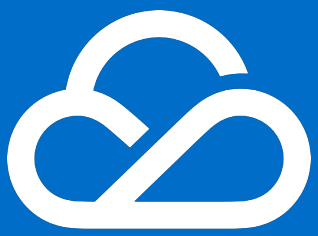
腾讯云点播

开发者手册

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

视频上传

视频上传综述

简介

互联网时代，精彩热门的视频资源越来越受到用户欢迎。APP 厂商如果能够将炙手可热的视频发布到互联网中，提供给亿万用户观看，将为自身创造巨大的价值。腾讯云点播提供了强大的视频管理、编辑和发布功能，让 APP 厂商快速将视频发布到腾讯云中，实现视频高效共享。

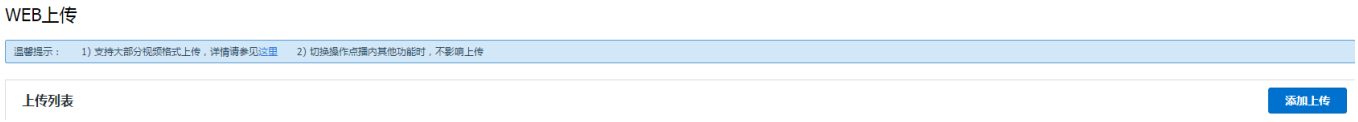
腾讯云点播针对不同的使用场景，提供了多种上传方式，协助 APP
厂商将本地视频资源上传到腾讯云。下面将分别介绍控制台上传、服务端上传和客户端上传三种方式。

控制台上传

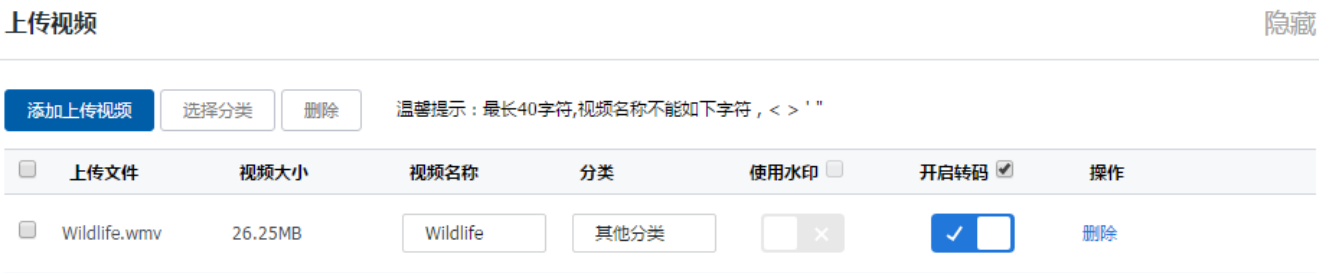
控制台上传是指 APP 管理员登录腾讯云点播控制台后，通过 Web
页面将本地视频上传到腾讯云点播系统的方式。控制台上传页面从这里进入。

上传步骤

- 点击【添加上传】按钮，在对话框中选择本地要上传的文件。



- 选定后可以指定上传的视频是否使用水印，上传成功后是否自动开启转码。



- 添加视频后，点击【开始上传】，即可将待上传视频列表中的视频上传到腾讯云点播系统。

方式特点

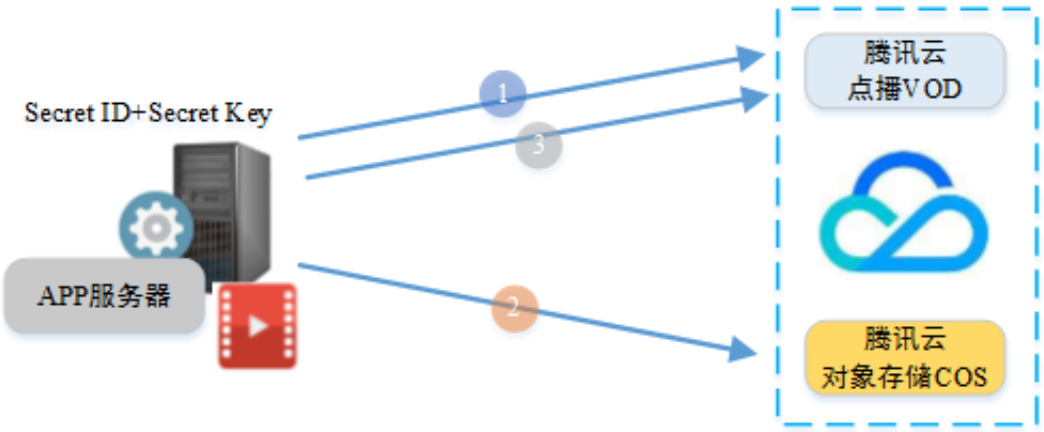
此上传方式操作简单，只需 APP 管理员登录腾讯云点播控制台页面即可进行视频文件的上传。按照控制台页面的指引，管理员可以指定视频的分类信息、是否转码等，并可在视频上传后为视频设置和修改封面。

服务端上传

服务端上传是指通过点播提供的 API 或 SDK，将保存在 APP 后台中的视频资源上传到腾讯云点播系统的方式。

上传步骤

服务端上传视频文件需要经过3个步骤，分别是向 VOD 发起上传、向 COS 上传文件和向 VOD 确认上传，具体各个步骤的含义和使用方法请参见服务端上传综述。



方式特点

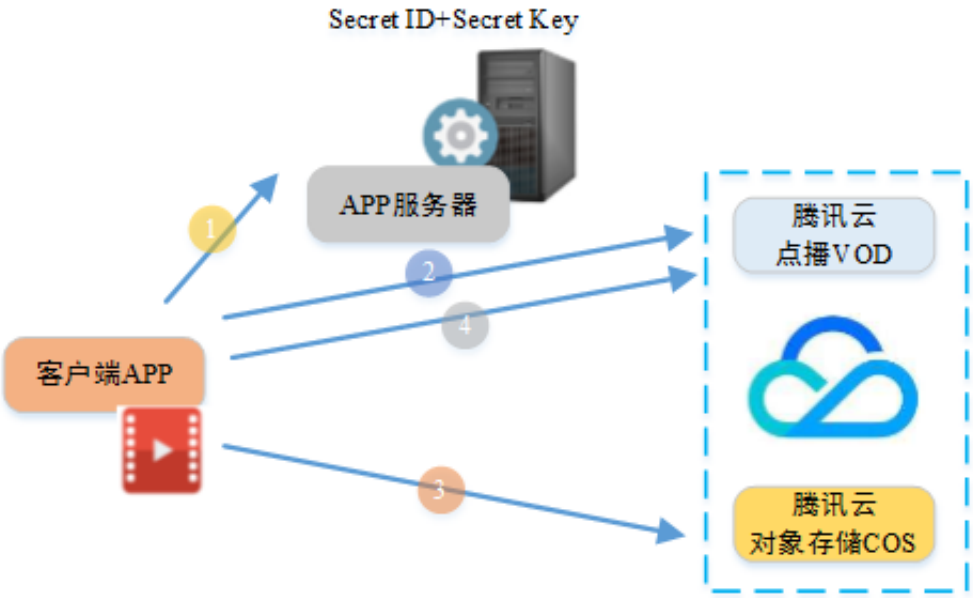
服务端上传适合 APP 将收集在服务器中的视频快速可靠地上传到腾讯云点播系统。相比于控制台上传，使用服务端上传 SDK 可以解放 APP 管理员的人工操作，从而提高发布和管理效率。

客户端上传

客户端上传是指通过点播提供的移动端（iOS 和 Android）和 Web 端 SDK，将客户端上的视频上传到腾讯云点播系统的方式。

上传步骤

客户端上传视频文件需要经过4个步骤，分别是从 APP 服务器获取上传签名、向 VOD 发起上传、向 COS 上传文件和向 VOD 确认上传。具体各个步骤的含义和使用方法请参见客户端上传综述。



方式特点

客户端上传可以让用户直接将移动设备中拍摄、录制或下载的视频快速上传到腾讯云点播系统。通过这样的方式，用户无需先把视频上传到 APP 服务器，再由服务器上传到腾讯云点播，从而有效简化了上传链路，节约了上传耗时和服务器上传带宽。

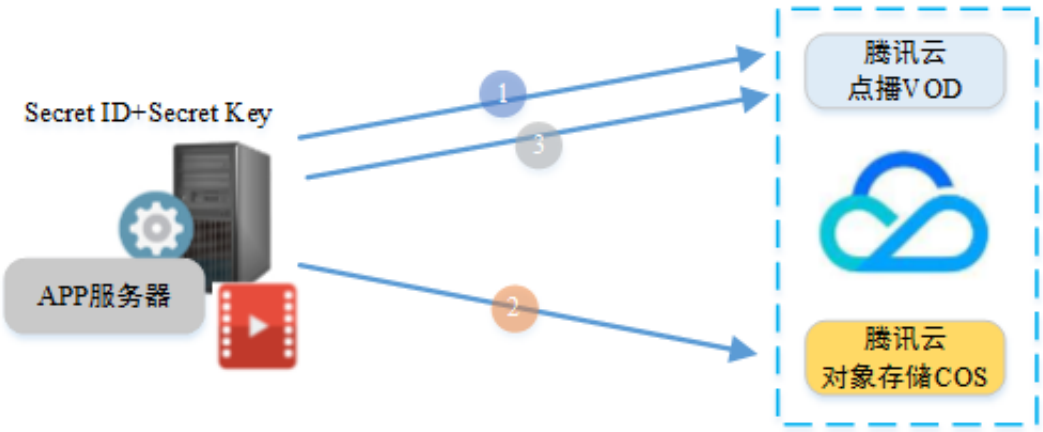
服务端上传

服务端上传综述

简介

APP 从各个渠道收集到大量精彩的视频，可能会直接将这些视频资源保存在自己的服务器中。这时，APP 会希望对视频进行加工处理（例如转码、截图、打水印等），并将处理后的视频发布到内容分发网络（CDN）中，以供用户观看并为自身创造价值。腾讯云点播支持用户将服务端上的视频文件直接上传到云端，并与点播的视频处理功能（比如转码、截图、防盗链等）无缝结合。

上传流程



上图为服务端上传流程示意图，主要有3个参与对象，分别是：APP 服务器、腾讯云点播系统 VOD 和腾讯云存储系统 COS。下面对流程图中的各步骤进行详细说明。

第一步：APP 服务器向 VOD 发起上传

APP服务器将文件上传到 COS 前，需要知道目标 COS Bucket、路径等信息。服务端向 VOD 发起上传，获取上传 COS 所需的信息。VOD 校验服务端的请求通过后为其分配上传信息。

服务端请求上传信息时，可以选择只上传视频（只请求视频文件的上传信息），或者同时上传视频和封面（同时请求视频文件和封面文件的上传信息）。

第二步：APP 服务器向 COS 上传文件

APP 服务器根据之前发起上传时获取的上传信息，调用 COS 的上传接口完成文件上传。

第三步：APP 服务器向 VOD 确认上传

APP 服务器成功上传文件到 COS 后，需要向 VOD 发起确认请求。收到确认上传的请求后，VOD 将返回视频的 fileId、视频播放地址、封面图片地址（如果有）等信息，并按照用户的要求对上传后的视频进行转码、打水印、截图等操作。

上传方式

API

- APP 服务器向 VOD 发起上传 API
- APP 服务器向 COS 上传文件 API
- APP 服务器向 VOD 确认上传 API

SDK

为了方便用户开发客户端上传功能，云点播提供了基于多语言平台 SDK ：

- PHP SDK
- Java SDK

SDK 基于云图 API 和 COS 的 SDK，封装了服务端和 VOD、COS 交互的细节，上传流程中的第一步到第三步，只需调用 SDK 中的一个接口即可全部完成。因此 强烈推荐用户在 APP 中参考点播提供的 SDK 来实现服务端上传。

Java SDK

简介

基于 COS 的 Java SDK 和云 API 的 SDK，VOD 提供了一个 Java 语言的 SDK，通过该 SDK，用户可以将服务端的视频和封面文件直接上传到腾讯云点播系统。

SDK获取与安装

可以通过以下方式获取JAVA SDK，开发者可以结合自身的情况，选择SDK源码、添加maven依赖项。

1. 获取源码

   从 Github 访问 >>

   点击下载 Java SDK >>

2. maven

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>vod_api</artifactId>
<version>1.0.0</version>
</dependency>
```

示例

upload 接口

```
public static void main(String[] args) {
  try {
  VodApi vodApi = new VodApi("your secretId", "your secretKey");
  //设置签名过期时长
  //VodApi vodApi = new VodApi("your secretId", "your secretKey", 24 * 3600);
```

```
  vodApi.upload("videos/Wildlife.wmv", "videos/Wildlife-cover.png");
  } catch(Exception e) {
  //打日志
  log.error("上传视频失败", e)
  }
}
```

PHP SDK

简介

基于 COS 的 PHP SDK 和云 API 的 SDK，VOD 提供了一个 PHP语言的 SDK，通过该 SDK，用户可以将服务端的视频和封面文件直接上传到腾讯云点播系统。

下载地址

- 从 Github 访问 >>
- 点击下载 PHP SDK >>

使用方式

With Composer

- 引入依赖

```
{
  "require": {
  "qcloud/vod-sdk-v5": "v1.2.1"
  }
}
```

- 调用示例

```
<?php
require 'vendor/autoload.php';

use Vod\VodApi;

VodApi::initConf("your secretId", "your secretKey");
```

```
$result = VodApi::upload(
  array (
  'videoPath' => './test/Wildlife.wmv',
  ),
  array (
  'videoName' => 'WildAnimals',
// 'procedure' => 'myProcedure',
// 'sourceContext' => 'test',
  )
);
echo "upload to vod result: " . json_encode($result) . "\n";
```

上传成功后将获取文件的播放地址和 fileid

Without Composer

- 复制src文件下的源码和test/non-composer文件的cos-sdk-v5、qcloudapi-sdk-php到同级目录
- 调用示例

```
<?php
require './cos-sdk-v5/cos-autoloader.php';
require './qcloudapi-sdk-php/src/QcloudApi/QcloudApi.php';
require './src/Vod/VodApi.php';
require './src/Vod/Conf.php';

use Vod\VodApi;

VodApi::initConf("your secretId", "your secretKey");

$result = VodApi::upload(
  array (
  'videoPath' => '../Wildlife.wmv',
  ),
```

```
  array (
  'videoName' => 'WildAnimals',
// 'procedure' => 'myProcedure',
// 'sourceContext' => 'test',
  )
);
echo "upload to vod result: " . json_encode($result) . "\n";
```

上传成功后将获取文件的播放地址和 fileid

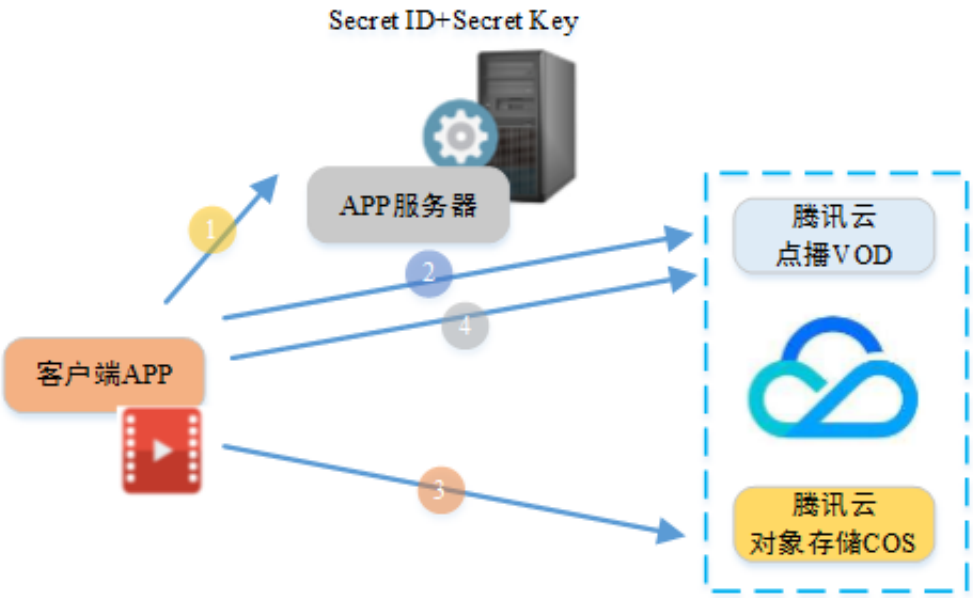
客户端上传

客户端上传综述

简介

随着终端用户个性化的需求愈加强烈，简单的文字交互已经不能满足用户的分享诉求。腾讯云点播支持用户将客户端上的视频文件直接上传到云端，并与点播的其它业务功能（比如转码、截图、防盗链等）无缝结合。

上传流程



上图为客户端上传流程示意图，主要有4个参与对象，分别是：客户端、APP 服务器、腾讯云点播系统 VOD 和腾讯云存储系统 COS。下面对流程图中的各步骤进行详细说明。

第一步：客户端从 APP 服务器获取上传签名

客户端上传文件到腾讯云之前，VOD 需要检查客户端的上传行为是否已获得 APP 服务器的授权。因此，客户端必须先从 APP 服务器获取上传签名，并在请求上传信息时将该签名传递给 VOD。VOD 校验签名，只有在校验结果为“签名合法”的情况下，才允许客户端进行后续上传操作。

APP 服务器生成签名的方式参照客户端上传签名。

注意：千万不要 将 API 密钥（即 Secret ID 和 Secret

Key）直接暴露给客户端，该信息一旦泄露可能引发严重的安全问题。API 密钥应只由 APP 服务器保管。

第二步：客户端向 VOD 发起上传

客户端将文件上传到 COS 前，需要知道目标 COS Bucket、路径等信息。客户端向 VOD 发起上传，获取上传 COS 所需的信息。VOD 签名校验通过后为其分配上传信息。

客户端请求上传信息时，可以选择只上传视频（只请求视频文件的上传信息），或者同时上传视频和封面（同时请求视频文件和封面文件的上传信息）。

第三步：客户端向 COS 上传文件

客户端根据之前发起上传时获取的上传信息，调用 COS 的上传接口完成文件上传。

第四步：客户端向 VOD 确认上传

客户端成功上传文件到 COS 后，需要向 VOD 发起确认请求。收到确认上传的请求后，VOD 将返回视频的 fileId、视频播放地址、封面图片地址（如果有）等信息，并按照用户的要求对上传后的视频进行转码、打水印、截图等操作。

上传方式

SDK

为了方便用户开发客户端上传功能，云点播提供了多种平台的 SDK：

- Android SDK
- iOS SDK
- Web SDK

SDK 封装了客户端和 VOD、COS 交互的细节，上传流程中的第二步到第四步，只需调用 SDK 中的一个接口即可全部完成。因此 强烈推荐 用户在 APP 中集成点播提供的 SDK 来实现客户端上传。

客户端上传签名

简介

客户端在发起上传前，需要向 APP 服务器请求上传签名（背景请参考客户端上传综述）。如果 APP 服务器允许客户端上传，则应按照本文介绍的签名规则为客户端生成一个上传签名。客户端执行上传操作时，必须携带该签名，让腾讯云点播验证客户端的上传是否被授权。

符号说明

签名参数

生成签名需要APP服务器确定4个必选参数（以及任意个可选参数），用户构造签名的明文串，参数如下：

必选参数

参数名称	必选	类型	说明
secretId	是	String	构造签名明文串的参数，云API 密钥中的 Secret ID。
currentTimeStamp	是	Integer	构造签名明文串的参数，当前 Unix 时间。
expireTime	是	Integer	构造签名明文串的参数，签名到期 Unix 时间。 expireTime = currentTimeStamp + 签名有效时长 签名有效时长最大取值为7776000，即90天。
random	是	Integer	构造签名明文串的参数，无符号32位随机数。

可选参数

参数名称	必选	类型	说明
classId	否	Integer	视频文件分类，默认为0。
procedure	否	String	视 频后 续任务操 作，详见任务流综述。

签名生成步骤

APP 服务器生成一个上传签名包括以下步骤。

第一步：获取 API 密钥

在 云 API 密钥 管理里获取或者创建一个 SecretID，并拿到其对应的 SecretKey。

第二步：拼接明文串

按照 URL QueryString 的格式要求生成签名明文串 Original[注 1]，格式如下：

secretId
=[secretId]&currentTimeStamp=[currentTimeStamp]&expireTime=[expireTime]&random=[random]

第三步：将明文串转为最终签名

生成签名明文串 Original 后，用已获取的 secretKey 对明文串进行 HMAC-SHA1 加密，得到 SignatureTmp[注 2]：

SignatureTmp = HMAC-SHA1(secretKey, Original)

将密文串 SignatureTmp 放在明文串 Original 前面，拼接后进行 Base64 编码，得到最终的签名 Signature：

Signature = Base64(append(SignatureTmp, Original))

注意事项

注 1

签名明文串 Original 需要满足：

- 至少包含 secretId， currentTimeStamp，expireTime 和 random 四个必选参数，可包含任意多个选填参数
- 参数值必须经过 URL 编码，否则可能导致 Query String 解析失败

建议 ：直接操作字符串的方式生成 Original 很容易出错，大多数编程语言均提供了相关类库帮助开发者完成 QueryString 的拼接和编码。因此，建议开发者尽量使用标准的类库来构造 Original。

注 2

签名密文串 SignatureTmp 的输出结果是20字节的二进制串。

多语言签名示例

此处提供多种语言平台的客户端签名生成示例，APP 可以根据开发语言的偏好参考对应的示例：

客户端上传签名示例

签名生成和校验工具

此处提供了一组签名工具，帮助用户验证自己生成的签名是否正确：

点播客户端上传-签名生成工具：在页面上填写签名所需要的参数和密钥，即可生成一个合法的签名。

点播客户端上传-签名校验工具：对一个合法的签名进行解析，获得生成签名时所使用的参数。

客户端上传签名示例

PHP 签名示例

```
<?php
// 确定APP的云API密钥
$secret_id = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
$secret_key = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";

// 确定签名的当前时间和失效时间
$current = time();
$expired = $current + 86400; // 签名有效期：1天

// 向参数列表填入参数
$arg_list = array(
  "secretId" => $secret_id,
  "currentTimeStamp" => $current,
  "expireTime" => $expired,
  "random" => rand());

// 计算签名
$orignal = http_build_query($arg_list);
$signature = base64_encode(hash_hmac('SHA1', $orignal, $secret_key, true).$orignal);

echo $signature;
echo "\n";
?>
```

Java 签名示例

```
import java.util.Random;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Encoder;
```

```
class Signature {
  private String secretId;
  private String secretKey;
  private long currentTime;
  private int random;
  private int signValidDuration;

  private static final String HMAC_ALGORITHM = "HmacSHA1";
  private static final String CONTENT_CHARSET = "UTF-8";

  public static byte[] byteMerger(byte[] byte1, byte[] byte2) {
  byte[] byte3 = new byte[byte1.length + byte2.length];
  System.arraycopy(byte1, 0, byte3, 0, byte1.length);
  System.arraycopy(byte2, 0, byte3, byte1.length, byte2.length);
  return byte3;
  }

  public String getUploadSignature() throws Exception {
  String strSign = "";
  String contextStr = "";

  long endTime = (currentTime + signValidDuration);
  contextStr += "secretId=" + java.net.URLEncoder.encode(secretId, "utf8");
  contextStr += "&currentTimeStamp=" + currentTime;
  contextStr += "&expireTime=" + endTime;
  contextStr += "&random=" + random;

  try {
  Mac mac = Mac.getInstance(HMAC_ALGORITHM);
  SecretKeySpec secretKey = new SecretKeySpec(this.secretKey.getBytes(CONTENT_CHARSET),
mac.getAlgorithm());
  mac.init(secretKey);
```

```
byte[] hash = mac.doFinal(contextStr.getBytes(CONTENT_CHARSET));
byte[] sigBuf = byteMerger(hash, contextStr.getBytes("utf8"));
strSign = new String(new BASE64Encoder().encode(sigBuf).getBytes());
strSign = strSign.replace(" ", "").replace("\n", "").replace("\r", "");
} catch (Exception e) {
throw e;
}
return strSign;
}

public void setSecretId(String secretId) {
this.secretId = secretId;
}

public void setSecretKey(String secretKey) {
this.secretKey = secretKey;
}

public void setCurrentTime(long currentTime) {
this.currentTime = currentTime;
}

public void setRandom(int random) {
this.random = random;
}

public void setSignValidDuration(int signValidDuration) {
this.signValidDuration = signValidDuration;
}
}

public class Test {
public static void main(String[] args) {
Signature sign = new Signature();
```

```
  sign.setSecretId("个人API密钥中的Secret Id");
  sign.setSecretKey("个人API密钥中的Secret Key");
  sign.setCurrentTime(System.currentTimeMillis() / 1000);
  sign.setRandom(new Random().nextInt(java.lang.Integer.MAX_VALUE));
  sign.setSignValidDuration(3600 * 24 * 2);

  try {
  String signature = sign.getUploadSignature();
  System.out.println("signature : " + signature);
  } catch (Exception e) {
  System.out.print("获取签名失败");
  e.printStackTrace();
  }
  }
}
```

注意

- 需要导入第三方包 javax-crpyto.jar 和 sun.misc.BASE64Encoder.jar。
- 如果导入第三方包 sun.misc.BASE64Encoder.jar 出现 Access restriction 的错误，可以通过调整错误级别解决：

  Windows -> Preferences -> Java -> Compiler -> Errors/Warnings -> Deprecated and trstricted API -> Forbidden reference (access rules): -> change to warning

Node.js 签名示例

```
var querystring = require("querystring");
var crypto = require('crypto');

// 确定APP的云API密钥
```

```
var secret_id = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
var secret_key = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";

// 确定签名的当前时间和失效时间
var current = parseInt((new Date()).getTime() / 1000)
var expired = current + 86400; // 签名有效期：1天

// 向参数列表填入参数
var arg_list = {
  secretId : secret_id,
  currentTimeStamp : current,
  expireTime : expired,
  random : Math.round(Math.random() * Math.pow(2, 32))
}

// 计算签名
var orignal = querystring.stringify(arg_list);
var orignal_buffer = new Buffer(orignal, "utf8");

var hmac = crypto.createHmac("sha1", secret_key);
var hmac_buffer = hmac.update(orignal_buffer).digest();

var signature = Buffer.concat([hmac_buffer, orignal_buffer]).toString("base64");

console.log(signature);
```

C# 签名示例

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
```

```
class Signature
{
  public string m_strSecId;
  public string m_strSecKey;
  public int m_iRandom;
  public long m_qwNowTime;
  public int m_iSignValidDuration;
  public static long GetIntTimeStamp()
   {
  TimeSpan ts = DateTime.UtcNow - new DateTime(1970, 1, 1);
  return Convert.ToInt64(ts.TotalSeconds);
  }
  private byte[] hash_hmac_byte(string signatureString, string secretKey)
   {
  var enc = Encoding.UTF8; HMACSHA1 hmac = new HMACSHA1(enc.GetBytes(secretKey));
  hmac.Initialize();
  byte[] buffer = enc.GetBytes(signatureString);
  return hmac.ComputeHash(buffer);
  }
  public string GetUploadSignature()
   {
  string strContent = "";
  strContent += ("secretId=" + Uri.EscapeDataString((m_strSecId)));
  strContent += ("&currentTimeStamp=" + m_qwNowTime);
  strContent += ("&expireTime=" + (m_qwNowTime + m_iSignValidDuration));
  strContent += ("&random=" + m_iRandom);

  byte[] bytesSign = hash_hmac_byte(strContent, m_strSecKey);
  byte[] byteContent = System.Text.Encoding.Default.GetBytes(strContent);
  byte[] nCon = new byte[bytesSign.Length + byteContent.Length];
  bytesSign.CopyTo(nCon, 0);
  byteContent.CopyTo(nCon, bytesSign.Length);
  return Convert.ToBase64String(nCon);
  }
```

```
}
class Program
{
  static void Main(string[] args)
   {
  Signature sign = new Signature();
  sign.m_strSecId = "个人API密钥中的Secret Id";
  sign.m_strSecKey = "个人API密钥中的Secret Key";
  sign.m_qwNowTime = Signature.GetIntTimeStamp();
  sign.m_iRandom = new Random().Next(0, 1000000);
  sign.m_iSignValidDuration = 3600 * 24 * 2;

  Console.WriteLine(sign.GetUploadSignature());
  }
}
```

移动端上传 SDK

概述

目前，腾讯云点播为移动端提供了Android，iOS 两种 SDK。

移动端 SDK 不仅提供了上传视频的 API，还针对用户的需求提供了丰富的视频编辑方面的 API，包括视频裁剪、拼接、滤镜、字幕等等。

Android SDK

- 发布上传视频
- 视频编辑
- 视频拼接
- SDK 下载地址

iOS SDK

- 发布上传视频
- 视频编辑
- 视频拼接
- SDK 下载地址

Web 端上传 SDK

简介

用于浏览器的客户端上传 SDK，可向腾讯云点播系统上传视频和封面文件。

集成方式

开发环境

- 使用 SDK 需要浏览器支持 HTML 5
- 需要 APP 服务器派发客户端上传签名，生成签名的方法请见上传签名

集成

在页面引入ugcuploader.js即可。

```
<script src="//imgcache.qq.com/open/qcloud/js/vod/sdk/ugcUploader.js"></script>
```

Demo

http://video.qcloud.com/sdk/ugcuploader.html

上传步骤

第一步：获取上传签名

```
var getSignature = function(callback){
  $.ajax({
  url: 'yourinterface', //服务器获取客户端上传签名的URL
  type: 'POST',
  dataType: 'json',
  success: function(result){
```

```
  //result.returnData.signature为获取到的签名
  callback(result.returnData.signature);
  }
  });
};
```

第二步：指定上传目标

上传目标有视频和封面信息。如果只上传视频，则封面参数可不填。

参数名称	必填	类型	参数描述
videoFile	是	File	要上传的视频文件
coverFile	否	File	要上传的封面文件
getSignature	是	Function	获取签名的回调函数
success	否	Function	上传成功的回调函数
error	否	Function	上传失败的回调函数
progress	否	Function	上传进度的回调函数
finish	否	Function	上传结果的回调函数

回调函数说明

函数名	说明	参数类型	参数含义
getSignature	获取签名回调	Function	callback：把获取到的签名作为 callback 函数的参数，即callback(signature);
success	上传成功回调	Object	type：上传成功的文件种类，'video'（视频）或者'cover'（封面）
error	上传失败回调	Object	type：上传失败的文件种类，'video'（视频）或者'cover'（封面）
progress	上传进度回调	Object	type：上传进行中的文件种类，'video'（视频）或者'c

函数名	说明	参数类型	参数含义
			over'（封面） name：上传中的文件名 curr：文件上传进度
finish	上传结果回调	Object	fileId：视频文件 ID videoName：视频名称 videoUrl：视频播放地址 coverName：封面名称 coverUrl：封面展示地址

第三步：执行上传操作

仅上传视频

```
qcVideo.ugcUploader.start({
  videoFile: videoFile,
  getSignature: getSignature,
  success: function(result){
  console.log('上传成功的文件类型：' + result.type);
  },
  error: function(result){
  console.log('上传失败的文件类型：' + result.type);
  console.log('上传失败的原因：' + result.msg);
  },
  progress: function(result){
  console.log('上传进度的文件类型：' + result.type);
  console.log('上传进度的文件名称：' + result.name);
  console.log('上传进度：' + result.curr);
  },
  finish: function(result){
  console.log('上传结果的fileId：' + result.fileId);
  console.log('上传结果的视频名称：' + result.videoName);
  console.log('上传结果的视频地址：' + result.videoUrl);
  }
});
```

同时上传视频和封面

```
qcVideo.ugcUploader.start({
  videoFile: videoFile,
  coverFile: coverFile,
  getSignature: getSignature,
  success: function(result){
  console.log('上传成功的文件类型：' + result.type);
  },
  error: function(result){
  console.log('上传失败的文件类型：' + result.type);
  console.log('上传失败的原因：' + result.msg);
  },
  progress: function(result){
  console.log('上传进度的文件类型：' + result.type);
  console.log('上传进度的文件名称：' + result.name);
  console.log('上传进度：' + result.curr);
  },
  finish: function(result){
  console.log('上传结果的fileId：' + result.fileId);
  console.log('上传结果的视频名称：' + result.videoName);
  console.log('上传结果的视频地址：' + result.videoUrl);
  console.log('上传结果的封面名称：' + result.coverName);
  console.log('上传结果的封面地址：' + result.coverUrl);
  }
});
```

任务流

任务流综述

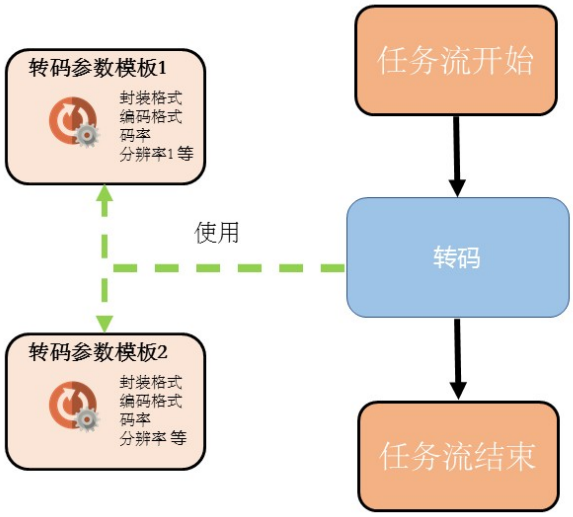
任务流简介

不同的业务场景，要求我们对视频文件进行不同的处理。比如一家游戏直播网站，会将直播的内容录制成为点播视频，然后转码，再截取图片作为封面；而一个做教育视频的网站，则是在服务端上传完视频后进行转码，同时对视频进行打水印或加密以保护产权。对于某个特定的场景，视频处理流程一般是固定的，如果开发者对每一个视频的每一次处理都分别调用一次服务端 API，那么开发过程会变得很复杂。
为了解决这一问题，云点播提供了 视频处理任务流 机制：云点播根据最常用的业务场景，预先将一系列具体的视频处理流程定义为任务流，开发者只需要调用相关的任务流接口即可对指定视频进行一系列的处理操作，而无需分别调用独立的视频处理接口。

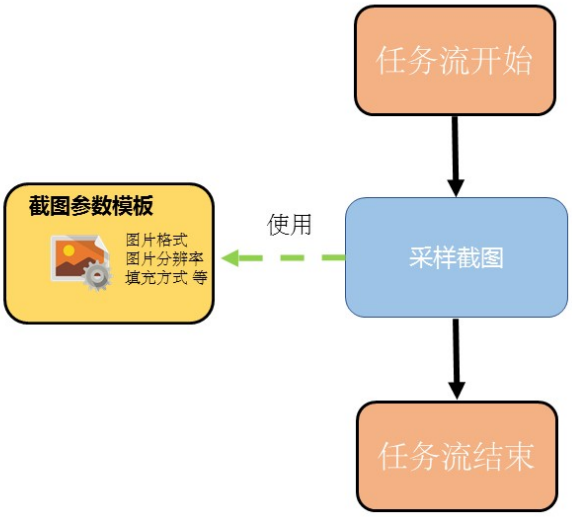
任务流示意图

任务流机制的实现基于视频处理参数模版，他们之间的关系举例如下：

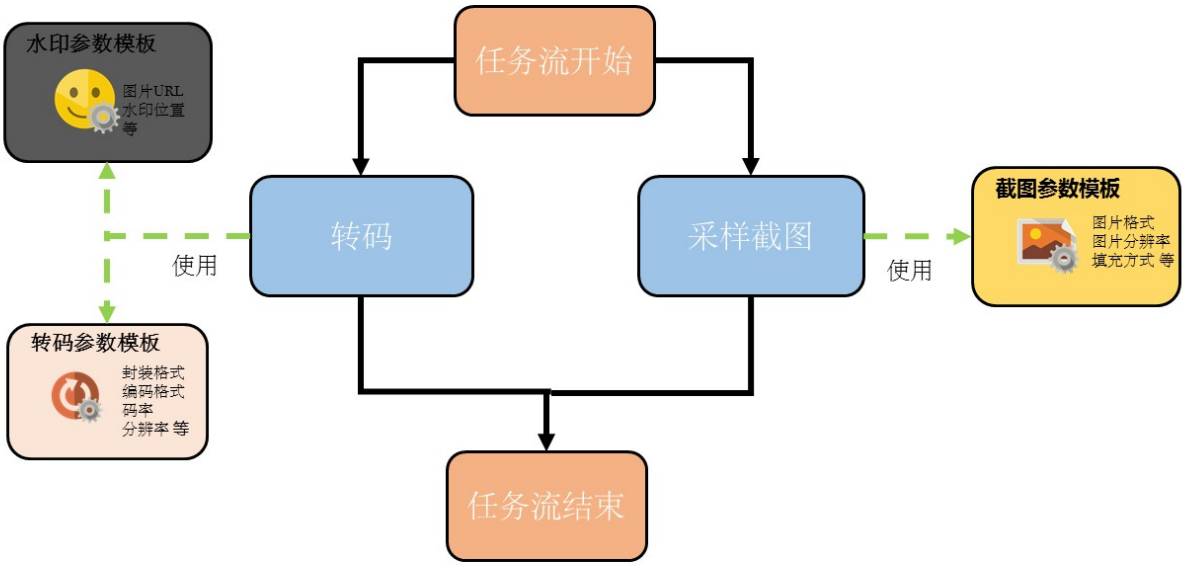
1. 任务流：转码



2. 任务流：采样截图



3. 任务流：转码-水印-采样截图



预设任务流

云点播的预设任务流包括下述几种使用场景。

场景	使用方式	示例
服务端上传	在发起上传API中携带任务流参数：procedure。详情见VOD发起上传API	procedure=QCVB_SimpleProcessFile(1,0,0,0)
客户端上传	在上传签名中携带任务流参数：procedure。详情见客户端上传签名	procedure=QCVB_SimpleProcessFile(1,0,0,0)
直播录制	推流时携带任务流参数：vod_procedure	vod_procedure=QCVB_SimpleProcessFile(1,0,0,0)

对已有视频进行处理

依照指定任务流处理视频

参见文档依照指定流程处理视频文件

对视频文件进行处理

参见文档对视频文件进行处理

获取任务流结果

1. 任务流状态变更（或者处理完成）会触发事件通知-任务流状态变更通知。APP后台可据此监听任务流的执行状态。
2. 任务流执行完毕之后，任务结果可以从GetVideoInfo中获取。

预设任务流模版

简介

预设任务流是点播系统内置的通用化的任务流模板，目前暂时只有QCVB_SimpleProcessFile。

注意：任务流的概念以及用法可以参见任务流综述。

内置任务流列表

任务流名称	任务流功能
QCVB_SimpleProcessFile	转码，设置水印，采样截图，使用截图设置视频封面

QCVB_SimpleProcessFile

简单的ProcessFile，支持转码，水印，设置封面，采样截图。
QCVB_SimpleProcessFile(transcodeDefinition, watermarkDefinition, coverBySnapshotDefinition, sampleSnapshotDefinition)

参数说明

参数名称	必填	类型	说明
transcodeDefinition	是	Integer/Array	转码模板号，参见转码参数模板。
watermarkDefinition	是	Integer	水印模板号，参见水印参数模板
coverBySnapshotDefinition	是	Integer	使用截图设置视频封面模板号，参见截图参数模板。
sampleSnapshotDefinition	是	Integer	采样截图模板号，参见采样截图参数模板

使用示例

获取元信息示例

QCVB_SimpleProcessFile()

以下示例的含义是：

1. 对视频获取元信息，即视频的长、宽、码率等数据；

使用控制台配置的规格转码示例

QCVB_SimpleProcessFile(1, 1, 10, 10)

以下示例的含义是：

1. 对视频进行转码，转码模板为控制台配置的模板；
2. 转码过程需要设置水印，水印模板号为1，即默认模板；
3. 对视频设置封面，封面图片为视频首帧，截图模板号为10；
4. 对视频进行采样截图，采样截图的模板号为10；

指定视频规格转码示例

QCVB_SimpleProcessFile({210,220}, 10, 10, 10)

以下示例的含义是：

1. 对视频文件进行转码，目标输出模板为210，220；
2. 转码过程需要设置水印，水印模板号为10；
3. 对视频设置封面，封面图片为视频首帧，截图模板号为10；
4. 对视频进行采样截图，采样截图的模板号为10；

获取结果

1. 任务流状态变更（或者处理完成）会触发事件通知-任务流状态变更通知。APP后台可据此监听任务流的执行状态。
2. 任务流执行完毕之后，任务结果可以从GetVideoInfo中获取。

视频处理参数模板

参数模板综述

参数模板简介

云点播支持多种视频处理操作，包括转码、加密、打水印、剪辑和截图等，每一种操作都可能会使用到多个参数。以转码为例，涉及到的参数就有封装格式、码率、分辨率和帧率，参数组合的总数量有数十种之多。大部分情况下，开发者只需要用到少数几种参数组合，而过多的参数会给开发者带来不必要的麻烦。为了简化开发者的使用，云点播提供了 视频处理参数模板 机制：对于一种视频处理操作，开发者可以将一组参数组合定义为一个参数模板（即创建模版），腾讯云为该模板分配唯一的模版
ID。开发者在调用视频处理接口时仅需指定模板 ID，而无需传递复杂的参数组合。

预设参数模板

对于每一种视频处理操作，往往存在几组通用的参数组合，能够满足大多数应用场景。云点播将这类最常用的参数组合预设为内置参数模板，使得开发者无需创建模版即可直接使用。目前云点播提供以下的预设参数模版：

- 转码参数模板；
- 水印参数模板；
- 截图参数模板；
- 采样截图参数模板；
- 雪碧图参数模板；

自定义参数模板

当云点播的预设参数模板无法满足需求时，开发者可以通过服务端 API
接口自定义参数模板。当前支持自定义的模版类型有：

- 转码模板

自定义转码参数模板

- 创建转码模板；
- 更新转码模板；
- 查询转码模板；
- 拉取转码模板列表；
- 删除转码模板；

转码参数模板

转码参数模板介绍

对音视频文件进行转码，涉及很多复杂参数，开发者可以将这些参数组合成参数模板，在调用点播相关接口来处理视频时，可以仅指定参数模板，而无需传递一系列复杂参数。
转码参数模板会在以下几种场景中用到：

1. 对视频文件进行处理接口（ProcessFile）中的转码控制参数。

转码模板通用参数说明

参数名称	说明	取值范围
模板ID（definition）	一个模板的唯一标志，由点播系统分配	大于0
封装格式（container）	视频的封装格式	HLS/MP4/FLV
是否去除视频流（isFiltrateVideo）	若去除，则为纯音频文件	1为去除视频流，0为保留，默认为0
是否去除音频流（isFiltrateAudio）	若去除，则为纯视频文件	1为去除视频流，0为保留，默认为0

转码模板视频参数说明

参数名称	说明	取值范围
编码器（codec）	能够对视频进行压缩或者解压缩的程序或者设备	libx264（H264编码），libx265（H265编码）
帧率（fps）	用于测量显示帧数的量度	24、25、30。单位hz
分辨率开启自适应（resolutionSelfAdapting）	若开启，则宽的值用于较长边，高的值用于较短边	1为开启，0为关闭。 默认为1
宽（width）	视频播放时的宽	[128，1920]。 单位：px
高（height）	视频播放时的高	[128，最大1920]。 单位：px
色度空间（colorSpace）	在某些标准下用通常可接受的方式对彩色加以说明	只支持 YUV420P
位深（bitDepth）	每一个像素在计算机中所使用的位数就是位深	只支持 8
码率模式（rateControlMode）	视频码率的格式	只支持 ABR
码率（bitrate）	视频流中，单位时间内传输或处理的	[128，10000]。单位kbps

参数名称	说明	取值范围
	比特的数量	

转码模板音频参数说明

参数名称	说明	取值范围
编码器（codec）	能够对音频进行压缩或者解压缩的程序或者设备	AAC/MP3
码率（bitrate）	音频流中，单位时间内传输或处理的比特的数量	[26，256]。单位kps
通道（soundSystem）	音频通道个数	1或者2
采样频率（sampleRate）	录音设备在一秒钟内对声音信号的采样次数，采样频率越高声音的还原就越真实越自然	44100,48000。单位hz

点播系统标准转码输出模板

为简化调用，点播预定义了一系列标准转码模板，每一个模板都对应一组转码参数。
标准转码模板中，有一些参数都是统一的：

1. 不会过滤视频流，音频流，
2. 开启分辨率自适应，
3. 色度空间为YUV420P，
4. 位深为8
5. 码率模式为ABR
6. 音频都是双通道的

模板ID	封装格式	视频参数	音频参数
编码器	帧率	宽	高
10	MP4	H.264	24

20	MP4	H.264	24
30	MP4	H.264	24
40	MP4	H.264	24
210	HLS	H.264	24
220	HLS	H.264	24
230	HLS	H.264	24
240	HLS	H.264	24
10046	FLV	H.264	24

10047	FLV	H.264	24
10048	FLV	H.264	24
10049	FLV	H.264	24

截图参数模板

截图参数模板介绍

对音视频文件进行截图，涉及多个参数，开发者可以将这些参数组合成参数模板，在调用点播相关接口来处理视频时，可以仅指定参数模板，而无需传递一系列参数。
截图参数模板会在以下几种场景中用到：

1. 按时间点截图接口（CreateSnapshotByTimeOffset）；
2. 对视频文件进行处理接口（ProcessFile）中的使用截图设置封面参数。

截图模板参数说明

- 模板ID（definition）：一个模板的唯一标志，由点播系统分配。
- 截图输出格式（format）：目前支持jpg，png。
- 截图宽度（width）：截图宽度，单位：px。
- 截图高度（height）：截图高度，单位：px。
- 截图填充方式（type）：目前仅支持拉伸填充。

内置截图参数模板

模板ID	截图输出格式	截图宽度	截图高度	截图填充方式
10	JPG	同源	同源	N/A

采样截图参数模板

采样截图参数模板介绍

对音视频文件进行截图，涉及多个参数，开发者可以将这些参数组合成参数模板，在调用点播相关接口来处理视频时，可以仅指定参数模板，而无需传递一系列参数。

截图模板参数说明

- 模板ID（definition）：一个模板的唯一标志，由点播系统分配。
- 截图输出格式（format）：目前支持jpg，png。
- 截图宽度（width）：截图宽度，单位：px。
- 截图高度（height）：截图高度，单位：px。
- 采样方式（type）：采样方式，有Percent和Time两种。Percent为根据百分比间隔采样，Time为根据时间间隔采样。
- 截图间隔（interval）：若type为Percent，表示多少百分比一张图；
  若type为Time，表示多少时间间隔一张图，单位秒。 第一张图均为视频首帧

使用该参数模板的场景包括：

1. 对视频文件进行处理接口（ProcessFile）中的采样截图控制参数；
2. 视频转码接口（ConvertVodFile），当请求参数中指定isScreenshot时（此时系统默认使用模板ID 10）。

除了内置采样截图参数模板之外，如果您存在定制化的截图规格，可以提需求工单，点播后台可以为您配置。

内置采样截图参数模板

模板ID	截图输出格式	宽度	高度	采样方式	截图间隔
10	JPG	同源	同源	Percent	10%

雪碧图参数模板

ImageSprite，俗称雪碧图，是客户端性能优化的一种重要技术。视频截取雪碧图涉及的参数较多。为简化调用，点播预定义了一系列标准雪碧图参数模板，每一个模板都对应一组截图参数。

使用该参数模板的场景包括：

1. 截取雪碧图接口（CreateImageSprite）；
2. 对视频文件进行处理接口（ProcessFile）中的雪碧图控制参数。

内置雪碧图参数模板

模板ID	截图输出格式	小图宽度	小图高度	小图行数	小图列数	截图间隔	填充方式
10	JPG	142	80	10	10	10	留黑

水印参数模板

水印参数模板介绍

对音视频文件进行打水印操作，涉及多个参数，开发者可以将这些参数组合成参数模板，在调用点播相关接口来处理视频时，可以仅指定参数模板，而无需传递一系列参数。

水印参数模板会在以下几种场景中用到：

1. 对视频文件进行处理接口（ProcessFile）中的转码控制参数。
2. 视频转码接口（ConvertVodFile
   ），当请求参数中指定isWatermark为1时（此时系统使用用户在控制台配置的默认模板）。

转码模板参数说明

- 模板ID（definition）：一个模板的唯一标志，由点播系统分配。
- 水印类型（type）：目前支持image。
- 水平相对位置（left）：位置参数，水印在视频水平位置的百分比，最小为0，最大为99，0在最左边。不填默认为0。
- 垂直相对位置（top）：位置参数，水印在视频垂直位置的百分比，最小为0，最大为99，0在最上面。不填默认为0。
- 相对宽度（widthPercentage）：大小参数，水印占视频宽的百分比，最小为1，最大为100。该值与left之和不能超过100，目前均默认为10。

服务端事件通知

服务端事件通知综述

事件通知简介

在某些事件发生，或者异步任务执行完毕之后，点播服务端可以将此类事件通知到APP的服务端。目前点播系统支持的事件通知包括：

事件类型	eventType
任务流状态变更通知	ProcedureStateChanged
视频上传完成	NewFileUpload
URL转拉完成	PullComplete
视频转码完成	TranscodeComplete
视频拼接完成	ConcatComplete
视频剪辑完成	ClipComplete
视频截取雪碧图完成	CreateImageSpriteComplete
视频按时间点截图完成	CreateSnapshotByTimeOffsetComplete

点播支持APP通过两种方式来获取事件通知：

- HTTP回调；
- 基于消息队列的可靠通知。

两种方式相比：前者配置简单；后者安全性高、可靠性强，但配置略为复杂。

HTTP回调

HTTP回调的基本流程是：APP后台搭建一个HTTP服务来接收回调，并在点播控制台上配置回调URL；当事件发生时，点播服务端会向该URL发起HTTP POST请求，具体事件内容将通过HTTP Body送达APP后台。

配置方法

在"控制台" -> "全局设置" -> "回调配置"页面中进行设置："回调URL"设置为APP后台接收回调的地址；"回调

模式"选择为"普通回调"；同时选择您需要开启的事件回调类型。

如下图所示：

回调协议

请求：HTTP POST请求，包体内容为JSON，每一种回调的具体包体内容参见各自文档。

应答：点播服务端忽略应答包内容。

公共参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	事件类型，APP后台可以根据该字段区分不同的回调
data	Object	具体回调数据
data.status	Integer	错误码, 0: 成功, 其他值: 失败
data.message	String	错误信息

回调请求示例

如下为视频拼接成功回调示例（完整HTTP请求），此处假定回调URL为

https://www.example.com/path/to/your/service

。

POST /path/to/your/service
HOST: www.example.com

{
"version": "4.0",
"eventType": "ConcatComplete",
"data": {
"vodTaskId": "concat-1edb7eb88a599d05abe451cfc541cfbd",
"fileInfo": [
{
"fileType": "m3u8",
"status": 0,
"message": "",
"fileId": "14508071098244931831",
"fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244931831/playlist.f6.m3u8"
},
{
"fileType": "mp4",
"status": 0,
"message": "",
"fileId": "14508071098244929440",
"fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/f0.mp4"
}
]
}
}

回调应答示例

对于HTTP回调，APP后台需要对回调请求进行应答，点播后台会忽略HTTP返回码与包体。示例如下：

```
HTTP/1.1 200 OK
Server: nginx/1.7.10
Date: Fri, 09 Oct 2015 02:59:55 GMT
Content-Length: 4

{
}
```

回调超时时间与重试

点播服务端发起HTTP回调的超时时间为5秒；如果回调超时，点播后台会重试两次（即：总共最多发送三次）。如果您对回调的可靠性十分敏感，建议使用基于消息队列的可靠通知。

安全考虑

为确保安全，APP可以将回调URL设置为HTTPS。此种方案的安全性可以达到：

- 点播服务端能够认证APP回调URL的合法性；
- 回调数据无法被人监听。

安全性无法达到：

- APP后台服务器无法认证请求方是否为点播服务端。

如果您需要更高的安全强度，建议使用基于消息队列的可靠通知。

基于消息队列的可靠通知

APP服务端可能会遭遇网络问题或者宕机等故障。简单HTTP回调，即时增加重试，也无法确保可靠性。为确保回调的可靠性，点播提供了第二种回调方式：基于消息队列的可靠通知。另外，此种方案的安全性更高。

配置方法

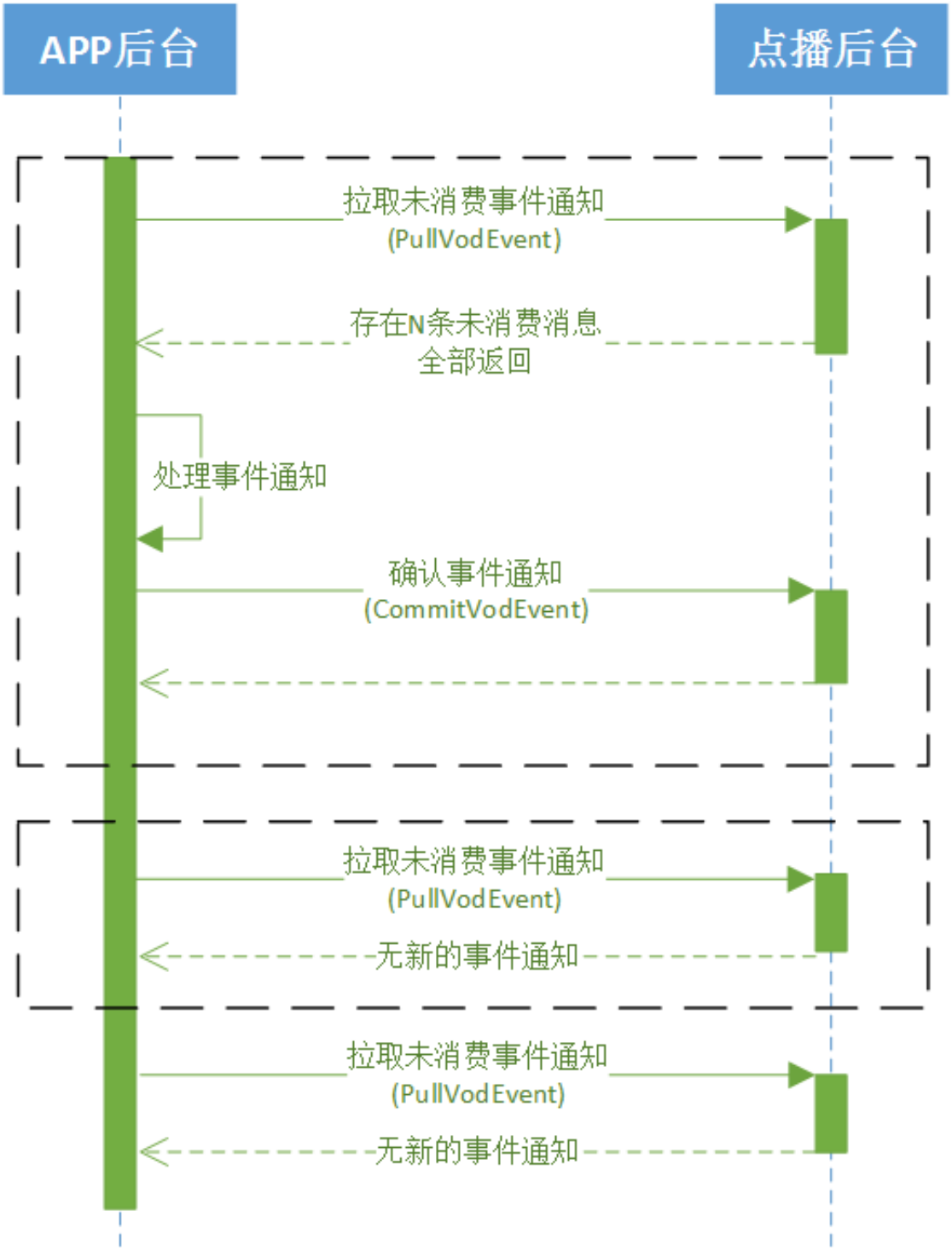
在"控制台" -> "全局设置" -> "回调配置"页面中进行设置："回调模式"选择为"可靠回调"；同时选择您需要开启的事件回调类型。注意：在可靠回调模式下，"回调URL"将不起作用。

如下图所示：

基本原理介绍

- 点播后台会为APP维护一个消息队列，并扮演生产者的角色；
- 当新的事件产生时，点播后台会将该消息放入消息队列中；
- APP后台扮演消费者的角色，需要通过长轮询的方式来获取消息队列中的内容。所谓长轮询，即：APP后台必须循环调用PullEvent
  接口来拉取事件通知。如果当前消息队列中存在消息，则该接口会立即返回；如果当前没有未消费事件通知，则点播后台会把该请求挂起，直到有新事件产生为止；每个请求最多挂起5秒。
- APP后台通过PullEvent拉取到消息之后，必须调用ConfirmEvent
  接口来确认该事件通知已经消费到，否则该消息可能会被再次消费。

如下图所示：第一个虚线框描述了一次事件通知的的完整消费流程；第二个虚线框表示PullEvent没有获取到新的事件通知，APP后台需要继续发起PullEvent调用。



相关接口

基于消息队列的可靠回调通过两个服务端API实现，参见：

- 拉取事件通知（PullEvent）；
- 确认事件通知（ConfirmEvent）。

示例

1，APP后台调用PullEvent，来获取未消费事件通知，请求URL为：

https://vod.api.qcloud.com/v2/index.php?Action=PullEvent
&COMMON_PARAMS

2，假设当前存在未消费事件通知，服务端应答包体如下。eventList中的msgHandle字段比较关键，后续会用到。

{
"code": 0,
"message": "",
"eventList": [// 拉取到的服务端事件通知列表，总共两个事件通知
{
"msgHandle": "MsgHandle1", // 第一个事件通知的句柄
"eventContent": { // 第一个事件通知的内容
"version": "4.0",
"eventType": "NewFileUpload", // 该字段表明事件通知的类型为“视频上传完成”
"data": {
"fileId": "14508071098244959037",
"fileUrl": "http://251000330.vod2.myqcloud.com/vod251000330/14508071098244959037/f0.flv",
"transcodeTaskId": "transcode-0bee89b07a248e27c83fc3d5951213c1",
"porncheckTaskId": "porncheck-f5ac8127b3b6b85cdc13f237c6005d80"
}
}
},
{
"msgHandle": "MsgHandle2", // 第二个事件通知的句柄
"eventContent": { // 第二个事件通知的内容
"version": 4,
"eventType": "ConcatComplete", // 该字段表明事件通知的类型为“视频拼接完成”
"data": {

"status": 0,
"message": "",
"vodTaskId": "concat-1edb7eb88a599d05abe451cfc541cfbd",
"fileInfo": [
{
"fileId": "14508071098244931831",
"fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244931831/playlist.f6.m3u8",
"fileType": "m3u8"
},
{
"fileId": "14508071098244929440",
"fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/f0.mp4",
"fileType": "mp4"
}
]
}
}
}
}
]
}

3，APP后台在消费到事件通知后，调用ConfirmEvent
接口来确认该事件通知已经收到。需要将PullEvent所返回eventList.msgHandle作为参数。

https://vod.api.qcloud.com/v2/index.php?Action=ConfirmEvent
&msgHandle.1=MsgHandle1
&msgHandle.2=MsgHandle2
&COMMON_PARAMS

4，点播服务端应答如下：

{

"code": 0,
"message": ""
}

至此，两条服务端事件通知接收完毕。APP后台需要再次调用PullEvent来轮询服务端事件通知。

事件通知-视频上传完成

事件名称

NewFileUpload

事件说明

如果APP配置了事件通知，则在视频上传任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为NewFileUpload
data.fileId	String	文件唯一id
data.author	String	作者信息
data.sourceType	String	文件的上传来源。目前有Record：录制，Pull：转拉，ClientUpload：客户端上传，ServerUpload：服务端上传
data.sourceContext	String	如果sourceType是Record，该字段即推流url；其它情况该字段暂未支持。该字段目前最多256字节。
data.streamId	String	推流id，录制上传特有
data.procedureTaskId	String	该视频上传之后进行了指定流程，则该参数为流程任务id
data.transcodeTaskId	String	如果该视频上传之后发起了转码，则该参数为转码任务id

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。


```
{
  "version": "4.0",
  "eventType": "NewFileUpload",
  "data": {
  "fileId": "145080710982449590xx",
  "fileUrl": "http://12530394xx.vod2.myqcloud.com/vod251000330/14508071098244959037/f0.flv",
  "transcodeTaskId": "transcode-0bee89b07a248e27c83fc3d5951213c1",
  "procedureTaskId": "12530394xx-mango-fa2fdf6a0f850d673be119cf51a7603a",
  "sourceType": "Record",
  "sourceContext": "rtmp://54xx.livepush.myqcloud.com/live?bizid=54xx&record=mp4&xx",
  "author": "",
  "streamId": "54xx_45"
  }
}
```

事件通知-URL 拉取视频上传完成

事件名称

PullComplete

事件说明

如果APP配置了事件通知，则在URL转拉任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为PullComplete
data	Object	具体回调数据
data.status	Integer	错误码，0: 成功；其他值: 失败
data.message	String	错误信息
data.fileId	String	发起拼接请求后获取到的唯一id
data.fileUrl	String	视频上传完成之后的URL
data.transcodeTaskId	String	如果该视频上传之后发起了转码，则该参数为转码任务id
data.porncheckTaskId	String	如果该视频上传之后发起了鉴黄，则该参数为鉴黄任务id

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

```
{
```

```
  "version": "4.0",
  "eventType": "PullComplete",
  "data": {
  "status": 0,
  "message": "",
  "vodTaskId": "Pull-f5ac8127b3b6b85cdc13f237c6005d8",
  "fileId": "14508071098244959037",
  "fileUrl": "http://251000330.vod2.myqcloud.com/vod251000330/14508071098244959037/f0.flv",
  "transcodeTaskId": "transcode-0bee89b07a248e27c83fc3d5951213c1",
  "porncheckTaskId": "porncheck-f5ac8127b3b6b85cdc13f237c6005d80"
  }
}
```

事件通知-转码完成

事件名称

TranscodeComplete

事件说明

如果APP配置了事件通知，则在视频转码任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

注意：

- 整个转码过程可能会输出多个规格的视频，例如标清、高清等；只有当所有规格的视频均生成成功之后，才会触发该回调。

参数说明

参数名称	类型	说明
version	String	事件通知版本号，固定为4.0
eventType	String	事件类型，固定为TranscodeComplete
data	Object	具体回调数据
data.status	Integer	错误码, 0: 成功, 其他值: 失败
data.message	String	错误信息
data.fileId	String	被转码的文件ID
data.vodTaskId	String	转码任务ID

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：转码成功


```
{
  "version": "4.0",
  "eventType": "TranscodeComplete",
  "data": {
  "status": 0,
  "message": "",
  "vodTaskId": "Transcode-1edb7eb88a599d05abe451cfc541cfbd",
  "fileId": "14508071098244931831",
  "fileName": "13425173277_2015-09-06-19-06-11_2015-09-06-19-16-11",
  "duration": 599,
  "coverUrl": "http://p.qpic.cn/videoyun/0/1203_8a5015084d4f47cd9a0bc5ecfe78aecb_1/640",
  "playSet": [
  {
  "url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f0.mp4",
  "definition": 0,
  "vbitrate": 246000,
  "vheight": 480,
  "vwidth": 640
  },
  {
  "url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f10.mp4",
  "definition": 10,
  "vbitrate": 149193,
  "vheight": 240,
  "vwidth": 320
  },
  {
  "url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f20.mp4",
```

"definition": 20,
"vbitrate": 297656,
"vheight": 480,
"vwidth": 640
},
{
"url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f220.av.m3u8",
"definition": 220,
"vbitrate": 524288,
"vheight": 480,
"vwidth": 640
},
{
"url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f30.mp4",
"definition": 30,
"vbitrate": 899976,
"vheight": 960,
"vwidth": 1280
},
{
"url": "http://vcloud1203.tc.qq.com/1203_8a5015084d4f47cd9a0bc5ecfe78aecb.f40.mp4",
"definition": 40,
"vbitrate": 1746652,
"vheight": 1440,
"vwidth": 1920
}
]
}
}

示例：转码失败

{

"version": "4.0",
"eventType": "TranscodeComplete",
"data": {
"status": -43001,
"message": "",
"fileId": "14508071098244931831",
"vodTaskId": "Transcode-1edb7eb88a599d05abe451cfc541cfbd"
}
}

错误码说明

事件通知包体中的status字段表示本次视频处理任务的执行结果。其含义参见视频处理类操作错误码说明。

事件通知-指定时间点截图完成

事件名称

CreateSnapshotByTimeOffsetComplete

事件说明

如果APP配置了事件通知，则在视频指定时间点截图任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为CreateSnapshotByTimeOffsetComplete
data	Object	具体回调数据
data.vodTaskId	String	指定时间点截图任务ID
data.fileId	String	指定时间点截图的FileId
data.definition	Integer	截图规格，参见指定时间点截图规格说明
data.picInfo	Array	指定时间点截图输出的截图信息

data.picInfo数组中每个元素均为Object，每个参数含义如下：

参数名称	类型	说明
timeOffset	Integer	截图的具体时间点，单位：毫秒
url	String	截图文件url
status	Integer	错误码, 0: 成功, 其他值: 失败

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：视频按时间点截图成功


```
{
  "version": "4.0",
  "eventType": "CreateSnapshotByTimeOffsetComplete",
  "data": {
  "vodTaskId": "CreateSnapshotByTimeOffset-1edb7eb88a599d05abe451cfc541cfbd",
  "fileId": "14508071098244929440",
  "definition": 10,
  "picInfo": [
  {
  "status": 0,
  "timeOffset": 3123213,
  "url": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/xx1.png"
  },
  {
  "status": 0,
  "timeOffset": 3123123,
  "url": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/xx2.png"
  }
  ]
  }
}
```


示例：视频按时间点截图失败


```
{
  "version": "4.0",
  "eventType": "CreateSnapshotByTimeOffsetComplete",
  "data": {
```

"vodTaskId": "CreateSnapshotByTimeOffset-1edb7eb88a599d05abe451cfc541cfbd",
"fileId": "9031868222841494656",
"definition": 20,
"picInfo": [
{
"status": -46204,
"timeOffset": 1000000
}
]
}
}

错误码说明

事件通知包体中的status字段表示本次视频处理任务的执行结果。其含义参见视频处理类操作错误码说明。

事件通知-截取雪碧图完成

事件名称

CreateImageSpriteComplete

事件说明

如果APP配置了事件通知，则在视频截取雪碧图任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为CreateImageSpriteComplete
data	Object	具体回调数据
data.status	Integer	错误码, 0: 成功, 其他值: 失败
data.message	String	错误信息
data.vodTaskId	String	截取雪碧图任务ID
data.fileId	String	截取雪碧图的FileId
data.definition	Integer	雪碧图规格，参见雪碧图截图规格说明
data.totalCount	Integer	雪碧图小图总数量
data.imageSpriteUrl	Array	截图雪碧图输出的雪碧图信息

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：视频截取雪碧图成功


```
{
  "version": "4.0",
  "eventType": "CreateImageSpriteComplete",
  "data": {
  "status": 0,
  "message": "",
  "vodTaskId": "CreateImageSprite-1edb7eb88a599d05abe451cfc541cfbd",
  "fileId": "14508071098244929440",
  "definition" : 10,
  "totalCount": 106,
  "imageSpriteUrl": [
  "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/xx1.png",
  "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/xx2.png"
  ]
  }
}
```


示例：视频截取雪碧图失败


```
{
  "version": "4.0",
  "eventType": "CreateImageSpriteComplete",
  "data": {
  "status": -43001,
  "message": "",
  "vodTaskId": "CreateImageSprite-1edb7eb88a599d05abe451cfc541cfbd",
  "fileId": "14508071098244929440"
  }
}
```

错误码说明

事件通知包体中的status字段表示本次视频处理任务的执行结果。其含义参见视频处理类操作错误码说明。

事件通知-剪辑完成

事件名称

ClipComplete

事件说明

如果APP配置了事件通知，则在视频剪辑任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为ClipComplete
data	Object	具体回调数据
data.vodTaskId	String	剪辑任务ID
data.srcFileId	String	剪辑任务源文件fileId
data.fileInfo	Object	剪辑输出的视频文件信息

data.fileInfo每个参数含义如下：

参数名称	类型	说明
fileType	String	剪辑的文件类型
status	Integer	该类型文件的错误信息，0为成功，其他为失败，详见错误码说明
message	String	错误信息
fileId	String	剪辑的文件的fileId
fileUrl	String	剪辑的文件url

示例

- 对于HTTP回调，以下内容为HTTP Post Body；
- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：视频剪辑成功


```
{
  "version": "4.0",
  "eventType": "ClipComplete",
  "data": {
  "vodTaskId": "clipVideo-0a78cf44c4285026a4c",
  "srcFileId": "16092504232103571364",
  "fileInfo": {
  "fileType": "mp4",
  "status": 0,
  "message": "",
  "fileId": "14508071098244929440",
  "fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/f0.mp4"
  }
  }
}
```


示例：视频剪辑失败


```
{
  "version": "4.0",
  "eventType": "ClipComplete",
  "data": {
  "vodTaskId": "clipVideo-0a78cf44c4285026a4c",
  "srcFileId": "16092504232103571364",
  "fileInfo": {
  "status": -43001,
  "message": ""
  }
```

```
  }
}
```

错误码说明

事件通知包体中的status字段表示本次视频处理任务的执行结果。其含义参见视频处理类操作错误码说明。

事件通知-拼接完成

事件名称

ConcatComplete

事件说明

如果APP配置了事件通知，则在视频拼接任务完成之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

参数说明

参数名称	类型	说明
version	String	回调版本号，固定为4.0
eventType	String	回调类型，固定为ConcatComplete
data	Object	具体回调数据
data.vodTaskId	String	拼接任务ID
data.fileInfo	Array	拼接输出的视频文件信息

data.fileInfo数组中每个元素均为Object，每个参数含义如下：

参数名称	类型	说明
fileType	String	拼接出的文件类型
status	Integer	任务执行结果，0为成功，-1或者4为失败
message	String	错误信息
fileId	String	拼接处的文件的fileId
fileUrl	String	拼接出的文件url

示例

- 对于HTTP回调，以下内容为HTTP Post Body；

- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：视频拼接成功


```
{
  "version": "4.0",
  "eventType": "ConcatComplete",
  "data": {
  "vodTaskId": "Concat-1edb7eb88a599d05abe451cfc541cfbd",
  "fileInfo": [
  {
  "fileType": "m3u8",
  "status": 0,
  "message": "",
  "fileId": "14508071098244931831",
  "fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244931831/playlist.f6.m3u8"
  },
  {
  "fileType": "mp4",
  "status": 0,
  "message": "",
  "fileId": "14508071098244929440",
  "fileUrl": "http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/f0.mp4"
  }
  ]
  }
}
```


示例：视频拼接失败


```
{
  "version": "4.0",
  "eventType": "ConcatComplete",
```

"data": {
"message": "",
"vodTaskId": "Concat-1edb7eb88a599d05abe451cfc541cfbd",
"fileInfo": [
{
"fileType": "m3u8",
"status": 4,
"message": ""
},
{
"fileType": "mp4",
"status": 4,
"message": ""
}
]
}
}

错误码说明

事件通知包体中的status字段表示本次视频处理任务的执行结果，0为成功，-1或者4为失败。

事件通知-任务流状态变更

事件名称

ProcedureStateChanged

事件说明

如果APP配置了事件通知，则在任务流状态变更之后，点播后台会将该事件通知给APP后台。

APP后台接收该事件通知的方法参见服务端事件通知简介。

注意：

- 任务流状态变更通知模式由发起任务流时的参数notifyMode决定。 当notifyMode为Finish：只有当任务流全部执行完毕时，才发起一次事件通知；当notifyMode为Change：只要任务流中每个子任务的状态发生变化，都进行事件通知。 默认为Finish。

参数说明

参数名称	类型	说明
version	String	事件通知版本号，固定为4.0
eventType	String	事件类型，固定为ProcedureStateChanged
data	Object	具体回调数据
data.status	String	任务流状态，有PROCESSING和FINISH两种
data.errCode	Integer	错误码。 0: 成功, 其他值: 失败
data.message	String	错误信息
data.fileId	String	文件ID
data.metaData	Object	视频元信息，该字段一定存在，字段信息参见metaData（视频元信息）

参数名称	类型	说明
data.drm	Object	文件加密信息，用户在发起任务流时在转码控制参数 指定了加密，该字段才存在。 字段信息参见drm（视频加密信息）
data.processTaskList	Array	任务流包含的任务列表，字段信息参见processTaskList（任务列表）

metaData（视频元信息）

参数名称	类型	描述
size	Integer	视频大小。单位：字节
container	String	容器类型，例如m4a，mp4等
bitrate	Integer	视频流码率平均值与音频流码率平均值之和。 单位：kbps
height	Integer	视频流高度的最大值。单位：px
width	Integer	视频流宽度的最大值。单位：px
md5	String	视频的md5值
duration	Integer	视频时长。单位：秒
videoStreamList	Array	视频流信息
videoStreamList.bitrate	Integer	视频流的码率，单位：kbps
videoStreamList.height	Integer	视频流的高度，单位：px
videoStreamList.width	Integer	视频流的宽度，单位：px
videoStreamList.codec	String	视频流的编码格式，例如h264
videoStreamList.fps	Integer	帧率，单位：hz
audioStreamList	Array	音频流信息
audioStreamList.bitrate	Integer	音频流的码率。 单位：kbps
audioStreamList.samplingRate	Integer	音频流的采样率。 单位：hz
audioStreamList.codec	String	音频流的编码格式，例如aac

drm（视频加密信息）

参数名称	类型	描述
definition	Integer	加密模板ID
keySource	String	KMS的类型，总共三种，分别为 VodBuildInKMS：腾讯云点播内置KM

参数名称	类型	描述
		S；QCloudKMS：腾讯云KMS系统（暂不支持）；PrivateKMS：用于自有KMS系统（暂不支持）。
getKeyUrl	String	获取解密秘钥的URL
edkList	Array	加密后的数据秘钥列表

processTaskList（任务列表）

任务信息列表，目前有Trancode（转码任务），SampleSnapshot（采样截图任务），CoverBySnapshot（截图图片作为视频封面任务），SnapshotByTimeOffset（按时间点截图任务），PullFile (拉取视频文件)，ImageSprites（雪碧图截图任务）。

Trancode（转码任务）

参数名称	类型	描述
taskType	String	任务类型，固定为Transcode
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败,其中30009为原文件异常失败；30010为系统失败或未知
message	String	错误信息
input	Object	任务的输入信息
input.definition	Integer	转码模板ID
input.watermark	Integer	是否设置水印，1为设置，0为没设置。 是否设置取决于用户转码配置
output	Object	任务的输出信息，任务成功时会有该字段，失败则没有
output.url	String	视频url
output.size	Integer	视频大小。单位：字节
output.container	String	容器类型，例如m4a，mp4等
output.bitrate	Integer	视频流码率和音频流码率之和，单位：kbps
output.height	Integer	视频流高度的最大值。单位：px

参数名称	类型	描述
output.width	Integer	视频流宽度的最大值。单位：px
output.md5	String	视频的md5值
output.duration	Integer	视频时长。单位：秒
output.videoStreamList	Array	视频流信息，元素字段和元信息中videoStreamList相同
output.audioStreamList	Array	音频流信息，元素字段和元信息中audioStreamList相同

SampleSnapshot（采样截图任务）

参数名称	类型	描述
taskType	String	任务类型，固定为SampleSnapshot
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败，其中30009为原文件异常失败；30010为系统失败或未知
message	String	错误信息
input	Object	任务的输入信息
input.definition	Integer	采样截图模板ID
output	Object	任务的输出信息，任务成功时会有该字段，失败则没有
output.imageUrls	Array	字符串数组，生成的截图url列表

SnapshotByTimeOffset（按时间点截图任务）

参数名称	类型	描述
taskType	String	任务类型，固定为SnapshotByTimeOffset
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败，其中30009为原文件异常失败；30010为系统失败或未知
message	String	错误信息

参数名称	类型	描述
input	Object	任务的输入信息
input.definition	Integer	按时间点截图模板ID
input.timeOffset	Array	整形数组，截图的时间偏移，单位为毫秒
output	Object	任务的输出信息，任务成功时会有该字段，失败则没有
output.imgInfo	Array	生成的截图信息列表
output.imgInfo.timeOffset	Integer	该张截图的时间偏移，单位毫秒
output.imgInfo.url	String	截图url

CoverBySnapshot（截图图片作为视频封面任务）

参数名称	类型	描述
taskType	String	任务类型，固定为CoverBySnapshot
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败，其中30009为原文件异常失败；30010为系统失败或未知
message	String	错误信息
input	Object	任务的输入信息
input.definition	Integer	采样截图模板ID
input.positionType	String	截图方式。Time：依照时间点截图；Percent：依照百分比截图
input.position	Integer	截图位置。对于依照时间点截图，该值表示指定视频第几秒的截图作为封面；对于依照百分比截图，该值表示使用视频百分之多少的截图作为封面
output	Object	任务的输出信息，任务成功时会有该字段，失败则没有
output.imageUrl	Array	作为视频封面的截图url

PullFile（拉取视频文件任务）

参数名称	类型	描述
taskType	String	任务类型，固定为PullFile
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败
message	String	错误信息
input	Object	任务的输入信息
input.url	String	需要拉取的视频的 URL
input.fileName	String	视频文件的名称
input.md5	Integer	视频文件的MD5
output	Object	任务的输出信息，任务成功时会有该字段，失败则没有
output.fileId	String	视频文件ID
output.fileSize	String	视频文件大小
output.url	String	视频文件的播放地址

ImageSprites（雪碧图截图任务）

参数名称	类型	描述
taskType	String	任务类型，固定为ImageSprites
status	String	任务状态，有PROCESSING，SUCCESS，FAIL三种
errCode	Integer	错误码。 0: 成功, 其他值: 失败
message	String	错误信息
input	Object	任务的输入信息，任务成功时会有该字段，失败则没有
input.definition	Integer	雪碧图模板ID
output	Object	任务的输出信息
output.totalCount	Integer	雪碧图小图总数
output.urlList	Array	字符串数组，生成的雪碧图的url列表

示例

- 对于HTTP回调，以下内容为HTTP Post Body；

- 对于基于消息队列的可靠通知，以下内容为PullEvent接口返回包体中eventList.eventContent的内容。

示例：


```
{
  "version": "4.0",
  "eventType": "ProcedureStateChanged",
  "data": {
  "vodTaskId": "251000333-mango-c27bdf65ea06646171e714f25f5aac63",
  "status": "PROCESSING",
  "message": "",
  "errCode": 0,
  "fileId": "123123123",
  "metaData": {
  "size": 10556,
  "container": "m4a",
  "bitrate": 246035,
  "height": 480,
  "width": 640,
  "md5": "b3ae6ed07d9bf4efeeb94ed2d37ff3e3",
  "duration": 3601,
  "videoStreamList": [
  {
  "bitrate": 246000,
  "height": 480,
  "width": 640,
  "codec": "h264",
  "fps": 22
  }
  ],
  "audioStreamList": [
  {
  "codec": "aac",
```

"samplingRate": 44100,
"bitrate": 35
}
]
},
"drm": {
"definition": 10,
"getKeyUrl": "https://123.xxx.com/getkey",
"keySource": "VodBuildInKMS",
"edkList": [
"232abc30"
]
},
"processTaskList": [
{
"taskType": "Transcode",
"status": "PROCESSING",
"errCode": 0,
"message": "",
"input": {
"definition": 10,
"watermark": 1
}
},
{
"taskType": "Transcode",
"status": "SUCCESS",
"errCode": 0,
"message": "",
"input": {
"definition": 20,
"watermark": 1
},
"output": {

"url":
"http://125xx.vod2.myqcloud.com/9fcd41e6vodgzp1251953760/7585975f9031868222912340/f20.mp4",
"size": 10556,
"container": "m4a",
"md5": "b3ae6ed07d9bf4efeeb94ed2d37ff3e3",
"bitrate": 246035,
"height": 480,
"width": 640,
"duration": 3601,
"videoStreamList": [
{
"bitrate": 246000,
"height": 480,
"width": 640,
"codec": "h264",
"fps": 222
}
],
"audioStreamList": [
{
"codec": "aac",
"samplingRate": 44100,
"bitrate": 35
}
]
}
},
{
"taskType": "SampleSnapshot",
"status": "SUCCESS",
"errCode": 0,
"message": "",
"input": {

"definition": 10
},
"output": {
"imageUrls": [
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx1.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx2.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx3.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx4.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx5.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx6.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx7.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx8.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx9.png",
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx10.png"
]
}
},
{
"taskType": "CoverBySnapshot",
"status": "SUCCESS",
"errCode": 0,
"message": "",
"input": {
"definition": 10,
"position": 10,
"positionType": "Percent"
},
"output": {
"imageUrl":
"http://125xx.vod2.myqcloud.com/vodtrans125xx/14508071098244929440/shotup/xx1.png"
}
},
{
"taskType": "SnapshotByTimeOffset",

"status": "SUCCESS",
"message": "",
"errCode": 0,
"input": {
"definition": 10,
"timeOffset": [
300,
400
]
},
"output": {
"imgInfo": [
{
"timeOffset": 300,
"url":
"http://125xx.vod2.myqcloud.com/42f909a8vodtransgzp251000333/dee2963524820810452267193/snapshot/1502280085_887773835.100_300.jpg"
},
{
"timeOffset": 400,
"url":
"http://125xx.vod2.myqcloud.com/42f909a8vodtransgzp251000333/dee2963524820810452267193/snapshot/1502280085_887773835.100_400.jpg"
}
]
}
},
{
"taskType": "ImageSprites",
"status": "SUCCESS",
"message": "SUCCESS",
"errCode": 0,
"input": {
"definition": 10

},
"output": {
"totalCount": 106,
"urlList": [

"http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/1502280085_887773835.100_0.png",

"http://125xx.vod2.myqcloud.com/vod125xx/14508071098244929440/1502280085_887773835.100_1.png"
]
}
}
]
}
}

错误码说明

事件通知包体中的errCode字段表示本次视频处理任务的执行结果。其含义参见视频处理类操作错误码说明。