 URL 。
PUBLISH_CDN_STREAM_STATE_FAILURE	表示推流 URL ，在60s内建连失败或者恢复推流失败，此时您可以替换有问题的 URL ，调用 UpdatePublishCdnStream 更新 Publish 参数。
PUBLISH_CDN_STREAM_STATE_DISCONNECTING	表示，正在移除推流 URL ，移除成功后，会回调 PUBLISH_CDN_STREAM_STATE_IDLE，无需处理。

基本回调转移示例

- **发起转推/新增转推地址到转推成功的事件转移**
  PUBLISH_CDN_STREAM_STATE_CONNECTING -> PUBLISH_CDN_STREAM_STATE_RUNNING
- **停止转推/删除转推地址到停止转推成功的事件转移**
  PUBLISH_CDN_STREAM_STATE_RUNNING -> PUBLISH_CDN_STREAM_STATE_DISCONNECTING -> PUBLISH_CDN_STREAM_STATE_IDLE
- **转推过程中，链接失败到重试链接成功的事件转移**
  PUBLISH_CDN_STREAM_STATE_RUNNING -> PUBLISH_CDN_STREAM_STATE_RECOVERING -> PUBLISH_CDN_STREAM_STATE_RUNNING
- **转推过程中，链接失败到重试链接超时失败的事件转移**
  PUBLISH_CDN_STREAM_STATE_RUNNING -> PUBLISH_CDN_STREAM_STATE_RECOVERING -> PUBLISH_CDN_STREAM_STATE_FAILURE -> PUBLISH_CDN_STREAM_STATE_IDLE

> ⚠ **注意：**
> 推流回调有可能会乱序到达您的回调服务器，此时您需要根据 EventInfo 中的 EventMsTs 做事件排序，如果您只关心 URL 最新状态，可以忽略后续到达的过期事件。

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

> ⚠ **注意：**

body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t103,\n\t\"CallbackTs
\":\t1615554923704,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t12345,\
n\t\t\"EventTs\":\t1608441737,\n\t\t\"UserId\":\t\"test\",\n\t\t\"
UniqueId\":\t1615554922656,\n\t\t\"Role\":\t20,\n\t\t\"Reason\":\t
1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#   key：控制台配置的密钥key
//#   body：腾讯云回调返回的body体
//#   sign：腾讯云回调返回的签名值sign
//# 返回值：
//#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#   Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
```

```
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
```

```
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
```

```
    }

    public function genEventSig() {
        return $this->__hmacsha256();
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
```

```
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

云端录制和页面录制回调

最近更新时间：2025-07-16 16:33:52

本文针对 新版云端录制功能 和 页面录制功能 的回调事件进行具体说明，旧版云端录制的SdkAppId配置回调录制请见 旧版录制回调 。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

注意：

您需要提前准备以下信息并在控制台完成 回调配置 。

- **必要项**：接收回调通知的 HTTP/HTTPS 服务器地址。
- **可选项**：计算签名的密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

回调接口

您可以提供一个接收回调的 HTTP/HTTPS 服务网关来订阅回调消息。当相关事件发生时，云录制系统会回调事件通知到您的消息接收服务器。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：

- **字符编码格式**：UTF-8。
- **请求**：body 格式为 JSON。
- **应答**：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。
- **包体示例**：下述为事件类型是“305”的包体示例，其他事件类型的示例可参见下文具体的事件信息说明。

```
{
  "EventGroupId": 3,
  "EventType": 305,
  "CallbackTs": 1752580150082,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": 1752580150,
    "EventMsTs": 1752580150081,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0
    }
  }
}
```

云端录制参数说明

事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	SDK 应用 ID 值

事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组 ID， 云端录制固定为3，页面录制固定为8

EventType	Number	回调通知的事件类型
CallbackTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒
EventInfo	JSON Object	事件信息

事件类型说明：

字段名	类型	含义
EVENT_TYPE_CLOUD_RECORDING_RECORDER_START	301	云端录制模块启动
EVENT_TYPE_CLOUD_RECORDING_RECORDER_STOP	302	云端录制模块退出
EVENT_TYPE_CLOUD_RECORDING_UPLOAD_START	303	云端录制文件上传任务启动，仅在选择对象存储时回调
EVENT_TYPE_CLOUD_RECORDING_FILE_INFO	304	云端录制 生成 m3u8 索引文件，第一次生成并且上传成功后回调，仅在通过 API 录制 选择对象存储时回调
EVENT_TYPE_CLOUD_RECORDING_UPLOAD_STOP	305	云端录制文件上传结束，仅在通过 API 录制 选择对象存储时回调
EVENT_TYPE_CLOUD_RECORDING_FAILOVER	306	云端录制发生迁移，原有的录制任务被迁移到新负载上时触发
EVENT_TYPE_CLOUD_RECORDING_FILE_SLICE	307	云端录制 生成 m3u8 索引文件（切出第一个 ts 切片）生成后回调，仅在通过 API 录制 选择对象存储时回调
EVENT_TYPE_CLOUD_RECORDING_DOWNLOAD_IMAGE_ERROR	309	云端录制下载解码图片文件发生错误
EVENT_TYPE_CLOUD_RECORDING_MP4_STOP	310	云端录制 MP4 录制任务结束，仅在通过 API录制 选择对象存储时回调(控制台开启自动录制，选择授权给点播的 cos 作为存储时，请关注311事件)
EVENT_TYPE_CLOUD_RECORDING_VOD_COMMIT	311	云端录制 VOD 录制任务上传媒体资源完成，在选择**云点播**时和通过控制台**自动**

		录制存储至 cos 时回调（录制文件结束后携带点播索引信息，请订阅此类型回调事件）
EVENT_TYPE_CLOUD_RECORDING_VOD_STOP	312	云端录制 VOD 录制任务结束，仅在选择云点播时回调

> **注意：**
> 301 – 309区间的回调状态为实时录制的中间状态，可以更加清晰的知晓录制任务的进行过程并记录状态，实际录制文件上传到点播成功会回调311事件，整体任务结束回调312事件。

事件信息说明：

字段名	类型	含义
RoomId	String/Number	房间名（类型与客户端房间号类型一致）
EventTs	Number	事件发生的 Unix 时间戳，单位为秒 (不建议使用该字段，建议使用EventMsTs)
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒
UserId	String	录制机器人的用户 ID
TaskId	String	录制 ID，一次云端录制任务唯一的 ID
Payload	JsonObject	根据不同事件类型定义不同

- **事件类型为301**（EVENT_TYPE_CLOUD_RECORDING_RECORDER_START）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表录制模块启动成功 1：代表录制模块启动失败

```
{
    "EventGroupId": 3,
    "EventType": 301,
    "CallbackTs": 1622186275913,
    "EventInfo": {
        "RoomId": "635055",
```

```
        "EventTs": "1622186275",
        "EventMsTs": 1622186275757,
        "UserId": "user1",
        "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-
gQvQE",
        "Payload": {
            "Status": 0
        }
    }
}
```

- **事件类型为302（EVENT_TYPE_CLOUD_RECORDING_RECORDER_STOP）时 Payload 的定义：**

字段名	类型	含义
LeaveCode	Number	0：代表录制模块正常调用停止录制退出 1：录制机器人被客户踢出房间 2：客户解散房间 3：服务器将录制机器人踢出 4：服务器解散房间 99：代表房间内除了录制机器人没有其他用户流，超过指定时间退出 100：房间超时退出 101：同一用户重复进入相同房间导致机器人退出

```
{
  "EventGroupId": 3,
  "EventType": 302,
  "CallbackTs": 1622186354806,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": "1622186354",
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "LeaveCode": 0
    }
  }
}
```

- **事件类型为303**（EVENT_TYPE_CLOUD_RECORDING_UPLOAD_START）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表上传模块正常启动 1：代表上传模块初始化失败。

```
{
  "EventGroupId": 3,
  "EventType": 303,
  "CallbackTs": 1752580200339,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": 1752580146,
    "EventMsTs": 1752580146201,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0
    }
  }
}
```

- **事件类型为304**（EVENT_TYPE_CLOUD_RECORDING_FILE_INFO ）时 Payload 的定义：

字段名	类型	含义
FileList	String	生成的 M3U8 文件名

```
{
  "EventGroupId": 3,
  "EventType": 304,
  "CallbackTs": 1752580146476,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": 1752580146,
    "EventMsTs": 1752580146474,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
```

```
      "Payload": {
        "FileList":
"1400704311_635055__UserId_s_dHVmdGpk__UserId_e_main_audio.m3u8"
      }
    }
  }
```

- **事件类型为305**（EVENT_TYPE_CLOUD_RECORDING_UPLOAD_STOP）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表此次录制上传任务已经完成，所有的文件均已上传到指定的第三方云存储 1：代表此次录制上传任务已经完成，但至少有一片文件滞留在服务器或者备份存储上 2：代表滞留在服务器或者备份存储上的文件已经恢复上传到指定的第三方云存储 注意：305代表 HLS 文件上传结束事件

```
{
  "EventGroupId": 3,
  "EventType": 305,
  "CallbackTs": 1752580150082,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": 1752580150,
    "EventMsTs": 1752580150081,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0
    }
  }
}
```

- **事件类型为306**（EVENT_TYPE_CLOUD_RECORDING_FAILOVER）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表此次迁移已经完成

```
{
```

```
  "EventGroupId": 3,
  "EventType": 306,
  "CallbackTs": 1622191989674,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191989,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0
    }
  }
}
```

- **事件类型为307（EVENT_TYPE_CLOUD_RECORDING_FILE_SLICE）时 Payload 的定义：**

字段名	类型	含义
FileName	String	m3u8 文件名
UserId	String	录制文件对应的用户 ID
TrackType	String	音视频类型 audio/video/audio_video
BeginTimeStamp	String	录制开始时，服务器Unix时间戳(毫秒)

```
{
  "EventGroupId": 3,
  "EventType": 307,
  "CallbackTs": 1752580145235,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": "1752580145",
    "EventMsTs": 1752580145170,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "FileName":
"1400704311_635055__UserId_s_dHVmdGpk__UserId_e_main_audio.m3u8",
```

```
        "UserId": "tuftjd",
        "TrackType": "audio",
        "BeginTimeStamp": 1752580135170
      }
    }
}
```

- **事件类型为309**（EVENT_TYPE_CLOUD_RECORDING_DOWNLOAD_IMAGE_ERROR）时 Payload 的定义：

字段名	类型	含义
Url	String	下载失败的 URL

```
{
  "EventGroupId": 3,
  "EventType": 309,
  "CallbackTs": 1622191989674,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191989,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Url":
"https://query******1b35638507c58999a9e7077c7edc2738a896106f"
    }
  }
}
```

- **事件类型为310**（EVENT_TYPE_CLOUD_RECORDING_MP4_STOP）时 Payload 的定义：

> **说明：**
> 310 是上传 MP4 文件到客户指定第三方云存储 COS 完成后的回调事件，一个任务对应一个310事件（一个事件中对应本次任务中所有的录制文件信息）

段名	类型	含义

Status	Number	0：代表此次录制 MP4 任务已经正常退出，所有的文件均已上传到指定的第三方云存储 1：代表此次录制 MP4 任务已经正常退出，但至少有一片文件滞留在服务器或者备份存储上 2：代表此次录制 MP4 任务异常退出(可能原因是拉取 cos 的 hls 文件失败)
FileList	Array	所有生成的 MP4 文件名
FileMessage	Array	所有生成的 MP4 文件信息
FileName	String	MP4 文件名
UserId	String	MP4 文件对应的用户 ID（当录制模式为混流模式时，此字段为空）
TrackType	String	audio 音频 / video 纯视频 / audio_video 音视频
MediaId	String	主辅流标识，main 代表主流（摄像头），aux 代表辅流（屏幕分享），mix 代表混流录制
StartTimeStamp	Number	MP4 文件开始的 Unix 时间戳(毫秒)
EndTimeStamp	Number	MP4 文件结束的 Unix 时间戳(毫秒)

```
{
  "EventGroupId": 3,
  "EventType": 310,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "635055",
    "EventTs": 1622191989,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0,
      "FileList":
["1400704311_635055__UserId_s_dHVmdGpk__UserId_e_main.mp4"],
      "FileMessage": [
        {
```

```
            "FileName":
"1400704311_635055__UserId_s_dHVmdGpk__UserId_e_main.mp4",
            "UserId": "user2",
            "TrackType": "audio_video",
            "MediaId": "main",
            "StartTimeStamp": 1622186279145,
            "EndTimeStamp": 1622186282145
        }
    ]
  }
 }
}
```

- **事件类型为311**（EVENT_TYPE_CLOUD_RECORDING_VOD_COMMIT）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表本录制文件正常上传至点播平台 1：代表本录制文件滞留在服务器或者备份存储上 2：代表本录制文件上传点播任务异常
UserId	String	本录制文件对应的用户 ID（当录制模式为合流模式时，此字段为空）
TrackType	String	audio 音频 / video 纯视频 / audio_video 音视频
MediaId	String	主辅流标识，main 代表主流（摄像头），aux 代表辅流（屏幕分享），mix 代表混流录制
FileId	String	本录制文件在点播平台的唯一 ID
VideoUrl	String	本录制文件在点播平台的播放地址
CacheFile	String	本录制文件对应的 MP4/HLS 文件名
StartTimeStamp	Number	本录制文件开始的 UNIX 时间戳（毫秒）
EndTimeStamp	Number	本录制文件结束的 UNIX 时间戳（毫秒）
Errmsg	String	status 不为0时，对应的错误信息

上传成功的回调：

```
{
  "EventGroupId": 3,
  "EventType": 311,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0,
      "TencentVod": {
        "UserId": "user2",
        "TrackType": "audio_video",
        "MediaId": "main",
        "FileId": "5145403691953718116",
        "VideoUrl": "http://1500013742***/f0.mp4",
        "CacheFile":
"1400704311_635055__UserId_s_dHVmdGpk__UserId_e_main.mp4",
        "StartTimeStamp": 1622186279153,
        "EndTimeStamp": 1622186282153
      }
    }
  }
}
```


上传失败的回调：


```
{
  "EventGroupId": 3,
  "EventType": 311,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
```

```
    "Payload": {
      "Status": 1,
      "Errmsg": "vod service stop",
      "TencentVod": {
        "UserId": "user1",
        "TrackType": "audio_video",
        "CacheFile": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-
gQvQE"
      }
    }
  }
}
```

> **说明：**
> 录制完成收到311回调后到**文件上传完成**，根据您本次录制文件的大小不同可能需等待30s - 3min。

- **事件类型为312**（EVENT_TYPE_CLOUD_RECORDING_VOD_STOP）时 Payload 的定义：

字段名	类型	含义
Status	Number	0：代表本次上传 VOD 任务已经正常退出 1：代表本次上传 VOD 任务异常退出

```
{
  "EventGroupId": 3,
  "EventType": 312,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "EventMsTs": 1622186275757,
    "UserId": "user1",
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 0
    }
  }
}
```

页面录制参数说明

页面录制事件类型：

字段名	类型	含义
EVENT_TYPE_WEB_RECORDER_START	801	页面录制模块启动
EVENT_TYPE_WEB_RECORDER_STOP	802	页面录制模块退出
EVENT_TYPE_WEB_RECORDER_STATUS_UPDATE	803	页面录制模块录制状态更新
EVENT_TYPE_WEB_RECORDER_RESOURCE_LIMIT	804	页面录制任务资源受限，结束录制任务，仅在录制时长超限、录制分辨率超限时回调

页面录制事件信息说明：

字段名	类型	含义
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	录制 ID，一次页面录制任务唯一的 ID
Payload	JsonObject	根据不同事件类型定义不同
EventMessage	String	对应Status的事件信息描述

- **事件类型为801**（EVENT_TYPE_WEB_RECORDER_START）时 Payload 的定义：

字段名	类型	含义
Status	Number	1：代表页面录制模块启动成功 2：代表页面录制模块启动失败 3：代表录制过程中异常退出 4：代表录制任务已迁移 5：代表录制任务异常，停止录制
EventMessage	String	Success：页面录制模块启动成功，开始录制 StartRecording error：页面录制模块启动失败 Goto url timeout：录制页面访问超时

		Exception error, exit task：录制异常结束 Chrome exception, exit task：Chrome异常退出，录制结束 Recording tasks have been migrated：录制任务已迁移 Page load timeout：页面加载超时

```
{
  "EventGroupId": 8,
  "EventType": 801,
  "CallbackTs": 1622186275913,
  "EventInfo": {
    "EventMsTs": 1622186275757,
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 1,
      "EventMessage": "Success"
    }
  }
}
```

说明：

- 收到状态码为2的801回调时，请检查 RecordUrl 是否可以正常访问。
- 801回调的状态码3和状态码4，为页面录制的中间状态，便于录制任务出现异常时的排查；页面录制会自动对上述两种状态进行处理，客户无需进行处理。

注意：

收到状态码为5的801回调时，录制任务将被终止，并不再重试，请您及时联系业务人员进行排查。

- **事件类型为802**（EVENT_TYPE_WEB_RECORDER_STOP）时 Payload 的定义：

字段名	类型	含义
Status	Number	1：代表本次页面录制任务正常结束
EventMessage	String	Success：页面录制模块正常调用停止录制退出

```
{
  "EventGroupId": 8,
  "EventType": 802,
```

```
  "CallbackTs": 1622186275913,
  "EventInfo": {
    "EventMsTs": 1622186275757,
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 1,
      "EventMessage": "Success"
    }
  }
}
```

说明：
收到802回调说明页面录制任务已完成，实际录制文件上传完成会回调311事件，整体完成回调312事件。

- **事件类型为803**（EVENT_TYPE_WEB_RECORDER_STATUS_UPDATE）时 Payload 的定义：

字段名	类型	含义
Status	Number	1：代表页面录制任务页面刷新 2：代表页面录制任务暂停录制 3：代表页面录制任务恢复录制
EventMessage	String	PageRefresh：刷新录制页面 RecordPaused：录制任务暂停 RecordResume：录制任务恢复 Page load timeout：页面加载超时

```
{
  "EventGroupId": 8,
  "EventType": 803,
  "CallbackTs": 1622186275913,
  "EventInfo": {
    "EventMsTs": 1622186275757,
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 1,
      "EventMessage": "PageRefresh"
    }
  }
```

```
}
```

- **事件类型为804（EVENT_TYPE_WEB_RECORDER_RESOURCE_LIMIT）时 Status 的定义：**

字段名	类型	含义
Status	Number	1：代表本次页面录制任务达到设定的最大录制时间，结束录制 2：代表本次页面录制任务的录制页面中出现了超过设定的最大录制分辨率的视频流，结束录制
EventMessage	String	Over time limit：录制时长达到设定的最大录制时长，自动结束录制 Over resolution limit：录制任务中出现超过最大分辨率限制的视频源，自动结束录制

```
{
  "EventGroupId": 8,
  "EventType": 804,
  "CallbackTs": 1622186275913,
  "EventInfo": {
    "EventMsTs": 1622186275757,
    "TaskId": "-m9-bVVU7id***K-m928oZWQndiborbEWH3zY-lIXlprc-gQvQE",
    "Payload": {
      "Status": 1,
      "EventMessage": "Over time limit"
    }
  }
}
```

说明：
收到804回调状态码为2时，请检查录制过程中是否出现超出分辨率限制（1920*1080）的视频流。

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

注意：

body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t103,\n\t\"CallbackTs
\":\t1615554923704,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t12345,\
n\t\t\"EventTs\":\t1608441737,\n\t\t\"UserId\":\t\"test\",\n\t\t\"
UniqueId\":\t1615554922656,\n\t\t\"Role\":\t20,\n\t\t\"Reason\":\t
1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#   key：控制台配置的密钥key
//#   body：腾讯云回调返回的body体
//#   sign：腾讯云回调返回的签名值sign
//# 返回值：
//#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#   Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
```

```
        String key = "123654";
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
```

```
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
```

```
        return base64_encode( $hash);
    }

    public function genEventSig() {
        return $this->__hmacsha256();
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
```

```
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

输入在线媒体流回调

最近更新时间：2024-07-17 17:33:31

服务端输入在线媒体流回调支持将您使用 输入在线媒体流 REST API 产生输入在线媒体流的事件，以 HTTP/HTTPS 请求的形式通知到您的服务器。您可以向腾讯云提供相关的配置信息来开通该服务。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

> **注意：**
> 您需要提前准备以下信息：
> - **必要项**：接收回调通知的 HTTP/HTTPS 服务器地址。
> - **可选项**：计算签名的 密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：

- **字符编码格式**：UTF-8。
- **请求**：body 格式为 JSON。
- **应答**：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。
- **包体示例**：下述为“输入在线媒体流开始成功”事件的包体示例。

```
{
        "EventGroupId": 7,
        "EventType":    701,
        "CallbackMsTs":   1701937900012,
        "EventInfo":    {
                "EventMsTs": 1701937900013,
                "TaskId":"xx",
                "Status":0
        }
```

```
}
```

参数说明

回调消息参数

- 事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	sdk application id

- 事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组ID，混流转推事件固定为4
EventType	Number	回调通知的事件类型
CallbackMsTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒
EventInfo	JSON Object	事件信息

事件组 ID

字段名	值	含义
EVENT_GROUP_STREAM_INGEST	7	输入在线媒体流事件组

事件类型

字段名	值	含义
EVENT_TYPE_STREAM_INGEST_START	701	输入在线媒体流开始

EVENT_TYPE_STREAM_INGEST_STOP	702	输入在线媒体流停止

事件类型为（EVENT_TYPE_STREAM_INGEST_START 701）时事件信息的定义：

字段名	类型	含义
EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	输入在线媒体流任务 ID
Status	Number	输入在线媒体流开始状态

输入在线媒体流开始状态

字段名	值	含义	回调频率
STATUS_START_SUCCESS	0	输入在线媒体流开始成功	成功时回调1次
STATUS_START_FAILURE	1	输入在线媒体流开始失败	失败时回调一次
STATUS_START_AGAIN	2	输入在线媒体流再次开始	在0，1，3秒时重试，重试时回调

输入在线媒体流状态推荐处理

状态	处理方法
STATUS_START_SUCCESS	表示成功，无需处理。
STATUS_START_FAILURE	当您收到三个输入在线媒体流失败的状态，请检查源流URL，重新发起输入在线媒体流
STATUS_START_AGAIN	• 开始输入在线媒体流1分钟内收到：表示URL建连失败，或RTMP推流失败，系统自动触发重试，若最终失败请检查URL是否正常建立连接 • 开始输入在线媒体流1分钟后收到：可能是源流或后台网络抖动引起触发重新拉起，无需处理。

基本回调转移示例

- **输入在线媒体流失败/输入在线媒体流再次开始/输入在线媒体流开始成功的事件转移**
  `STATUS_START_FAILURE` -> `STATUS_START_AGAIN` -> `STATUS_START_SUCCESS`

> ⚠ **注意：**
> 输入在线媒体流回调有可能会乱序到达您的回调服务器，此时您需要根据 EventInfo 中的 EventMsTs 做事件排序，如果您只关心 URL 最新状态，可以忽略后续到达的过期事件。

事件类型为（EVENT_TYPE_STREAM_INGEST_STOP 702）时事件信息的定义：

字段名	类型	含义
EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	输入在线媒体流任务 ID
Status	Number	输入在线媒体流停止状态

输入在线媒体流停止状态

字段名	值	含义	回调频率
STATUS_STOP_SUCCESS	0	输入在线媒体流停止成功	成功时回调1次

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

> ⚠ **注意：**
> body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="{\n\t\"Ebody="
{\"EventGroupId\":7,\"EventType\":701,\"CallbackMsTs\":17019379000
12,\"EventInfo\":
{\"EventMsTs\":1701937900012,\"TaskId\":\"WMdqEeEgj2ksqnyUsuXC+qLk
VypGmwjrgh1JC6ZefVP+rvsidDnZsAw8uWgX0XRGvdSVfAMunise2kcZaefdgHvx3-
```

```
M2v6fmTjRNgg..\",\"Status\":0}}"ventGroupId\":\t1,\n\t\"EventType\
":\t103,\n\t\"CallbackTs\":\t1615554923704,\n\t\"EventInfo\":\t{\n
\t\t\"RoomId\":\t12345,\n\t\t\"EventTs\":\t1608441737,\n\t\t\"User
Id\":\t\"test\",\n\t\t\"UniqueId\":\t1615554922656,\n\t\t\"Role\":
\t20,\n\t\t\"Reason\":\t1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#   key：控制台配置的密钥key
//#   body：腾讯云回调返回的body体
//#   sign：腾讯云回调返回的签名值sign
//# 返回值：
//#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#   Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
```

```
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
```

```
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
    }

    public function genEventSig() {
        return $this->__hmacsha256();
```

```
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

AI 实时对话与语音转文字回调

最近更新时间：2024-11-04 19:44:42

本文用于介绍 AI 服务（ AI 实时对话 和 语音转文字 功能）相关的云 API 接口产生的事件，以 HTTP/HTTPS 请求的形式通知到您的服务器。您可以向腾讯云提供相关的配置信息来开通该服务。您也可以结合TRTC的 房间与媒体回调 使用，实现更多自定义逻辑。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

> **注意：**
> 您需要提前准备以下信息：
> - **必要项**：接收回调通知的 HTTP/HTTPS 服务器地址。
> - **可选项**：计算签名的 密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：

- **字符编码格式**：UTF-8。
- **请求**：body 格式为 JSON。
- **应答**：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。
- **包体示例**：下述为“启动AI对话任务成功”事件的包体示例。

```
{
    "EventGroupId":  9,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "hKPD2Q7kBVzu-
6ezFiqmcEBJQCykqbZrS9OOTE46uYlb4NvQDIaEXlpOlLXFtGBiado5oP0zfLDZs",
        "RoomId": "1234",
```

```
        "RoomIdType": 0,
        "Payload": {
            "Status": 0
        }
    }
}
```

参数说明

回调消息参数

- 事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	sdk application id

- 事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组 ID，混流转推事件固定为4
EventType	Number	回调通知的事件类型
CallbackMsTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒
EventInfo	JSON Object	事件信息

事件组 ID

字段名	值	含义
EVENT_GROUP_AI_SERVICE	9	AI 服务事件组

事件类型

字段名	值	含义
EVENT_TYPE_AI_SERVICE_START	901	AI 任务开始状态回调
EVENT_TYPE_AI_SERVICE_STOP	902	AI 任务结束状态回调
EVENT_TYPE_AI_SERVICE_MSG	903	回调完整的一句话。 • AI 实时对话：识别出完整的一句话后回调 • 语音转文字：回调转录的完整句子

事件类型为（EVENT_TYPE_AI_SERVICE_START 901）时事件信息的定义：

字段名	类型	含义
EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	AI 任务 ID
RoomId	String	TRTC 的房间 ID
RoomIdType	Integer	• 0：表示数字房间号 • 1：表示字符串房间号
Payload.Status	Number	• 0：启动 AI 任务成功 • 1：启动 AI 任务失败

```
{
    "EventGroupId": 9,
    "EventType": 901,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "xx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "Payload": {
            "Status": 0
        }
    }
}
```

事件类型为（EVENT_TYPE_AI_SERVICE_STOP 902）时事件信息的定义：

字段名	类型	含义
EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	AI 任务 ID
RoomId	String	TRTC 的房间 ID
RoomIdType	Integer	• 0：表示数字房间号 • 1：表示字符串房间号
Payload.LeaveCode	Integer	• 0：正常调用停止接口后任务退出 • 1：业务自己踢掉转录机器人后任务退出 • 2：业务自己解散房间后任务退出 • 3：TRTC 服务端踢掉机器人 • 4：TRTC 服务端解散房间 • 98：内部异常错误，建议业务进行重试 • 99：代表房间内除了转录机器人没有其他用户流，超过指定时间退出

```
{
    "EventGroupId": 9,
    "EventType": 902,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "xx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "Payload": {
            "LeaveCode": 0
        }
    }
}
```

事件类型为（EVENT_TYPE_AI_SERVICE_MSG 903）时事件信息的定义：

字段名	类型	含义

EventMsTs	String	事件发生的 Unix 时间戳，单位为毫秒
TaskId	String	AI 任务 ID
RoomId	String	TRTC 的房间 ID
RoomIdType	Integer	• 0：表示数字房间号 • 1：表示字符串房间号

Payload	JSON Object	为JSON对象，与客户端自定义消息回调格式一致
		{
		"UserId":"",
		"Text":"",
		"StartTimeMs":1234,
		"EndTimeMs":1269,
		"RoundId":"xxxxxx" // uuid
		}

```
{
    "EventGroupId":  9,
    "EventType":     903,
    "CallbackTs":    1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
       "TaskId": "xx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "Payload": {
            "UserId":"",
            "Text":"",
            "StartTimeMs":1234,
            "EndTimeMs":1269,
            "RoundId":"xxxxxx"
        }
    }
}
```

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

注意：

body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="{\n\t\"Ebody="
{\"EventGroupId\":7,\"EventType\":701,\"CallbackMsTs\":17019379000
12,\"EventInfo\":
{\"EventMsTs\":1701937900012,\"TaskId\":\"WMdqEeEgj2ksqnyUsuXC+qLk
VypGmwjrgh1JC6ZefVP+rvsidDnZsAw8uWgX0XRGvdSVfAMunise2kcZaefdgHvx3-
M2v6fmTjRNgg..\",\"Status\":0}}"ventGroupId\":\t1,\n\t\"EventType\
":\t103,\n\t\"CallbackTs\":\t1615554923704,\n\t\"EventInfo\":\t{\n
\t\t\"RoomId\":\t12345,\n\t\t\"EventTs\":\t1608441737,\n\t\t\"User
Id\":\t\"test\",\n\t\t\"UniqueId\":\t1615554922656,\n\t\t\"Role\":
\t20,\n\t\t\"Reason\":\t1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#    key：控制台配置的密钥key
//#    body：腾讯云回调返回的body体
//#    sign：腾讯云回调返回的签名值sign
//# 返回值：
//#    Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#    Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
```

```
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
```

```
# 返回值:
#    Status: OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#    Info: 成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;
```

```
    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
    }

    public function genEventSig() {
        return $this->__hmacsha256();
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
```

```
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

AI 转录/翻译 2.0 回调事件

最近更新时间：2026-04-03 09:55:13

本文针对 TRTC 后台接入 AI 转录/翻译 2.0 的回调事件进行具体说明。

配置信息

实时音视频 TRTC 控制台支持自助配置回调信息，选择 **AI 转录 2.0 回调**，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

> **注意：**
> 您需要提前准备以下信息：
> - 必要项：接收回调通知的 HTTP / HTTPS 服务器地址。
> - 可选项：计算签名的 密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。

超时重试

事件回调服务器在发送消息通知后，5秒内没有收到您的服务器的响应，即认为通知失败。首次通知失败后会立即重试，后续失败会以10秒的间隔继续重试，直到消息存续时间超过1分钟，不再重试。

事件回调消息格式

事件回调消息以 HTTP / HTTPS POST 请求发送给您的服务器，其中：

- 字符编码格式：UTF-8。
- 请求：body 格式为 JSON。
- 应答：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。
- 包体示例：下述为“启动 AI 转录与翻译 2.0”事件的包体示例。

```
{
    "EventGroupId": 14,
    "EventType": 1401,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "xxx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "RobotId": "trtc_partner_test_1",
```

```
        "Payload": {
            "Status": 0
        }
    }
}
```

参数说明

回调消息参数

- 事件回调消息的 header 中包含以下字段：

字段名	含义
Content-Type	application/json。
Sign	签名值。
SdkAppId	创建应用时控制台分配的 SdkAppId。

- 事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组 ID，转录事件组事件固定为14。
EventType	Number	回调通知的事件类型。
CallbackTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒。
EventInfo	JSON Object	事件信息 。

事件组 ID

字段名	值	含义
EVENT_GROUP_CLOUD_TRANSCRIPTION	14	转录事件组。

事件类型

字段名	值	含义
EVENT_TYPE_CLOUD_TRANSCRIPTION_START	1401	转录开始状态回调。

EVENT_TYPE_CLOUD_TRANSCRIPTION_STOP	1402	转录任务结束状态回调。
EVENT_TYPE_CLOUD_TRANSCRIPTION_ASR_MSG	1403	回调转录 ASR 的完整句子。
EVENT_TYPE_CLOUD_TRANSCRIPTION_TRANSLATE_MSG	1404	回调转录翻译的完整句子。

EVENT_TYPE_CLOUD_TRANSCRIPTION_START

字段名	类型	含义
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒。
TaskId	String	任务 ID。
RoomId	String	TRTC 的房间 ID。
RoomIdType	Integer	• 0：表示数字房间号。 • 1：表示字符串房间号。
RobotId	String	机器人 ID。
Payload.Status	Number	• 0：启动转录任务。 • 1：启动转录任务失败。

```
{
    "EventGroupId": 14,
    "EventType": 1401,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "xxx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "RobotId": "trtc_partner_test_1",
        "Payload": {
            "Status": 0
        }
    }
}
```

EVENT_TYPE_CLOUD_TRANSCRIPTION_STOP

字段名	类型	含义
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒。
TaskId	String	任务 ID。
RoomId	String	TRTC 的房间 ID。
RoomIdType	Integer	• 0：表示数字房间号。 • 1：表示字符串房间号。
RobotId	String	机器人 ID。
Payload.LeaveCode	Integer	• 0：转录模块正常调用停止转录退出。 • 1：转录机器人被客户踢出房间。 • 2：客户解散房间。 • 3：服务器将转录机器人踢出。 • 4：服务器解散房间。 • 99：代表房间内除了转录机器人没有其他用户流，超过指定时间退出。 • 101：同一用户重复进入相同房间导致机器人退出。

```
{
    "EventGroupId": 14,
    "EventType": 1402,
    "CallbackTs": 1687770730166,
    "EventInfo": {
        "EventMsTs": 1622186275757,
        "TaskId": "xxx",
        "RoomId": "1234",
        "RoomIdType": 0,
        "RobotId": "trtc_partner_test_1",
        "Payload": {
            "LeaveCode": 0
        }
    }
}
```

EVENT_TYPE_CLOUD_TRANSCRIPTION_ASR_MSG

字段名	类型	含义
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒。
TaskId	String	任务 ID。
RoomId	String	TRTC 的房间 ID。
RoomIdType	Integer	• 0：表示数字房间号。 • 1：表示字符串房间号。
RobotId	String	机器人 ID。

Payload	JSON Object	为 JSON 对象：
		{
		"UserId": "Trtc_User_0",
		"Text": "xxxx",
		"StartTimeMs": 108,
		"EndTimeMs": 10568,
		"RoundId": "40c9e724-3268-4b66-a9ff-41ed44d8edb6",
		"StartUtcMs": 1761568438912,
		"EndUtcMs": 1761568449372
		}


```
{
    "EventGroupId":  14,
    "EventType":     1403,
    "CallbackTs":    1687770730166,
    "EventInfo": {
      "EventMsTs": 1761568449890,
      "TaskId": "xxx",
      "RoomId": "1234",
      "RoomIdType": 0,
      "RobotId": "trtc_partner_test_1",
      "Payload": {
          "UserId": "Trtc_User_0",
          "Text": "Oh yeah? What's the ultimate predator? What's the ultimate predator? What's the enemy you harbor in your own heart? Who
```

```
hates you? That's the ultimate predator.",
            "StartTimeMs": 108,
            "EndTimeMs": 10568,
            "RoundId": "40c9e724-3268-4b66-a9ff-41ed44d8edb6",
            "StartUtcMs": 1761568438912,
            "EndUtcMs": 1761568449372
        }
    }
}
```

EVENT_TYPE_CLOUD_TRANSCRIPTION_TRANSLATE_MSG

字段名	类型	含义
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒。
TaskId	String	任务 ID。
RoomId	String	TRTC 的房间 ID。
RoomIdType	Integer	• 0：表示数字房间号。 • 1：表示字符串房间号。
RobotId	String	机器人 ID。
Payload	JSON Object	为 JSON 对象： `{ "UserId": "Trtc_User_0", "Text": "Presume, was exactly the same way. ", "TranslateMsg": [ { "Language": "fr", "Text": "Je suppose, c'était exactement la même chose. " }, { "Language": "zh", "Text": "大概也是一模一样的。"`

```
      }
    ],
    "StartTimeMs": 108,
    "EndTimeMs": 10568,
    "RoundId": "40c9e724-3268-
4b66-a9ff-41ed44d8edb6",
    "StartUtcMs": 1761568438912,
    "EndUtcMs": 1761568449372
}
```

```
{
    "EventGroupId":  14,
    "EventType":     1404,
    "CallbackTs":    1687770730166,
    "EventInfo": {
      "EventMsTs": 1761568449890,
      "TaskId": "xxx",
      "RoomId": "1234",
      "RoomIdType": 0,
      "RobotId": "trtc_partner_test_1",
      "Payload": {
          "UserId": "Trtc_User_0",
          "Text": "presume, was exactly the same way. ",
          "TranslateMsg": [
              {
                  "Language": "fr",
                  "Text": "Je suppose, c'était exactement la même
chose."
              },
              {
                  "Language": "zh",
                  "Text": "大概也是一模一样的。"
              }
          ],
          "StartTimeMs": 108,
          "EndTimeMs": 10568,
          "RoundId": "40c9e724-3268-4b66-a9ff-41ed44d8edb6",
```

```
            "StartUtcMs": 1761568438912,
            "EndUtcMs": 1761568449372
        }
    }
}
```

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

注意：

body 为您收到回调请求的原始包体，不要做任何转化，示例如下：

```
body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t103,\n\t\"CallbackTs
\":\t1615554923704,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t12345,\
n\t\t\"EventTs\":\t1608441737,\n\t\t\"UserId\":\t\"test\",\n\t\t\"
UniqueId\":\t1615554922656,\n\t\t\"Role\":\t20,\n\t\t\"Reason\":\t
1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#    key：控制台配置的密钥key
//#    body：腾讯云回调返回的body体
//#    sign：腾讯云回调返回的签名值sign
//# 返回值：
```

```
//#    Status: OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#    Info: 成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256
```

```
# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```php
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
    }

    public function genEventSig() {
        return $this->__hmacsha256();
    }
}

$key="789";
$data="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $data);
echo $api->genEventSig();
```

Golang

```golang
package main
import "fmt"
import (
    "crypto/hmac"
```

```
    "crypto/sha256"
    "encoding/base64"
)

func main () {
    var data = "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1
608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"Event
Ts\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(data,key))
}

func hmacsha256(data string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(data))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

签名校验示例

最近更新时间：2024-08-09 19:11:21

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。在配置回调信息前，您需提前准备计算签名的 密钥 key，由您自定义一个最大32个字符的 key，以大小写字母及数字组成。
本文档将帮助您在计算签名后，如何校验签名进行示例。

计算签名

签名由 HMAC SHA256 加密算法计算得出，您的事件回调接收服务器收到回调消息后，通过同样的方式计算出签名，相同则说明是腾讯云的实时音视频的事件回调，没有被伪造。签名的计算如下所示：

```
//签名 Sign 计算公式中 key 为计算签名 Sign 用的加密密钥。
Sign = base64(hmacsha256(key, body))
```

注意：
body 为您收到回调请求的原始包体，**不要做任何转化，需要完整保留\n\t转义字符**，示例如下：

```
body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t103,\n\t\"CallbackTs
\":\t1615554923704,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t12345,\
n\t\t\"EventTs\":\t1608441737,\n\t\t\"UserId\":\t\"test\",\n\t\t\"
UniqueId\":\t1615554922656,\n\t\t\"Role\":\t20,\n\t\t\"Reason\":\t
1\n\t}\n}"
```

签名校验示例

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
//# 功能：第三方回调sign校验
//# 参数：
//#    key：控制台配置的密钥key
//#    body：腾讯云回调返回的body体
```

```
//#   sign：腾讯云回调返回的签名值sign
//# 返回值：
//#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
//#   Info：成功/失败信息

public class checkSign {
    public static String getResultSign(String key, String body) throws
Exception {
        Mac hmacSha256 = Mac.getInstance("HmacSHA256");
        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
"HmacSHA256");
        hmacSha256.init(secret_key);
        return
Base64.getEncoder().encodeToString(hmacSha256.doFinal(body.getBytes()));
    }
    public static void main(String[] args) throws Exception {
        String key = "123654";
        // 请确保 body 为您收到回调请求的原始包体，不要做任何转化，需要完整保留\n\t
转移字符，示例如下：
        String body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}";
        String Sign = "kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA=";
        String resultSign = getResultSign(key, body);

        if (resultSign.equals(Sign)) {
            System.out.println("{'Status': 'OK', 'Info': '校验通过'}");
        } else {
            System.out.println("{'Status': 'FAIL', 'Info': '校验失败'}");
        }
    }
}
```

Python

```
# -*- coding: utf8 -*-
import hmac
import base64
from hashlib import sha256

# 功能：第三方回调sign校验
# 参数：
#   key：控制台配置的密钥key
#   body：腾讯云回调返回的body体
#   sign：腾讯云回调返回的签名值sign
# 返回值：
#   Status：OK 表示校验通过，FAIL 表示校验失败，具体原因参考Info
#   Info：成功/失败信息

def checkSign(key, body, sign):
    temp_dict = {}
    computSign = base64.b64encode(hmac.new(key.encode('utf-8'),
body.encode('utf-8'), digestmod=sha256).digest()).decode('utf-8')
    print(computSign)
    if computSign == sign:
        temp_dict['Status'] = 'OK'
        temp_dict['Info'] = '校验通过'
        return temp_dict
    else:
        temp_dict['Status'] = 'FAIL'
        temp_dict['Info'] = '校验失败'
        return temp_dict

if __name__ == '__main__':
    key = '123654'
     # 请确保 body 为您收到回调请求的原始包体，不要做任何转化，需要完整保留\n\t转移
字符，示例如下：
    body = "{\n" + "\t\"EventGroupId\":\t2,\n" +
"\t\"EventType\":\t204,\n" + "\t\"CallbackTs\":\t1664209748188,\n" +
"\t\"EventInfo\":\t{\n" + "\t\t\"RoomId\":\t8489,\n" +
"\t\t\"EventTs\":\t1664209748,\n" +
"\t\t\"EventMsTs\":\t1664209748180,\n" +
"\t\t\"UserId\":\t\"user_85034614\",\n" + "\t\t\"Reason\":\t0\n" +
"\t}\n" + "}"
```

```
    sign = 'kkoFeO3Oh2ZHnjtg8tEAQhtXK16/KI05W3BQff8IvGA='
    result = checkSign(key, body, sign)
    print(result)
```

PHP

```
<?php

class TlsEventSig {

    private $key = false;
    private $body = false;

    public function __construct( $key, $body ) {
        $this->key = $key;
        $this->body = $body;
    }

    private function __hmacsha256() {
        $hash = hash_hmac( 'sha256', $this->body, $this->key, true );
        return base64_encode( $hash);
    }

    public function genEventSig() {
        return $this->__hmacsha256();
    }
}

$key="789";
// 请确保 body 为您收到回调请求的原始包体，不要做任何转化，需要完整保留\n\t转移字符，示例如下：
$body="
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"EventTs\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}";

$api = new  TlsEventSig($key, $body);
echo $api->genEventSig();
```

Golang

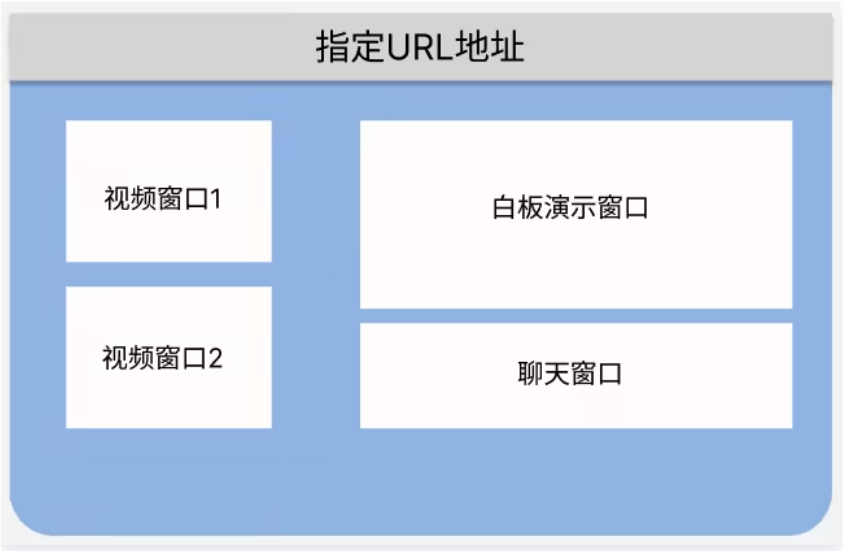
```
package main
import "fmt"
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
)

func main () {
     // 请确保 body 为您收到回调请求的原始包体，不要做任何转化，需要完整保留\n\t转移字符，示例如下：
    var body= "
{\n\t\"EventGroupId\":\t1,\n\t\"EventType\":\t101,\n\t\"CallbackTs\":\t1608086882372,\n\t\"EventInfo\":\t{\n\t\t\"RoomId\":\t20222,\n\t\t\"EventTs\":\t1608086882,\n\t\t\"UserId\":\t\"222222_phone\"\n\t}\n}"
    var key = "789"

    //JSRUN引擎2.0，支持多达30种语言在线运行，全仿真在线交互输入输出。
    fmt.Println(hmacsha256(body,key))
}

func hmacsha256(body string, key string) string {
    h := hmac.New(sha256.New, []byte(key))
    h.Write([]byte(body))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}
```

实现页面录制

最近更新时间：2025-06-05 18:23:52

应用场景

在 Web 端的远程教育、视频会议、远程定损、金融双录、在线医疗等应用场景中，考虑取证、质检、审核、存档和回放等需求，常需要将整个视频通话或互动直播过程录制和存储下来的情况。页面录制提供在云端录制任何一个浏览器页面并存储的能力，实现随时随地回看。
以在线教育，视频会议场景为例，通过页面录制可以全场景录制页面内的全部元素，包括音视频画面，白板演示以及聊天窗口等各类内容，并保证通话内容与白板时间同步，可以完整的录制这个“课堂”或“会议”的所有实时信息，达到所见即所得的目的。



费用说明：

- 能力位解锁：使用页面录制功能需为您发起调用的应用（SDKAppId）订阅 TRTC 包月套餐 部分版本才可使用，前往购买 。
- 用量费用：使用页面录制功能会产生用量费用，详细请见 页面录制费用 说明。

功能说明

通过 TRTC 的页面录制功能，您可以获取整个浏览器页面的所有原始内容，并按照需要上传到指定的对象存储平台或点播平台。录制结果文件支持 MP4 和 HLS 格式。

- 录制浏览器页面： 我们通过指定浏览器页面，可以实现录制实时浏览器页面的能力。
- 录制模式：支持 页面录制 和 页面录制与转推 两种模式。
- 录制文件格式：支持 MP4 格式和 HLS 格式。
- 输出分辨率：在不超过1920*1080的限制下，我们支持设置不同的输出分辨率。
- 文件存储：支持存储到对象存储COS 或存储到云点播VOD

- 回调通知：我们支持回调通知的能力，通过配置您的回调域名，页面录制的事件状态会通知到您的回调服务器。录制回调地址配置和事件说明请见 云端录制和页面录制回调 。

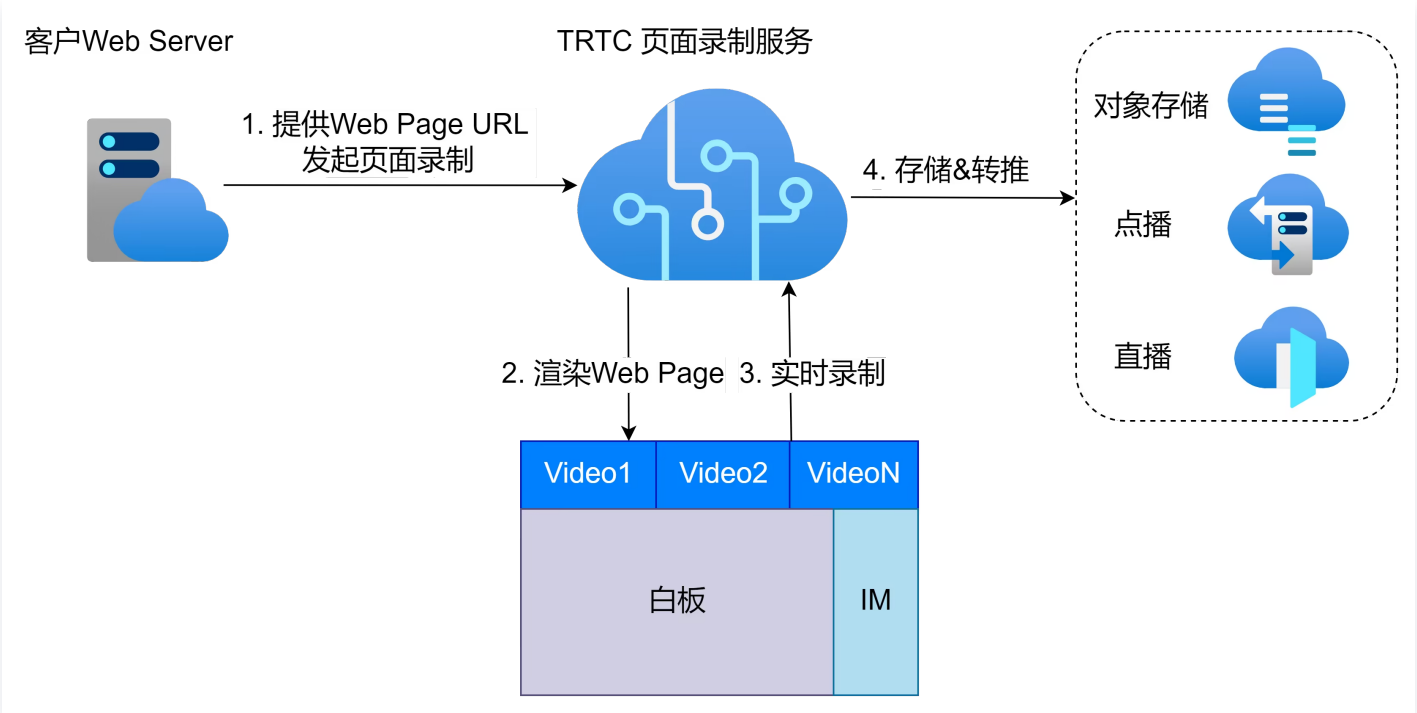
> ⚠ **注意：**
> 待录制页面需要 URL 加载完即播放，且不需要任何形式的交互操作，其他更多注意事项请参见 接入注意事项 。

录制模式说明

页面录制

如下图所示为使用页面录制的经典场景：

1. 客户可以通过云 API 发起页面录制请求，在该请求中指定待录制的 Web Page 的 URL，以及录制存储参数。
2. TRTC 页面录制服务会在云端访问指定的 URL 并实时渲染，保留 Web Page 所呈现的所有原始内容。
3. TRTC 页面录制服务对渲染出的 Web Page 进行实时录制，使录制结果能够还原 Web Page 的全景真实效果。
4. 在录制任务结束后，会根据录制参数生成 HLS 或者 MP4 文件并上传到指定云存储平台（目前支持腾讯云COS和腾讯云VOD）。



页面录制并转推

TRTC 对于具有同时使用页面录制并转推场景的客户，提供了可一次调用发起可同时实现转推并录制的使用方式（在页面录制请求中设置转推参数），在进行录制的同时，将音视频流实时转推到 CDN 平台，实现转推观看和录制内容完全一致。

> **注意：**
> 使用页面录制并转推模式时，转推相关操作请关注 旁路转推回调，若同时发起转推功能会产生 旁路转推费用。

文件切分说明

录制 MP4 文件切分的条件：

- 录制切分时长可设置范围1-1440分钟，默认120分钟。
- 单个 MP4 文件大小达到 2GB。

录制上传云存储说明

录制后台会在录制结束后将录制的文件通过您指定的方式上传到云存储平台（云点播 VOD 或对象存储 COS），并通过回调的形式把录制结果（播放地址或录制文件名）发送给您。

- 上传对象存储COS时，为确保您能够获取到媒体文件存储地址，请您关注310回调。310回调把录制任务ID(TaskId) 以及录制文件名(FileList)发送给您。您需要根据页面录制请求中的第三方存储桶信息（StorageParams.CloudStorage.Bucket）、第三方云存储的地域信息(StorageParams.CloudStorage.Region)、文件位置信息(StorageParams.CloudStorage.FileNamePrefix)以及录制任务ID(TaskId)和录制文件名(FileList)自行拼接出媒体文件播放地址。
- 上传云点播VOD时，为确保您能够获取到媒体文件播放地址，请您关注311回调。311回调会把录制文件在点播平台的播放地址发送给您。

> **注意：**
> - 文件录制后会上传至您指定云存储平台（云点播 VOD 或对象存储 COS），为确保录制文件成功，请确保您指定的云点播 VOD 或对象存储 COS 服务可用。
> - 页面录制仅支持上传单一云存储平台（云点播 VOD 或对象存储 COS），不支持同时设置对应存储参数。
> - 上传对象存储 COS 时，310回调返回的录制任务 ID(TaskId)会额外携带类似“_StartTimeMs_1717156238963”这样的后缀，后缀中的时间戳长度固定为13位，您可以按实际业务需要对返回的录制任务 ID 进行截取。

API 使用接口说明

API 接口和录制并发限制

- 录制接口的调用频率限制为20qps。
- 单个接口超时时间为5秒。
- 单个应用下默认并发录制支持200路，超过并发限制的任务会失败。

发起录制

通过调用 API（ StartWebRecord ）来启动页面录制，需要重点关注响应结果中的参数——任务 ID（TaskId）；这个参数是本次录制任务的唯一标识，您需要保存下这个任务 ID 作为后续针对这个录制任务接口操作的输入参数。

> **说明：**
> 您可以前往控制台配置回调地址，以接收录制回调事件，请见 云端录制和页面录制回调说明 。

查询录制状态

通过调用 API（ DescribeWebRecord ）来查询录制任务状态，输入参数为录制时响应结果中的任务 ID（TaskId，这个参数是本次录制任务的唯一标识），或在输入参数中使用发起页面录制时输入的SdkAppId和RecordId，通过上述参数可以查询到对应录制任务的录制状态。

- 录制任务进行中：调用 API 返回的响应中的 Status 为1时，代表录制任务正在进行中。
- 录制任务已结束：调用 API 返回的响应中的 Message 为“**task not exist**”时，代表录制任务已结束或尚未启动。

停止录制

通过调用 API（ StopWebRecord ）来停止录制任务，需要使用发起录制时响应结果中的参数——任务 ID（TaskId）；这个参数是本次录制任务的唯一标识，通过这个参数就可以停止对应录制任务。

> **说明：**
> 您可以通过启动录制中的 MaxDurationLimit 参数来指定录制任务的持续时间，录制任务持续时长达到指定的MaxDurationLimit值时，会自动停止录制，从而省去调用停止录制 API 的操作。默认录制任务最大录制时长为10小时。

录制回调事件

页面录制功能提供了多种的回调事件，帮助您及时了解录制任务的处理进度和完成情况，录制回调地址配置和事件说明请见 云端录制和页面录制回调 。

录制文件管理

查找录制文件

录制任务结束后，TRTC 页面录制中录制下来的文件上传至您指定的云存储平台（云点播VOD或对象存储COS）。您可以直接前往 云点播控制台 或 对象存储COS控制台 查找。

> **说明：**

> 对于对象存储 COS，如果您在启动录制参数中设置了 FileNamePrefix 参数，录制文件将保存在您指定的存储桶 `Bucket/${FileNamePrefix}/${TaskId}` 下；否则，录制文件将直接保存在存储桶 `Bucket/${TaskId}` 下。

接收录制文件

录制文件上传云点播 VOD 时，除了手动查找录制文件，您还可以通过在控制台 配置回调地址，让腾讯云主动把新录制文件的消息推送给您的服务器。
房间里的最后一路音视频流退出后，该过程大约默认需要30秒至数分钟（具体时间根据您所录制的文件大小而定）。转存完成后，腾讯云会通过您在 设置录制回调 中设置的回调地址（HTTP/HTTPS）向您的服务器发送通知。
腾讯云会将录制和录制相关的事件都通过您设置的回调地址推送给您的服务器，您可以通过接收事件类型为311的上传成功回调来获取录制文件的播放地址 VideoUrl，具体回调信息见下方：

```
{
  "EventGroupId": 3,
  "EventType": 311,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "UserId": "xx",
    "TaskId": "xx",
    "Payload": {
      "Status": 0,
      "TencentVod": {
        "UserId": "xx",
        "TrackType": "audio_video",
        "MediaId": "main",
        "FileId": "xxxx",
        "VideoUrl": "http://xxxx",
        "CacheFile": "xxxx.mp4",
        "StartTimeStamp": "xxxx",
        "EndTimeStamp": "xxxx"
      }
    }
  }
}
```

接入注意事项

针对 Web 应用的限制

- 由页面录制生成的视频分辨率上限为 1920 × 1080。
- 待录制的网页中任何视频源的分辨率不应超过 1920 × 1080。
- 待录制页面的下行带宽不得超过 5 Mbps，上行带宽不得超过 5 Mbps。
- 待录制页面不应使用 WebGL 功能。
- 请确保您的Web应用不会过度占用CPU、内存和带宽，并且该Web应用的使用应符合法律法规。
- 页面录制支持 在无用户交互的情况下自动播放已启用autoplay属性的video元素。然而，如果待录制的网页中的video元素未启用autoplay属性，其内容将不会自动播放，这可能导致页面录制无法录制该网页。
- 待录制页面不应跳转至不同域名的 URL，并尽可能避免其他形式的跳转。如果待录制的页面需要登录操作，请先处理登录操作，然后进行录制。否则，录制结果可能会一直是待登录界面。

云 API 请求

- 从请求发起到开始页面录制，可能会有约5秒的延迟。建议提前发起录制请求，以确保录制内容的完整性。
- 页面录制不支持更改布局。
- 如果您在StartWebRecord方法中填入的RecordUrl无法正常打开，录制服务将在StartWebRecord成功后自动退出。您可以参考云端录制集成最佳实践，使用退避策略多次调用DescribeWebRecord，以确认录制服务已正常启动。

检测页面加载超时

页面录制场景下，网络异常等偶然因素可能会造成以下问题：

- 无法正常访问待录制页面，如页面加载失败或时间过长，无法获知真正开始有效录制的时间点，可能会丢录重要内容。
- 可以正常访问待录制页面，但未能正确加载页面中的 HTML 元素。
- 录制过程中未能正常加载页面中发生变化的 HTML 元素，从而导致录制内容与预期不一致。
- 未能正常播放待录制页面中的音视频。

为了确保页面录制的内容与预期一致，建议您按照以下方案来提高页面录制的可靠性。

1. **设置页面加载超时时间**

   调用 StartWebRecord 方法时通过 `ReadyTimeout` 字段设置页面加载超时的时间限制。

   `ReadyTimeout`：Number 类型，单位为秒，取值范围 [0,60]：

   - `0` 或不设置，表示不检测页面加载状态。
   - `≥ 1`，表示页面加载超时时间。
   - `< 0` 或非整数，表示设置错误，云API会返回错误信息。

   > ⚠ **注意：**

> 当您设置了 `ReadyTimeout` 时，请务必确保待录制页面有判断页面是否加载完成，以避免因未检测到页面加载就绪从而导致录制任务启动失败。

2. 判断加载是否完成并通知浏览器

> ⚠ **注意：**
> 您需要自行判断页面是否加载完成，然后实现后续逻辑。

页面加载完成

如果页面加载完成，则在设定的 `ReadyTimeout` 时间内调用 `notifyReady` 方法通知浏览器页面加载成功。
`notifyReady` 的调用示例如下：

```
// notifyReady调用示例
<script>
function notifyReady() {
    if (window.notifyReady) {
        window.notifyReady();
    }
}
</script>
```

页面加载超时

如果页面加载超时，即在设定的 `ReadyTimeout` 时间内未调用 `notifyReady` 方法通知浏览器，则浏览器自动重新加载页面。您会收到 803 事件回调（EVENT_TYPE_WEB_RECORDER_STATUS_UPDATE），其中 `Status` 字段为 `1`，`EventMessage` 字段为 `Page load timeout`。

- 如果重新加载成功，则参考页面加载完成的逻辑，调用 `notifyReady` 方法通知浏览器。
- 如果页面加载再次超时，则表示页面重新加载失败，录制服务停止。您会收到 801 事件回调（EVENT_TYPE_WEB_RECORDER_START）通知您录制任务启动失败，其中 `Status` 字段为 `2`，`EventMessage` 字段为 `Page load timeout`。在收到回调通知后，您可根据实际业务需要，决定是否重新发起录制任务。

其他说明

- 在录制过程中，如果当前MP4文件的时长超过maxVideoDuration的值或大小超过2GB时，录制服务将创建一个新的MP4文件。
- 如果您在 start 方法中填入待录制页面的 URL 会自动触发 Web 客户端发布音视频流，录制服务也会成为一个发流端，因此，您的应用中可能会出现一个绿色背景色的用户画面。 为规避该问题，您可以在待录制页面的 URL 中增加查询字段 is_recorder=1，例如："https://url?is_recorder=1"，并在该页面内添加以下逻辑：
  - 如果 is_recorder 为 1，则 Web 客户端不发布音视频流。
  - 如果 is_recorder 不为 1，则 Web 客户端发布音视频流。
- 进行页面录制时，录制服务相当于一个使用 Web 应用的客户端，因此，如果您的Web应用包含用户列表，建议您在用户列表中隐藏该用户。

拓展场景

TRTC 页面录制解决方案不仅可以录制 TRTC 的 RTC 会话，对于传入的任意可访问的页面，该方案均可以录制。因此对于开发者来说，借助于 TRTC 页面录制的能力，可以衍生出更多的创意玩法，例如：

1. 通过技术手段，将多人的本地页面及云端渲染页面的操作同步，通过页面录制，将多人协作的过程录制下来，作为后续的教程资料。
2. 多人可以使用页面录制方案录制一个视频源，并转推到直播 CDN，后续多人可以通过直播流一起观看。

后续 TRTC 页面录制方案，会在录制及转推基础能力上，结合AI探索更多的音视频处理增值服务，助力客户进一步降本增效，扩展业务边界。

语音转录与翻译

最近更新时间：2025-09-29 20:27:42

应用场景

TRTC 支持**语音转文字**和**翻译**功能，将房间内指定用户或所有用户的音频流识别成对应的文字以及通过 AI 翻译成其他语言，实现实时字幕和实时翻译等效果。

前提条件

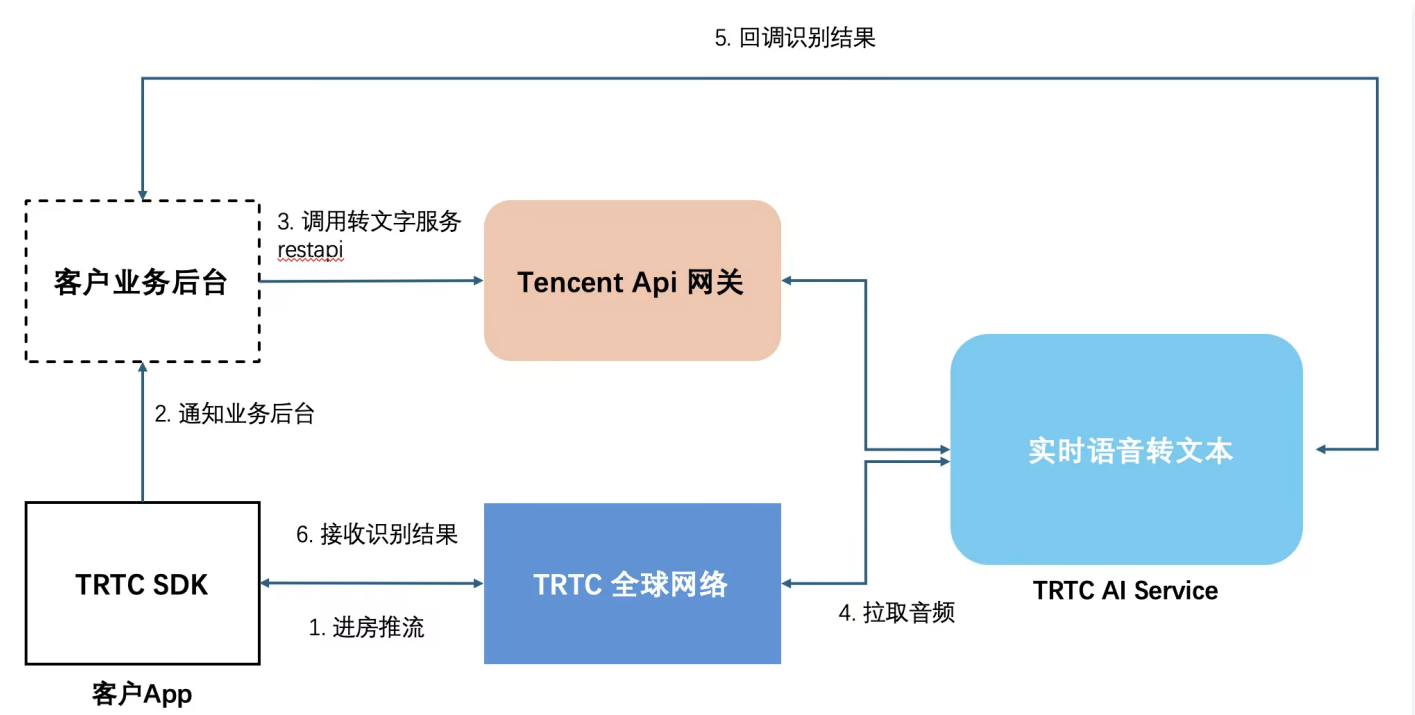
- 登录 TRTC 控制台，开通 TRTC 服务并 创建应用 。
- 前往 控制台 > 功能配置 > 增值功能 开启 AI 智能识别的语音转文字、实时翻译功能。

说明：

- 开启前需领取**包月套餐体验版**、购买 AI 智能识别套餐包 、或订阅 TRTC 包月套餐 部分版本才可使用，前往购买 。
- 语音转文字和实时翻译功能会根据调用量产生费用，详情请参见 费用详情 。

功能说明

任务发起后，TRTC AI Service 通过识别机器人进入 TRTC 房间拉流指定用户或所有用户的流，进行语音转文字识别，将识别结果实时回调给客户端和服务端。



接入说明

第一步：接收语音识别结果

方式一：通过客户端 SDK 接收文字消息

通过 TRTC SDK 接收自定义消息功能，在客户端上监听回调来接收实时的语音转文字、实时翻译的结果数据。客户端回调消息格式如下，以 Web 端为例：

```
trtc.on(TRTC.EVENT.CUSTOM_MESSAGE, event => { // receive custom message
    // event.userId: 语音识别机器人的userId
    // event.cmdId: 消息Id, 转录和字幕固定为1
    // event.seq: 消息的序号
    // event.data: ArrayBuffer 类型， 转录或字幕的内容，见下方data字段说明
    const data = new TextDecoder().decode(event.data)
    // data 字段说明如下
    console.log(`received custom msg from ${event.userId}, message: ${
data }`)
})
```

data 字段说明（实时的字幕消息）

字段名	类型	含义

type	Integer	10000: 实时字幕与完整的一句话下发消息类型。
sender	String	说话人的 userid。
receiver	Array	接收者 userid 列表，该消息实际是在房间内广播。
payload.text	String	识别出的文本。 Unicode 编码。
payload.start_time	String	消息产生的时间 任务启动后的绝对时间。
payload.end_time	String	消息结束的时间 任务启动后的绝对时间 。
payload.end	Boolean	如果为 true，代表这是一句完整的话。

```
{
  "type": 10000,
  "sender": "user_a",
  "payload": {
     "text":"",
     "start_time":"00:00:02",
     "end_time":"00:00:05",
     "end": true
  }
}
```

实时的翻译消息

```
{
  "type": 10000,
  "sender": "ai_951073",
  "payload": {
    "start_time_ms": 1760,
    "end_time_ms": 5530,
    "end": false,
    "roundid": "e6330a3c-eed7-40bb-8229-9bbe733a313f", // 一轮对话的 ID
    "translation_text": "simultaneous interpretation of the meeting", //
翻译文本
```

```
    "translation_language": "en",    // 翻译的语言代码
    "taskid": "x-dPLCz" // 转录任务唯一标识
  }
}
```

说明：
回调示例说明：
转录：会将完整的一句话转录并推送
“今天天气怎么样？”
字幕 & 翻译：将字幕句子或者翻译句子分段推送，后一段会包含前一段，确保实时性。
“今天”
“今天天气”
“今天天气怎么样”
顺序说明：字幕消息 > 字幕消息 > .... > 字幕消息(end = true)

方式二：通过服务端回调接收

语音转文字服务同时提供了服务端事件回调，便于您的服务接收实时对话的消息，查看 详细回调事件 。

第二步：发起语音转文字、实时翻译任务

TRTC 提供以下云 API 用于发起和管理语音转文字和实时翻译，具体如下：

- 开始语音转文字任务：StartAITranscription
- 查询语音转文字任务：DescribeAITranscription
- 停止语音转文字任务：StopAITranscription
- 开始实时翻译任务：在 StartAITranscription 的输入参数中配置 TranslationConfig 参数，目前支持配置以下语种翻译：

语言代码	对应语种
"zh"	中文
"en"	英语
"es"	西班牙语
"pt"	葡萄牙语
"fr"	法语
"de"	德语

"ru"	俄语
"ar"	阿拉伯语
"ja"	日语
"ko"	韩语
"vi"	越南语
"ms"	马来语
"id"	印度尼西亚语
"it"	意大利语
"th"	泰语

注意：

- 语音转文字和实时翻译功能单个 SDKAppId 任务并发数限制**100路**，如需提升提交工单处理。
- 实时翻译功能目前支持15种语言：中文、英语、西班牙语、葡萄牙语、法语、德语、俄语、阿拉伯语、日语、韩语、越南语、马来语、印度尼西亚语、意大利语、泰语，如需其他语种支持可以 联系我们 。
- 由于受不同语境或语种差异的影响，AI 翻译的译文内容适用于辅助参考，不应作为唯一专业意见或结论。

输入媒体流进房

最近更新时间：2026-03-27 12:58:31

概述

一起看、一起听、一起玩、一起学……原来需要线下面对面才能实现的各种体验正被不断搬到线上。相隔千里还能和好友们一起看电影、一起听音乐，然后一起交流吐槽，这样神奇的实时互动体验正受到当下年轻人的喜爱，并成为如今音视频产品的重点玩法和主流方向。

TRTC 提供 输入在线媒体流 和 RTMP 推流进房 两种推流进房方案，有各自对应的适用场景，具体如下：

- 输入在线媒体流 用于**拉取云端在线媒体流（在线流或云端点播文件）**推流至 TRTC 房间内进行观看。
- RTMP 推流进房 **用于将本地媒体文件、音视频设备采集音视频**通过 RTMP 标准协议推流到 TRTC 房间内。

> **说明：**
> 相关费用如下：
> - 功能位解锁：**输入在线媒体流**和 **RTMP 推流进房**功能需订阅 TRTC包月套餐 **尊享版**或**旗舰版**解锁。
> - 用量费用：
> - 使用推流功能会进行转码操作，产生转码费用，详情参见 云端混流转码计费说明 。
> - 收取推流机器人在房产生的音频时长费用（注：输入在线媒体流功能产生的机器人在房费用将限免于2024年8月15日，从2024年8月16日起开始收取）。
> - 房间内观众订阅推流进房的音视频内容会正常产生音视频通话费用，详情参见 音视频时长计费说明 。

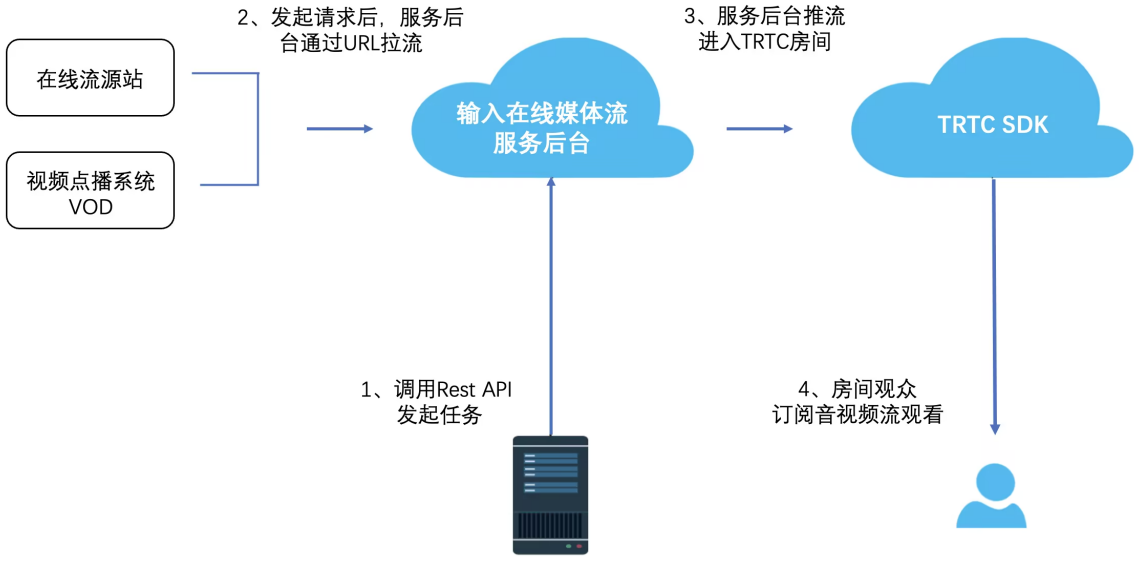
输入在线媒体流

应用场景

场景类型	说明
AI 互动课堂	依托 TRTC 输入在线媒体流能力，平台可通过录播真人教学视频结合 AI 技术进行线上直播互动教学，在保证教学效果的同时大幅降低运营成本。上课前，平台根据教师的课程设置，将知识点讲解、互动提问、问题反馈和解答等信息录制成视频片段，上传到视频库。课堂中，通过 TRTC 输入在线媒体流能力将对应视频推送到 TRTC 房间进行直播。学生通过语音、触屏实现互动式学习。服务端通过 AI 技术，智能识别学生的实时语音和作答，并根据学生的表现，无缝切换教学片段，实时给予不同的反馈，从而提供个性化的教学体验。
“一起看”房间服务	游戏直播、秀场、体育赛事等直播类内容，可以通过 TRTC 输入在线媒体流能力将直播流推送到 TRTC 房间，实现房间内超低延时同步观看，配合 TRTC 的实时互动能力，观众可实时交流，一起加油喝彩，沉浸式观赛。电影、音乐等点播类节目，同样可以通过该能力输入至 TRTC 房间，帮助用户实时共享，与好友边看边聊。

功能架构

1. 用户使用 REST API 创建输入在线媒体流任务，输入在线媒体流任务由中转服务（relay server）执行。
2. 中转服务拉取在线流或者点播文件。
3. 中转服务将拉取到的音视频推至 TRTC 房间，中转服务中会自动生成一个虚拟主播用户，该用户的用户名和要进入的房间号在创建任务时指定。
4. TRTC 其他端可观看这路流，也可以复用 TRTC 录制、转推等能力。



功能描述

输入在线媒体流功能说明如下：

类型	描述
发起任务方式	用户可以通过 REST API 发起输入在线媒体流任务，观众可观看这路流，支持录制、转推等功能。
多种源流协议和格式	协议：HTTP、HTTPS、RTMP、HLS 格式：FLV、MP3、MP4、MPEG-TS、MOV、MKV、M4A 视频编码：H.264、VP8 音频编码：AAC、OPUS
服务端回调	输入在线媒体流任务创建和结束时可回调给业务侧服务器，用于业务侧做逻辑，详细输入在线媒体流事件，

	前往查看 。

相关 Rest API

- 开启输入在线媒体流：StartStreamIngest
- 停止输入在线媒体流：StopStreamIngest
- 查询输入在线媒体流：DescribeStreamIngest

RTMP 推流进房

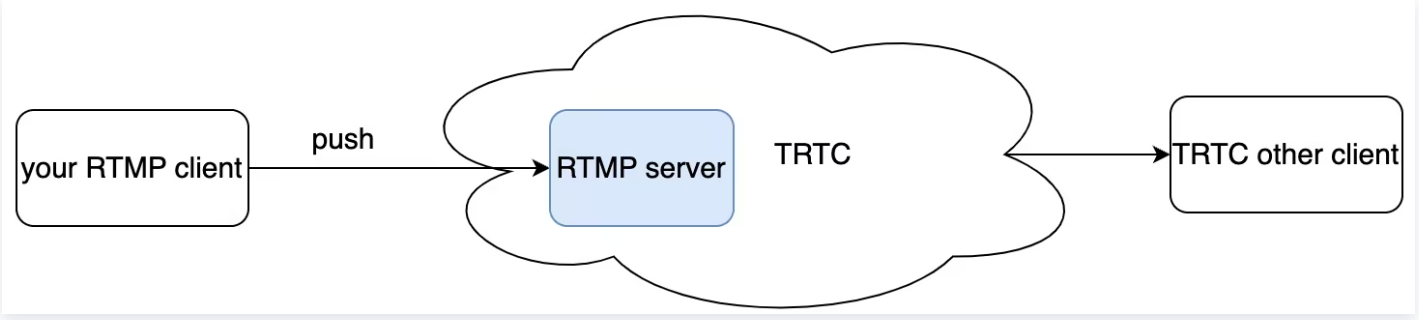
TRTC 支持将**本地媒体文件、音视频设备采集音视频**通过 RTMP 标准协议推流到 TRTC 房间内。为降低客户接入门槛，您可根据实际情况选择安装 OBS 、FFmpeg 或其他 RTMP 库进行推流。OBS 是一款好用的第三方开源程序直播流媒体内容制作软件，为用户提供免费使用，它可支持 OS X、Windows、Linux 操作系统，适用多种直播场景，满足大部分直播行为的操作需求，您可以到 OBS 官网 下载最新版本软件，使用 OBS 推流时无需安装插件。

应用场景

场景类型	说明
在线教育场景	老师展示视频课件教学视频时，可以通过 PC 端 OBS 或者 FFmpeg 把绝大多数媒体格式以 RTMP 推流至 TRTC 房间，房间内的学生通过 TRTC SDK 拉流，可以保证观看到相同进度的教学视频，课件播放跳转进度、调整速度、切换下一章等全部可由老师控制，各学生端观看对齐课堂秩序好，教学质量更稳定。
一起看球赛场景	比赛流媒体是赛事供应方固定以 RTMP 格式流的方式提供赛事画面，通过 RTMP 协议推流至 TRTC 房间，实现 TRTC 房间内同步观看超低延时的比赛直播，配合 TRTC 的实时互动能力，与好友语音/视频讨论，一起喝彩加油，不会错过每一个精彩瞬间的共享体验。
更多场景	任何基于媒体流的实时互动体验玩法，均可通过 RTMP 协议推流帮您实现，等多玩法等待您的探索。

网络架构

RTMP 属于 TRTC 的一个子模块，能与 TRTC 其他端互通，互通延迟在正常情况下小于600ms，也可使用 TRTC 录制、转推等已有能力。网络架构如下图所示。**不支持使用 RTMP 从 TRTC 拉流，只支持 RTMP 推流。**



流地址生成

推流地址

```
rtmp://rtmp.rtc.qq.com/push/房间号?sdkappid=应用&userid=用户名&usersig=签名
```

- 主域名是 rtmp.rtc.qq.com，备域名 rtmp.cloud-rtc.com，如果主域名解析有问题，可使用备域名。
- RTMP appName 是 push。
- 地址中的房间号、应用、用户名、签名需要换成业务的。
- 为简化参数，只支持字符串房间号，不超过64个字符，字符只能是数字、字母、下划线。

> **警告：**
> 1. TRTC 其他端如果要观看 RTMP 流，请使用**字符串房间号进房**。
> 2. 以 小程序端 为例填写 enterRoom 接口 strRoomID 字段，其他端参考相应的 API 文档。

- usersig 的生成规则，请参见 UserSig 相关 （**请注意签名要在有效期内**）。

示例：

```
rtmp://rtmp.rtc.qq.com/push/hello-string-room?
sdkappid=140*****66&userid=******rtmp2&usersig=eJw1jdE********RBZ8qKGRj8
Yp-wVbv*mGMVZqS7w-mMDQL
```

使用示例

您可以使用支持 RTMP 协议的软件或者代码库推流，下面列举几种。

OBS 推流

准备工作

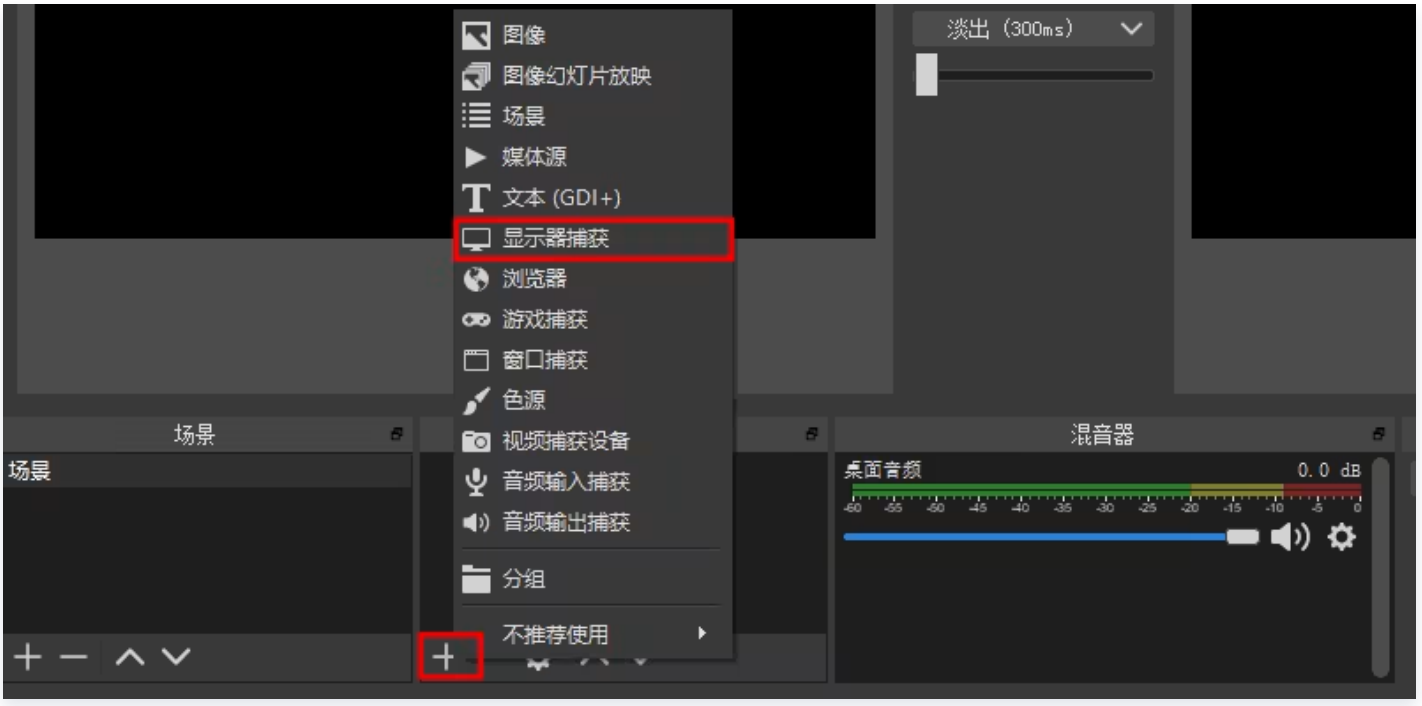
安装并打开 OBS 工具进行下述操作。

步骤1：选择输入源

查看底部工具栏的**来源**标签，单击+ ，根据您的业务需要选择输入源。常用来源输入有：

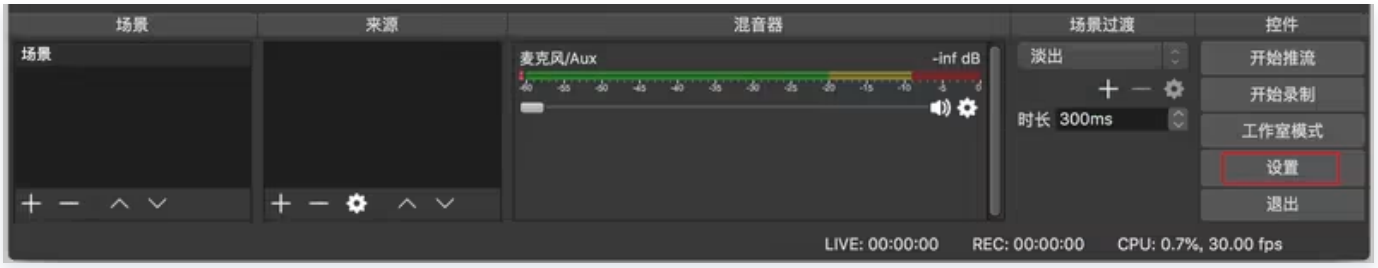
输入源	说明
图像	适用于单张图像直播
图像幻灯片放映	可循环或者顺序多张播放图片

场景	实现各种强大的直播效果。此时，另一个场景是作为来源被添加进当前场景的，可以实现整个场景的插入
媒体源	可上传本地视频，并本地点播视频文件进行直播化处理
文本	实时添加文字在直播窗口中
窗口捕获	可根据您选择的窗口进行实时捕获，直播仅显示您当前窗口内容，其他窗口不会进行直播捕获
视频捕获设备	实时动态捕捉摄像设备，可将摄像后的画面进行直播
音频输入捕获	用于音频直播活动（音频输入设备）
音频输出捕获	用于音频直播活动（音频输出设备）



步骤2：设置推流参数

1. 通过底部工具栏的**控件 > 设置**按钮进入设置界面。

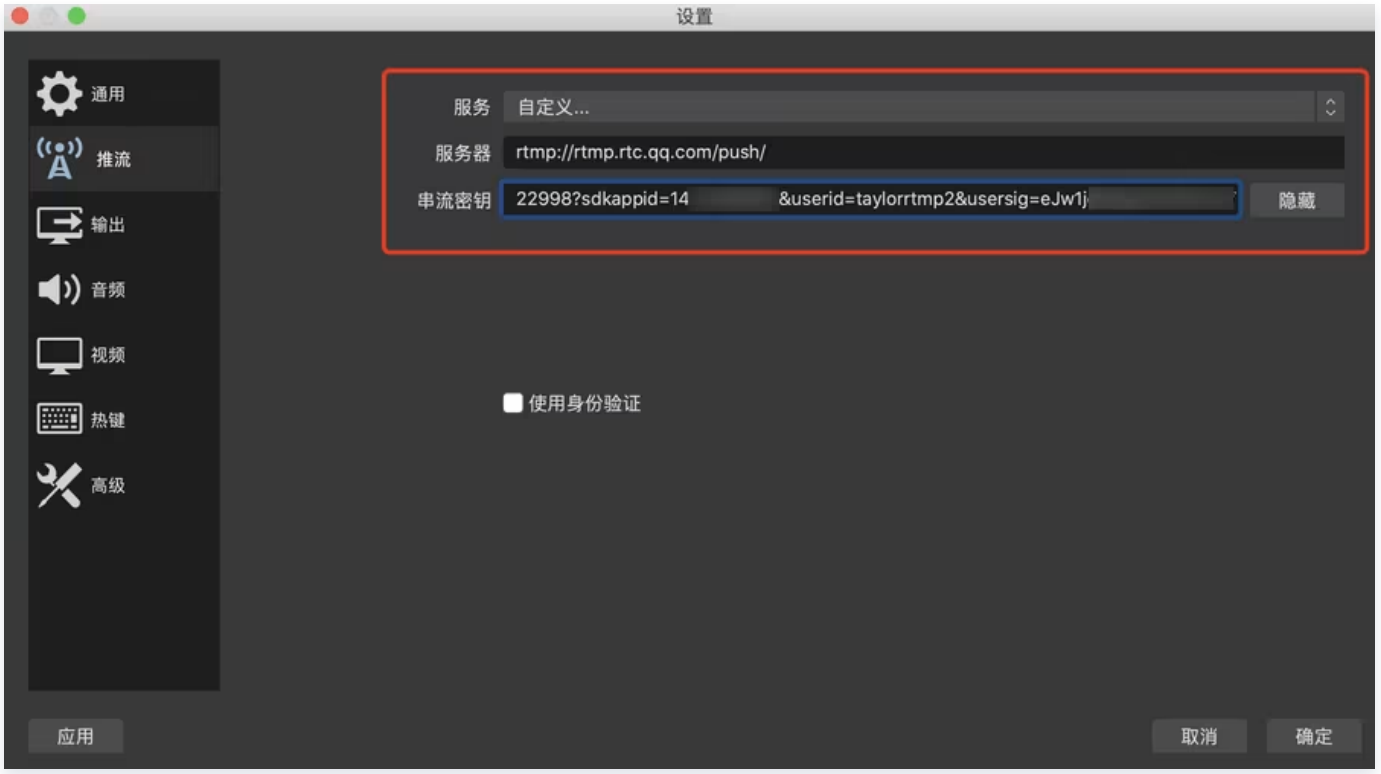


2. 单击**推流**进入推流设置页签，选择服务类型为**自定义**。
3. 服务器填写：`rtmp://rtmp.rtc.qq.com/push/` 。
4. 填写串流密钥格式如下：

```
房间号?sdkappid=应用&userid=用户名&usersig=签名
```

其中房间号、应用、用户名、签名需要换成业务的，参考流地址生成章节。例如：

```
hello-string-room?
sdkappid=140*****66&userid=******rtmp2&usersig=eJw1jdE***************Z
Lgi5UAgOzoMhrayt*cjbmiCJ699T09juc833IMT94Ld7I0iHZqVDzvVAqkZsG-
IKlzLiXOnEhswHu1iUyTc9pv*****D8MQwoA496Ke6U1ip4EAH4UMc5H9pSmv6MeTBWLam
hwFnWRBZ8qKGRj8Yp-wVbv*mGMVZqS7w-mMDQL
```

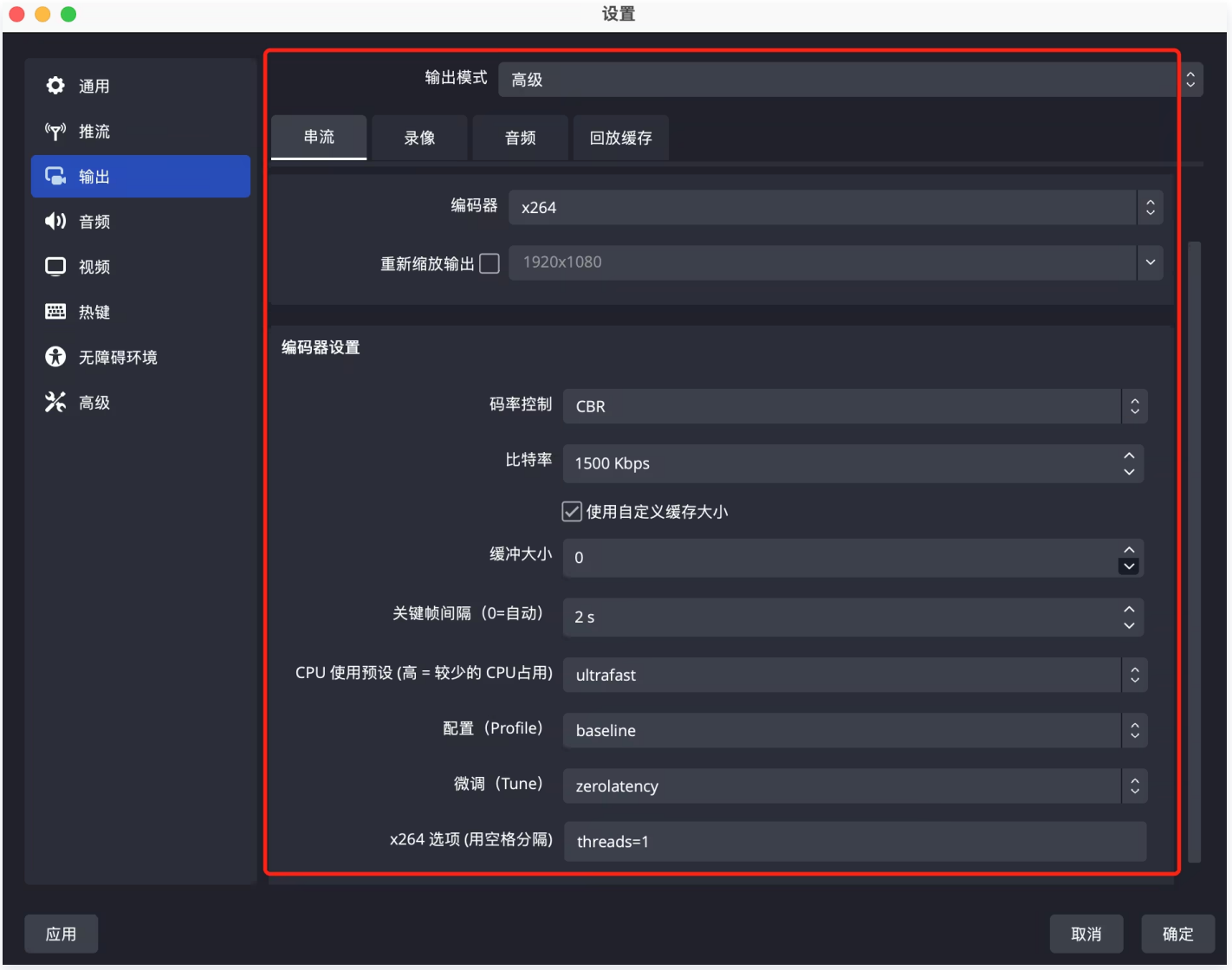


步骤3：设置输出

RTMP 后台不支持传输 B 帧，用户可以通过如下设置调整推流端软件的视频编码参数来去除 B 帧。

1. 在**设置**中单击**输出**页签进行配置。
2. 在**输出模式**中选择**高级**，**关键帧间隔建议填写1或2**，CPU 使用预设为 ultrafast，配置选择 baseline，微调选择 zerolatency，x264 选项填写 threads=1，单击**确定**保存设置。

> **警告：**
> 推流需要去除 B 帧，否则推流后连接会被断开，下面的配置选择 baseline 可去除 B 帧。



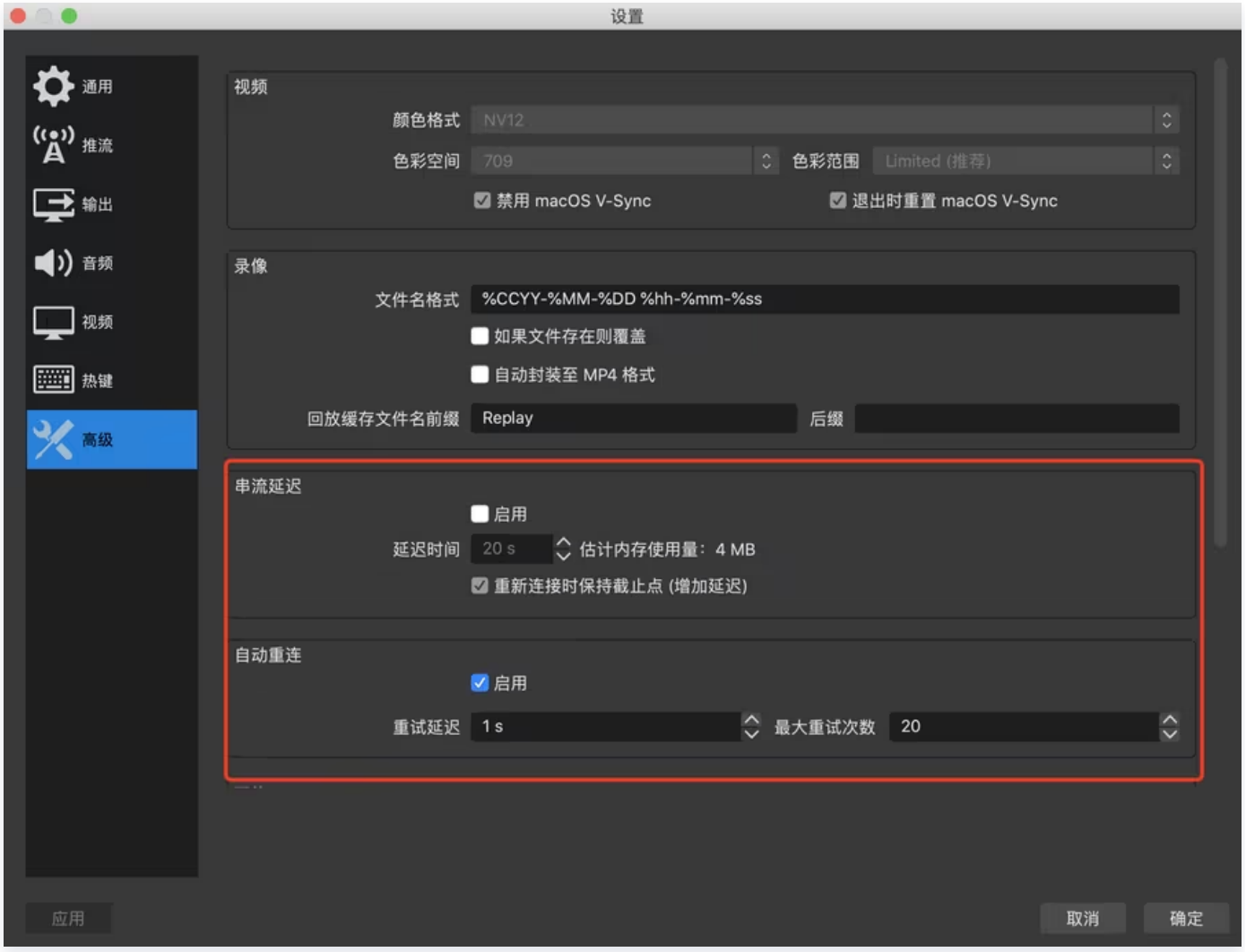
步骤4：设置视频选项

在**设置**中单击**视频**页签，设置分辨率和帧率。分辨率决定了观众看到的画面清晰程度，分辨率越高画面越清晰。FPS 是视频帧率，它控制观看视频的流畅，普通视频帧率有24帧 – 30帧，低于16帧画面看起来有卡顿感，而游戏

对帧率要求比较高，一般小于30帧游戏会显得不连贯。

步骤5：设置高级选项

- 建议不启用**串流延迟**以减少端到端延迟。
- 启动**自动重连**，建议设置**重试延迟**时长尽量短，网络抖动时如果连接断开可尽快重连上。



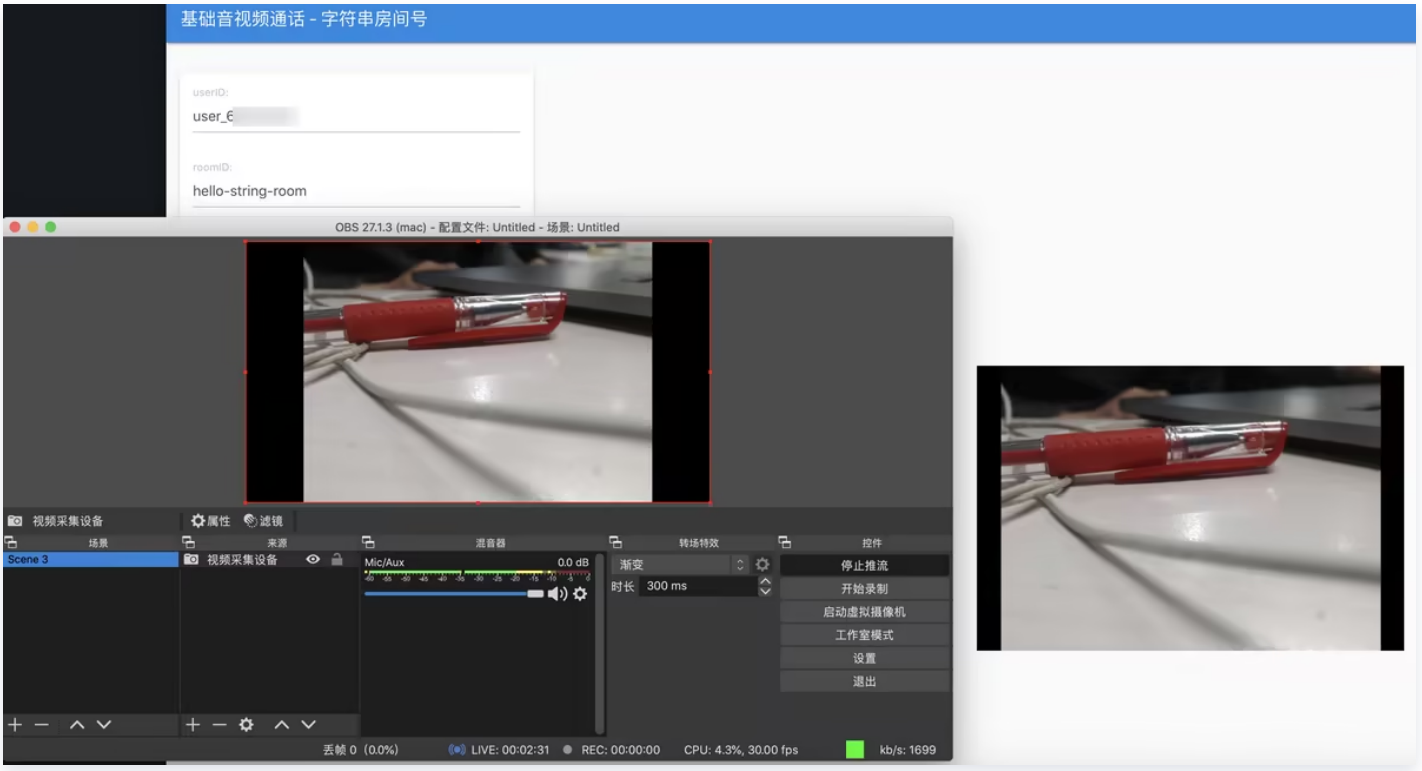
步骤6：单击推流

1. 查看 OBS 底部工具栏的**控件**，单击**开始推流**。

2. 推流成功后，正常情况在界面底部会展示推流状态，TRTC 控制台仪表盘 上有该用户进房记录。

步骤7：其他端观看

如前面 设置推流参数 所说，TRTC 其他端进房需要使用字符串房间号，Web 端 观看 RTMP 流的效果如下所示，您也可以选择使用其他端观看。



FFmpeg 推流

如果需要用命令行或其他 RTMP 库推流，使用完整的流地址供 FFmpeg 或其他 RTMP 库推流，视频编码使用 H.264，音频编码使用 AAC，容器格式使用 FLV，建议 GOP 设置为2s或1s。
FFmpeg 不同场景下指令配置参数不同，因此需要您具有一定的 FFmpeg 使用经验，以下列出 FFmpeg 常用命令行选项，更多 FFmpeg 选项请参见 FFmpeg 官网 。

FFmpeg 命令行

```
ffmpeg [global_options] {[input_file_options] -i input_url} ...
{[output_file_options] output_url}
```

常见的 FFmpeg 选项

选项	说明
-re	以 native 帧率读取输入，通常只用于读取本地文件

其中 output_file_options 可配置选项包括：

选项	说明

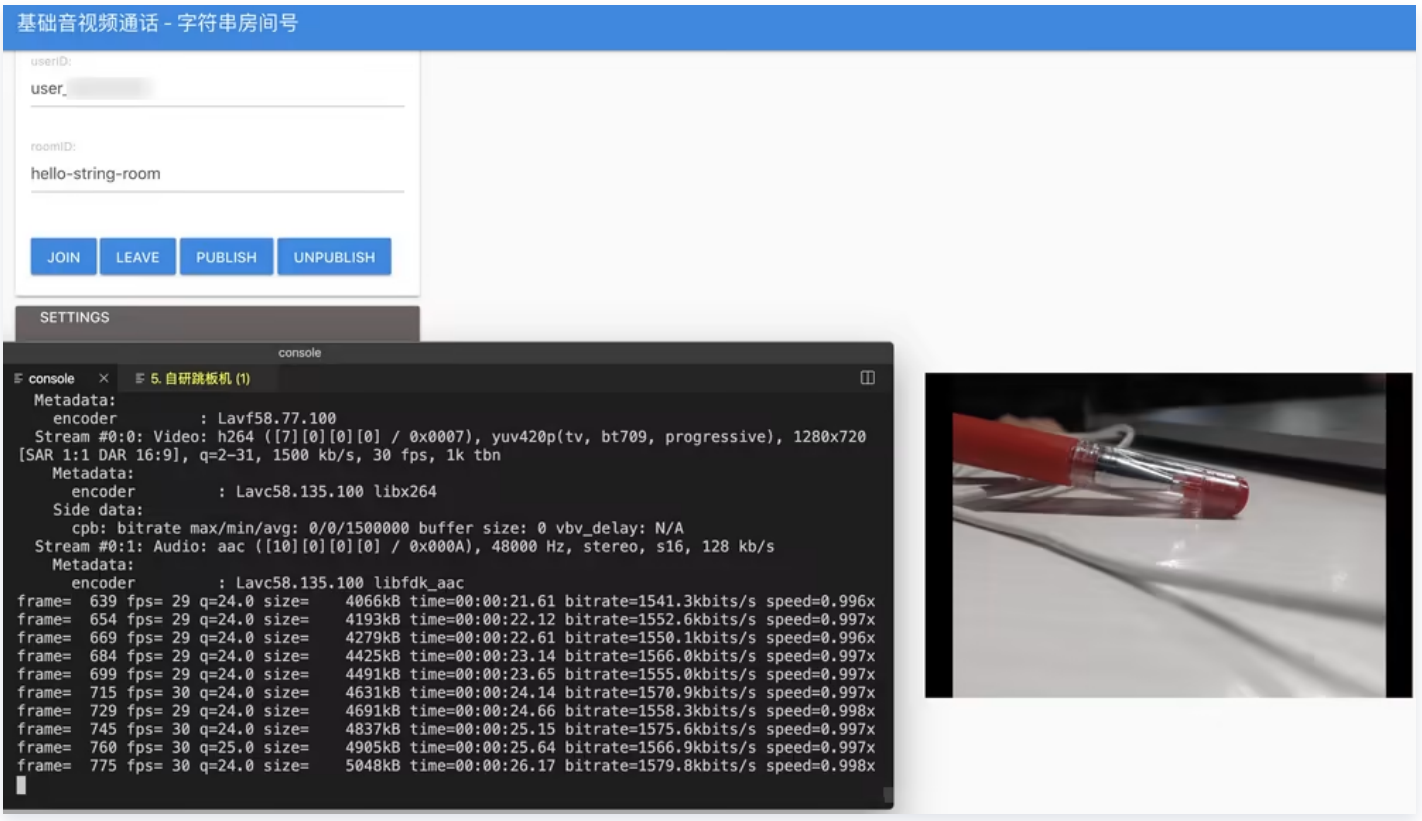
-c:v	视频编码，建议用 libx264
-b:v	视频码率，例如 1500k 表示 1500kbps
-r	视频帧率
-profile:v	视频 profile，指定 baseline 将不编码 B 帧，TRTC 后端不支持 B 帧
-g	GOP 帧数间隔
-c:a	音频编码，建议用 libfdk_aac
-ac	声道数，填2或1
-b:a	音频码率
-f	指定格式，固定填 `flv`，发送到 TRTC 使用 FLV 容器封装

下面的例子是读取文件推到 TRTC，注意 URL 两边加引号。

```
ffmpeg -loglevel debug -re -i sample.flv -c:v libx264 -preset ultrafast
-profile:v baseline -g 30 -sc_threshold 0 -b:v 1500k -c:a libfdk_aac -ac
2 -b:a 128k -f flv 'rtmp://rtmp.rtc.qq.com/push/hello-string-room?
userid=rtmpForFfmpeg&sdkappid=140xxxxxx&usersig=xxxxxxxxxx'
```

其他端观看

下面是使用 Web 端 观看的效果，您也可以选择使用其他端观看。



FAQ

推流失败

常见原因

- 没买套餐包或过期。
- 签名错误或过期。
- 推了B帧（仪表盘上的现象是“推流一秒就结束”），可设置 baseline 编码。

其他原因

- 如果是嵌入式硬件设备推流，可能将 URL 截断。
- 推了 H.265，改为 H.264。
- 端上 set chunk 太大，改为 1360。

卡顿、花屏

观察腾讯云实时音视频 控制台仪表盘，查看推流帧率是否稳定，如果稳定则大概率是播放端的问题，建议排查播放端；**如果帧率不稳**可排查以下几点：

- 检查推流客户端本地 CPU、内存是否高负载。如果使用的是 OBS 推流，观察软件底部状态栏，有丢帧、网络、CPU、帧率等信息。
- 检查本地**网络带宽**是否足够。ping 推流的域名观察 RTT；使用 网络诊断工具 检测推流域名，查看带宽，最好能达到 10M。
- 推流端可尝试降低码率、帧率减少客户端压力，参考正文中 OBS 的设置，720p 建议码率 1500 Kbps。

延迟大

- 拉流端如果使用主播角色，延迟通常低于观众角色，如果不是主播角色可尝试对比一下观察是否有改善。
- 推流端本地编码和网络影响较大。可尝试不同平台测试，如果使用的是 OBS，可尝试 Windows 系统推流；ping 推流域名观察 RTT。

其他端看不到推的流

推流端使用的是字符串房间号，拉流端使用了数字房间号，修改拉流端，改为字符串房间号进房。

频繁断开重推

- 用户名重名，两处互踢引起，请确保单个 sdkappid 下的用户名 userid 全局唯一。
- 推了 B 帧，可设置 baseline 编码。

服务端回调

RTMP 推流用户也是 TRTC 房间中的一个用户，和其他端用户没有本质区别，参见 TRTC 事件回调 。

使用业务侧域名

设置业务侧域名 CNAME 到官方域名，后续也建议这样使用。

拉流分辨率和帧率限制

拉流的最大视频分辨率为 1920 × 1080，帧率为 30 fps。

实现云端切片

最近更新时间：2025-08-28 18:15:31

场景说明

在远程教育、秀场直播、视频会议、远程定损、金融双录、在线医疗等应用场景中，考虑取证、质检、审核、存档和回放等需求，常需要将整个视频通话或互动直播过程通过云端切片存储下来的情况。

云端切片费用

通过 TRTC 发起的云端审核，TRTC 仅向您收取音频切片和视频截图的费用，音频切片和视频截图的费用详情请参见 音频切片和视频截图计费说明 。

功能说明

通过 TRTC 的云端切片功能，您可以将房间中每一个用户的音视频流进行云端音频切片或者视频截图处理，无需客户端接入。

名词解释

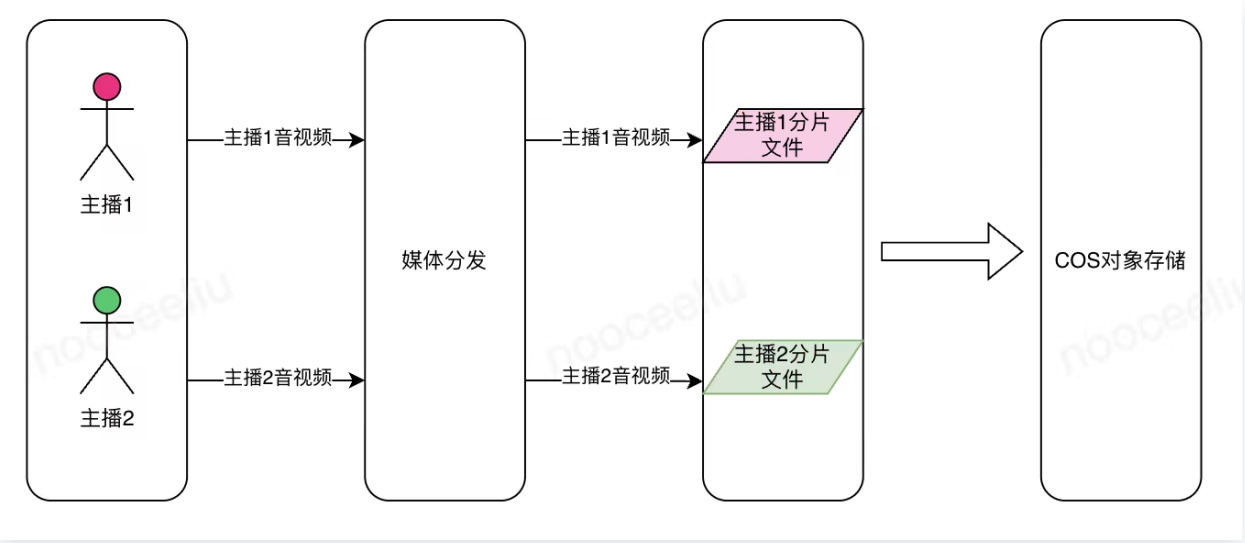
音频切片：将房间内的某一个用户的音频流按照一定的时间间隔进行切片处理，切片后为音频片段。
视频截图：将房间内的某一个用户的视频流按照一定的时间间隔进行截图处理，截图后为图片。
文件存储：支持将云端切片后的文件存储到对象存储 COS、AWS S3、阿里云 OSS。
回调通知：我们支持回调通知的能力，通过配置您的回调域名，云端切片的事件状态会通知到您的回调服务器。

音频切片及视频截图流程

如下图所示，房间里面主播1和主播2都上行了音视频流，假设您通过机器人订阅了主播1和主播2的音视频流，并设置同时进行视频截图和音频切片，后台会分别拉取主播1和主播2的音视频流，把音视频流切片成多个独立的音频文件和视频截图文件：

- 主播1的多个音频切片文件。
- 主播1的多个视频截图文件。
- 主播2的多个音频切片文件。
- 主播2的多个视频截图文件。

后台会把这些文件上传到您指定的云存储平台（对象存储 COS，AWS S3或者阿里云 OSS）。具体切片流程如下：



API 接口和切片并发限制

- 切片接口的调用频率限制为20 qps（如需提高 QPS 请 提交工单 ）。
- 单个接口超时时间为6秒。
- 单个应用下**默认并发切片支持500路**（API 切片任务的总和），超过并发限制的任务会失败，如需更多并发路数，请 提交工单 联系我们。
- 单次切片任务最大支持同时订阅的房间内主播数为25个，主播只上行音频也会单独占据一路。

切片任务执行流程

启动切片任务（ CreateCloudSlice ）

通过您的后台服务调用 REST API （ CreateCloudSlice ）来启动云端切片，需要重点关注参数——任务 ID（TaskId）；这个参数是本次切片任务的唯一标识，您需要保存下这个任务 ID 作为后续针对这个切片任务接口操作的输入参数。

1. 发起任务（ CreateCloudSlice ）

   发起云端切片任务的接口 CreateCloudSlice 中需要您指定分配切片机器人的进房参数 UserId 和 UserSig（ 如何获取 UserSig ），请不要与您房间内的正常主播或观众使用的 UserId 重复且不可与正在切片中的房间内指定的切片机器人 UserId 一致，否则会导致切片任务失败。
2. 指定切片用户（ SubscribeStreamUserIds ）

   您也可以通过参数 SubscribeStreamUserIds 指定想要切片或者不想切片的主播用户的黑白名单信息，当然我们也支持在切片的过程中进行更新操作。
3. 指定存储位置（ SliceStorageParams ）

   3.1 存储位置：支持存储至 AWS S3 或对象存储 COS，请通过在 SliceStorageParams 参数指定存储参数。

   3.2 切片格式：图片切片文件为 png， 音频切片文件为 ogg。

查询切片任务状态（ DescribeCloudSlice ）

如果需要，您可以调用该接口查询切片服务的状态。

修改切片任务参数（ ModifyCloudSlice ）

如果需要，您可以调用该接口修改切片服务的参数，例如修改指定切片用户（ SubscribeStreamUserIds ）。

停止切片任务（ DeleteCloudSliceTask ）

成功开启切片任务后，可以使用此接口来停止任务。

切片文件管理

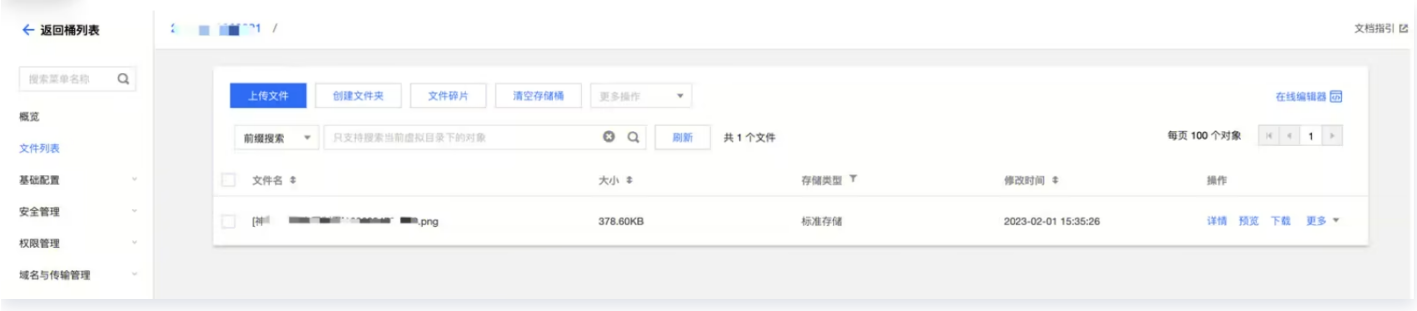
切片文件命名说明

音频切片默认命名为：
{您的桶名称}/{taskId}/{userId}/audios/{sdkappid}_{roomId}_{userid}_{UTC时间}.ogg
视频截图文件默认命名为：
{您的桶名称}/{taskId}/{userId}/images/{sdkappid}_{roomId}_{userid}_{UTC时间}.png
字段含义说明：

字段	含义
<taskId>	切片的任务 ID 。
<sdkappid>	切片任务的 SdkAppId。
<roomId>	切片的房间号。
<userid>	切片的用户 ID。
<UTC时间>	当前时间字符串 例如：20250106143143。

查找切片文件

登录 对象存储 COS 控制台，选择您指定的存储桶 Bucket 进行查找：
音频切片路径：{您的桶名称}/{taskId}/{userId}/audios/
视频截图路径：{您的桶名称}/{taskId}/{userId}/images/



切片回调事件

我们针对云端切片功能提供了多种的回调事件，帮助您及时了解切片任务的处理和完成情况。

切片回调地址配置

实时音视频 TRTC 控制台支持自助配置回调信息，配置完成后即可接收事件回调通知。详细操作指引请参见 回调配置 。

回调接口

您可以提供一个接收回调的 HTTP/HTTPS 服务网关来订阅回调消息。当相关事件发生时，云切片系统会回调事件通知到您的消息接收服务器。

事件回调消息格式

事件回调消息以 HTTP/HTTPS POST 请求发送给您的服务器，其中：
字符编码格式：UTF-8。
请求：body 格式为 JSON。
应答：HTTP STATUS CODE = 200，服务端忽略应答包具体内容，为了协议友好，建议客户应答内容携带 JSON： {"code":0}。

切片回调事件

参数说明

事件回调消息的 header 中包含以下字段：

字段名	值
Content-Type	application/json
Sign	签名值
SdkAppId	SDK 应用 ID 值

事件回调消息的 body 中包含以下字段：

字段名	类型	含义
EventGroupId	Number	事件组 ID， 云端切片固定为10。
EventType	Number	回调通知的事件类型。
CallbackTs	Number	事件回调服务器向您的服务器发出回调请求的 Unix 时间戳，单位为毫秒。
EventInfo	JSON Object	事件信息。

事件类型说明：

字段名	类型	含义
EVENT_TYPE_CLOUD_SLICE_START	1001	云端切片模块启动。
EVENT_TYPE_CLOUD_SLICE_STOP	1002	云端切片模块退出。
EVENT_TYPE_CLOUD_SLICE_UPLOAD_START	1003	云端切片文件上传任务启动，仅在选择对象存储时回调。
EVENT_TYPE_CLOUD_SLICE_FILE_INFO	1004	云端切片生成视频截图或者音频分片文件，上传成功后回调，仅在选择对象存储时回调。
EVENT_TYPE_CLOUD_SLICE_UPLOAD_STOP	1005	云端切片文件上传结束，仅在选择对象存储时回调。
EVENT_TYPE_CLOUD_SLICE_UPLOAD_ERROR	1006	云端切片投递模块发生错误。

事件信息说明：

字段名	类型	含义
RoomId	String/Number	房间名（类型与客户端房间号类型一致）。
EventTs	Number	事件发生的 Unix 时间戳，单位为秒 (不建议使用该字段，建议使用 EventMsTs)。
EventMsTs	Number	事件发生的 Unix 时间戳，单位为毫秒。
UserId	String	切片机器人的用户 ID。

TaskId	String	切片任务 ID，一次云端切片任务唯一的 ID。
Payload	JsonObject	根据不同事件类型定义不同。

事件类型为1001（EVENT_TYPE_CLOUD_SLICE_START）时 Payload 的定义：

字段名	类型	含义
Status	Number	• 0：代表切片模块启动成功。 • 1：代表切片模块启动失败。

```
{
    "EventGroupId": 10,
    "EventType": 1001,
    "CallbackTs": 1726125338219,
    "EventInfo": {
        "RoomId": "960025",
        "EventTs": 1726125338,
        "EventMsTs": 1726125338219,
        "UserId": "inspect",
        "TaskId": "-npVqpdU7sBobiK1iskE3BwlLIebCMrbKUbnL4K-rO+8oZWQndi
b9uvO4Deq9P1Na+sXGNGNuAE."
        "Payload": {
            "Status": 0
        }
    }
}
```

事件类型为1002（EVENT_TYPE_CLOUD_SLICE_STOP）时 Payload 的定义：

字段名	类型	含义
LeaveCode	Number	• 0：代表切片模块正常调用停止切片退出。 • 1：切片机器人被客户踢出房间。 • 2：客户解散房间。 • 3：服务器将切片机器人踢出。 • 4：服务器解散房间。 • 99：代表房间内除了切片机器人没有其他用户流，超过指定时间退出。

		• 100：房间超时退出。 • 101：同一用户重复进入相同房间导致机器人退出。


```
{
        "EventGroupId" :  10 ,
        "EventType" :     1002 ,
        "CallbackTs" :    1729601782073 ,
        "EventInfo" :     {
                "RoomId" :        "975626" ,
                "EventTs" :       "1729601782" ,
                "EventMsTs" :     1729601782073 ,
                "UserId" :        "SliceTaskDuration1-partner-robot" ,
                "TaskId" :        "-nHRjqhU7gTG0UIL-MquzG8D0Q+wehTbVTeeI
IK-rO+8oZWQndibtueIpQ8A0F3n9PEVRk0rngE." ,
                "Payload" :       {
                        "LeaveCode" :     99
                }
        }
}
```


事件类型为1003（EVENT_TYPE_CLOUD_SLICE_UPLOAD_START）时 Payload 的定义：

字段名	类型	含义
Status	Number	• 0：代表上传模块正常启动。 • 1：代表上传模块初始化失败。


```
{
    "EventGroupId" :  10 ,
    "EventType" :  1003 ,
    "CallbackTs" :  1726750023538 ,
    "EventInfo" :  {
        "RoomId" :  "295210" ,
        "EventTs" :  1726750023 ,
        "EventMsTs" :  1726750023538 ,
        "UserId" :  "inspect" ,
        "TaskId" :  "-nHwXIdU7mJvL22pFsXZ-v7OgEzq1OzbNXe9L4K-4pycoZWQndi
b3ZfzqN7Wq+AdiPLMBLxd0gE." ,
        "Payload" :  {
```

```
            "Status" :  0
        }
    }
}
```

事件类型为1004（EVENT_TYPE_CLOUD_SLICE_FILE_INFO ）时 Payload 的定义：

字段名	类型	含义
Payload	对象	切片结果。
StreamUserId	String	当前主播 ID。

```
{
    "EventGroupId" :  10 ,
    "EventType" :  1004 ,
    "CallbackTs" :  1726750309161 ,
    "EventInfo" :  {
        "RoomId" :  "963600" ,
        "EventTs" :  1730186326 ,
        "EventMsTs" :  1730186326109 ,
        "UserId" :  "SliceCustomUploadCase6-partner-robot" ,
        "StreamerUserId" :  "SliceCustomUploadCase6-user0" ,
        "TaskId" :  "-nHRjqhU7kGRgNhGI8Gq8JkT7wnFeUXbXG-6IYK-yb3t8ZWQndi
bhWMUwNxQwRlgnWuvEuBSqAE." ,
        "Payload" :  {
            "FileList" :  "20012177_2428_1929088_20250106143143.png"
        }
    }
}
```

事件类型为1005（EVENT_TYPE_CLOUD_SLICE_UPLOAD_STOP）时 Payload 的定义：

字段名	类型	含义
Status	Number	结束送审。

```
{
    "EventGroupId" :  10 ,
```

```
    "EventType" :  1005 ,
    "CallbackTs" :  1726751347072 ,
    "EventInfo" :  {
        "RoomId" :  "295211" ,
        "EventTs" :  1726751347 ,
        "EventMsTs" :  1726751347072 ,
        "UserId" :  "inspect" ,
        "TaskId" :  "-nHwXIdU7jx6C00Nt8Vr+3h4GwYdP7zbeHi9L4K-4pycoZWQndi
bqFeEaV4LvjFqSuQvaAkrNQE." ,
        "Payload" :  {
            "Status" :  0
        }
    }
}
```


事件类型为1006（EVENT_TYPE_CLOUD_SLICE_UPLOAD_ERROR）时 Payload 的定义：

字段名	类型	含义
Code	Number	COS 或者第三方存储错误码。
Message	String	COS 或者第三方存储错误消息。


```
{
    "EventGroupId" :  10 ,
    "EventType" :  1006 ,
    "CallbackTs" :  1726751347072 ,
    "EventInfo" :  {
        "RoomId" :  "295211" ,
        "EventTs" :  1726751347 ,
        "EventMsTs" :  1726751347072 ,
        "UserId" :  "inspect" ,
        "TaskId" :  "-nHwXIdU7jx6C00Nt8Vr+3h4GwYdP7zbeHi9L4K-4pycoZWQndi
bqFeEaV4LvjFqSuQvaAkrNQE." ,
        "Payload" :  {
            "Code" :  10002 ,
            "Message" :  "BadRequest"
        }
    }
```

```
}
```