云点播

开发指南

产品文档

腾讯云

文档目录

开发指南
媒体上传
媒体上传综述

最近更新时间：2024-10-22 16:38:02

媒体上传是指用户将视频、音频、封面图片等媒体文件上传到云点播的存储中，以进行后续的处理和分发等。

上传方式

云点播支持以下几种上传方式：

控制台本地上传

在云点播控制台上传页面进行操作，将本地媒体文件上传到云点播。适用于开发者直接管理少量媒体的场景，具有方便快捷、无技术门槛的优点。

控制台拉取上传

在云点播控制台上传页面进行操作，指定待上传媒体的 URL，由云点播后台进行离线拉取。

服务端上传

开发者将存储在其后台服务器中的媒体文件上传到云点播，适用于自动化、系统化的生产环境。云点播提供了以下编程语言的服务端上传 SDK：

Java SDK

C# SDK

PHP SDK

Python SDK

Node.js SDK

Golang SDK

客户端上传

终端用户将客户端本地视频上传到云点播，适用于 UGC、PGC 等场景。云点播提供了以下平台的客户端上传 SDK：

Android 上传 SDK

iOS 上传 SDK

Web 端上传 SDK

API 拉取上传

使用云点播提供的服务端 API 拉取上传接口，指定待上传媒体的 URL，由云点播后台进行离线拉取。适用于大量或自动化的媒体文件迁移场景。

直播录制

通过云直播提供的录制功能，将直播流的视频内容存储到云点播，以进行存档、剪辑和回看等。

存储地域

已支持地域列表

云点播在全球多个地域有存储节点，媒体上传过程中会选择其中一个地域进行存储。云点播目前支持的存储地域如下：

存储地域	地域英文简称
北京	ap-beijing
上海	ap-shanghai
广州	ap-guangzhou
重庆	ap-chongqing
天津	ap-beijing-1
南京	ap-nanjing
成都	ap-chengdu
中国香港	ap-hongkong
中国台北	ap-taipei
新加坡	ap-singapore
印度尼西亚雅加达	ap-jakarta
韩国首尔	ap-seoul
泰国曼谷	ap-bangkok
日本东京	ap-tokyo
美国硅谷（美西）	na-siliconvalley
美国弗吉尼亚（美东）	na-ashburn
巴西圣保罗	sa-saopaulo
德国法兰克福	eu-frankfurt

开通存储地域

多存储地域的一个重要作用是提升媒体上传质量（成功率和速度）。上传者与存储节点的距离对上传质量有影响，一般来说近距离的上传质量要优于远距离。

开发者开通云点播服务后，云点播会自动分配**新加坡**存储地域。开发者可以根据业务需要开通其它的存储地域，具体操作请参见 上传存储设置。**存储地域一旦开通将无法关闭**。

默认存储地域

开发者已有的存储地域中，有且仅有一个将作为默认存储地域。如果开发者仅有1个存储地域（即新加坡），那么它将是默认存储地域；如果开发者开通了多个存储地域，那么可以在控制台选择其它地域作为默认存储地域。具体操作请参见 存储地域设置。

默认存储地域的作用：在某些场景下，优先选择该地域作为媒体上传的目标地域。具体说明请阅读下文。

选择存储地域

媒体上传时需要选择一个存储地域，默认由云点播后台自动选择，也可以由开发者在上传请求中指定。

当云点播后台自动选择存储地域时：

如果开发者仅有1个存储地域（即新加坡），那么所有上传的媒体都会存储在该地域。

如果开发者开通了多个存储地域，那么各种上传方式的选择策略如下：

上传方式	地域选择策略
控制台本地上传	根据上传者地理位置，就近选择存储地域。
控制台拉取上传	固定选择默认存储园区
服务端上传	根据上传者地理位置，就近选择存储地域。
客户端上传	根据上传者地理位置，就近选择存储地域。
API 拉取上传	固定选择默认存储园区
直播录制	根据直播推流所在地域，就近选择存储地域。

当开发者指定存储地域时，各种上传方式的指定方法如下：

上传方式	地域指定方法
控制台本地上传	不支持
控制台拉取上传	不支持
服务端上传	Java SDK C# SDK PHP SDK Python SDK Node.js SDK

	Go SDK
客户端上传	客户端上传签名参数
API 拉取上传	拉取上传接口 StorageRegion 参数
直播录制	不支持

功能与限制

媒体类型

云点播支持上传以下类型的媒体文件：
视频：WMV、RM、MOV、MPEG、MP4、3GP、FLV、AVI、RMVB、TS、ASF、MPG、WEBM、MKV、M3U8、WM、ASX、RAM、MPE、VOB、DAT、MP4V、M4V、F4V、MXF、QT、OGG。
音频：MP3、M4A、FLAC、OGG、WAV、RA、AAC、AMR。
封面图片：JPG、JPEG、PNG、GIF、BMP、TIFF、AI、CDR、EPS、TIF。

事件通知

媒体上传完成后，云点播后台可以将这一事件通知给开发者。事件通知原理请参见 事件通知，配置方法请参见 事件通知配置。
各种上传方式对应的事件通知类型如下：

上传方式	事件通知类型
控制台本地上传 服务端上传 客户端上传 直播录制	视频上传完成
控制台拉取上传 API 拉取上传	URL 拉取视频上传完成

附属功能

云点播媒体上传支持多种附属功能，包括媒资管理相关、视频处理和事件通知相关、上传控制相关。

媒资管理相关

附带封面：在上传视频的同时附带上传一张图片，该图片将在云点播媒资系统中自动设为该视频的封面。
指定过期时间：在上传时指定该媒体文件的过期时间，当到达指定时间后，云点播后台自动删除该媒体文件及其附属文件（如转码文件、截图等）。
指定分类：在上传后将该媒体文件设置为某个分类。

各种上传方式的支持情况和用法如下表：

功能	控制台本地上传	控制台拉取上传	服务端上传	客户端上传	API 拉取上传	直播录制
附带封面	不支持	不支持	Java SDK C# SDK PHP SDK Python SDK Node.js SDK Go SDK	Web SDK Android SDK iOS SDK	拉取上传接口 CoverUrl 参数	不支持
指定过期时间	不支持	不支持	Java SDK 接口 ExpireTime 参数 C# SDK 接口 ExpireTime 参数 PHP SDK 接口 ExpireTime 参数 Python SDK 接口 ExpireTime 参数 Node.js SDK 接口 ExpireTime 参数 Go SDK 接口 ExpireTime 参数	不支持	拉取上传接口 ExpireTime 参数	录制配置
指定分类	指定分类	不支持	Java SDK 接口 ClassId 参数 C# SDK 接口 ClassId 参数 PHP SDK 接口 ClassId 参数 Python SDK 接口 ClassId 参数 Node.js SDK 接口 ClassId 参数 Go SDK 接口 ClassId 参数	客户端上传签名 classId 参数	拉取上传接口 ClassId 参数	不支持

视频处理和事件通知相关

自动视频处理：在上传媒体的同时指定一个 任务流，上传完成后，云点播自动执行该任务流。常见的场景有：截取视频首帧图像作为封面、转码和内容审核等。

视频处理事件通知透传字段：如果启用了自动视频处理，在处理完成后，云点播后台发起事件通知时透传该字段给开发者。

上传事件通知透传字段：在上传完成后，云点播后台发起事件通知时透传该字段给开发者。

各种上传方式的支持情况和用法如下表：

功能	控制台本地上传	控制台拉取上传	服务端上传	客户端上传	API 拉取上传	直播录制
自动视频处理	上传后自动进行视频处理	不支持	Java SDK C# SDK PHP SDK Python SDK Node.js SDK Go SDK	客户端上传签名 procedure 参数	拉取上传接口 Procedure 参数	不支持
视频处理事件通知透传字段	不支持	不支持	不支持	客户端上传签名 sessionContext 参数	拉取上传接口 SessionContext 参数	不支持
上传事件通知透传字段	不支持	不支持	Java SDK 接口 SourceContext 参数 C# SDK 接口 SourceContext 参数 PHP SDK 接口 SourceContext 参数 Python SDK 接口 SourceContext 参数 Node.js SDK 接口 SourceContext 参数 Go SDK 接口 SourceContext 参数	客户端上传签名 sourceContext 参数	不支持	不支持

上传控制相关

断点续传：上传过程如果意外终止（如网络中断、关闭浏览器等），当用户再次尝试上传同一文件时，可以从中断处继续上传，不需要重新上传整个文件。

暂停/恢复上传：上传过程中，用户可以主动暂停，并可主动恢复上传。

取消上传：上传过程中，用户可以主动终止本次上传。

获取上传进度：媒体已上传到云点播部分的大小占比。

分片上传：在上传时将媒体文件切分为多个小分片，分别上传。一方面可以在弱网环境下将网络异常导致的中断影响降低；另一方面可以在高带宽环境中并发上传多个分片，充分利用网络带宽。

各种上传方式的支持情况和用法如下表：

功能	控制台本地上传	控制台拉取上	服务端上传	客户端上传	API 拉取上传	直播录制

		传				
断点续传	不支持	不涉及	不支持	Web SDK Android SDK iOS SDK	不涉及	不涉及
暂停和恢复上传	不支持	不涉及	不支持	Web SDK Android SDK iOS SDK	不涉及	不涉及
取消上传	刷新或关闭浏览器页面	不涉及	不支持	Web SDK Android SDK iOS SDK	不涉及	终止录制任务
获取上传进度	页面默认显示进度	不支持	不支持	Web SDK Android SDK iOS SDK	不支持	不涉及
分片上传	已启用	不涉及	Java SDK C# SDK PHP SDK Python SDK Node.js SDK Go SDK	Web SDK 默认启用 Android SDK 默认启用 iOS SDK 默认启用	不涉及	不涉及

限制

媒体文件大小的限制如下：

上传方式	媒体大小限制
控制台本地上传 客户端上传 - Web SDK	60GB
服务端上传 控制台拉取上传 API 拉取上传	48.82TB（50,000GB）
客户端上传 - Android SDK 客户端上传 - iOS SDK	10GB
直播录制	MP4/FLV 格式为48.82TB（50,000GB） HLS 格式总大小无限制 其它限制取决于 直播录制

文件数量：无限制。

服务端上传
服务端上传指引

最近更新时间：2022-05-26 12:34:50

操作场景

服务端视频上传是指 App 后台将视频上传到云点播平台，本文将为您介绍如何使用服务端 API 上传视频。

前提条件

1. 开通服务

开通云点播服务，详细请参见 购买指引。

2. 获取云 API 密钥

获取调用服务端 API 所需的安全凭证，即 SecretId 和 SecretKey，具体步骤如下：

1. 登录控制台，选择【云产品】>【访问管理】>【API密钥管理】，进入“API 密钥管理”页面。
2. 获取云 API 密钥。如果您尚未创建密钥，则单击【新建密钥】即可创建一对 SecretId 和 SecretKey。

操作步骤

1. 发起上传

发起上传的方式分别为：通过 SDK 发起上传、通过 API 发起上传。

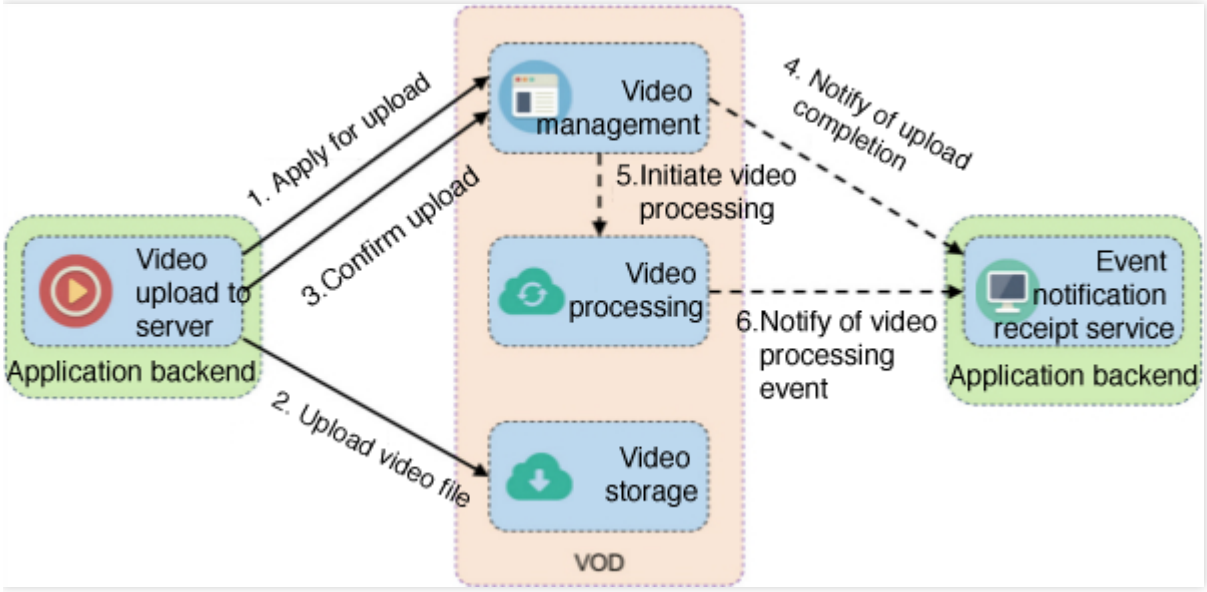
通过 SDK 发起上传

为了方便用户开发端上传功能，云点播提供基于多语言平台 SDK，每种语言的 SDK 均包含相应 Demo，详情请参见：

- PHP SDK
- Java SDK
- Python SDK
- Node.js SDK
- Go SDK
- C# SDK

通过 API 发起上传

如果云点播提供的上传 SDK 没有涵盖 App 后台所使用的语言，则 App 后台需要自行调用云点播的服务端 API 进行视频上传（这种方式较为复杂，不推荐），基于 API 上传的业务流程图如下：



相比 SDK 方式的上传，基于 API 方式的上传需要自行实现申请上传和上传文件等步骤。而上传文件也没有 SDK 的方式方便，对于大文件需要自己做分片上传等逻辑。详情请参见：

- 服务端 API - 申请上传
- 服务端 API - 上传文件
- 服务端 API - 确认上传

高级功能

- **上传时指定任务流**
  如果开发者需要在视频上传完成后自动发起 视频处理任务流（例如转码、截图等），可以在调用服务端 API 申请上传 时通过 `Procedure` 参数来实现。参数值为任务流模板名，云点播支持 创建任务流模板 并为模板命名。发起任务流时，可以用任务流模板名来表示要发起的任务，云点播提供的多语言 SDK 都支持指定任务流参数，详情请参见：
  - PHP SDK
  - Java SDK
  - Python SDK
  - Node.js SDKSDK
  - Go SDK
  - C# SDK
- **上传时指定存储地域**
  云点播默认提供的存储地域设置在新加坡，如果需要存储到其他区域，可以在控制台上自助添加其他存储地域，

详情请参见 上传存储设置。设置完成后调用服务端 API 申请上传 时通过 `StorageRegion` 参数来实现，参数值为存储地域的 英文简称。云点播提供的多语言 SDK 都支持上传时指定存储地域，详情请参见：

- PHP SDK
- Java SDK
- Python SDK
- Node.js SDKSDK
- Go SDK
- C# SDK

2. 事件通知

视频上传完成之后，云点播会给 App 后台发起 事件通知 - 视频上传完成，App 后台可通过该事件感知到视频上传行为。如果要接收事件通知，则 App 需要到 控制台 - 回调设置 开启事件通知。事件通知 - 视频上传完成 主要包含如下信息：

- 新视频文件的 FileId 和 URL。
- 云点播支持在视频上传时指定透传字段，事件完成将透传字段通知到 App 后台。在事件通知中有如下字段：
  - `SourceType` ：该字段被腾讯云固定成 `ServerUpload` ，表示上传来源为服务端上传。
  - `SourceContext` ：用户自定义透传字段，App 后台在派发签名时指定的透传内容，对应签名中的 `sourceContext` 参数。
- 云点播支持在视频上传完成时自动发起视频处理，如果上传时指定了 视频处理任务流，则在事件通知内容中也会携带任务 ID，即事件通知中的 `data.procedureTaskId` 字段。

更多信息请参见：

- 任务管理与事件通知
- 事件通知 - 视频上传完成

上传文件

最近更新时间：2020-03-12 16:46:02

接口说明

- 上传小文件（小于5MB）使用的 API，请参见 简单上传文件。
- 上传大文件（大于5MB）使用的 API，请参见 初始化分片上传、逐个上传分片 和 结束上传分片。

功能说明

1. 上传媒体（和封面）文件。
2. API 在服务端的上传步骤请参见 服务端上传综述。

SDK 方式

建议使用 封装的 SDK 进行 API 的调用。

API 方式

使用方式请参见上述 API 链接中的文档，各 API 的语法为：

```
PUT <ObjectName> HTTP/1.1
Host: <BucketName>-<APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

语法中的以下变量取值 VOD 申请上传的返回结果：

- `<ObjectName>` 为 **MediaStoragePath**（对于封面文件为 **CoverStoragePath**）。
- `<BucketName>-<APPID>` 为 **StorageBucket**。
- `<Region>` 为 **StorageRegion**。

对于 API 请求，需要注意的点如下：

- Authorization 签名使用 VOD 申请上传返回结果 中 **TempCertificate** 的 SecretId 和 SecretKey，请参见 签名文档 进行计算。

- 在 HTTP 头部或 POST 请求包的 form-data 中传入 **x-cos-security-token** 字段（标识该请求使用的安全令牌），并赋值为 **TempCertificate** 中的 Token 字段。

Java SDK

最近更新时间：2022-06-24 15:51:58

对于在服务端上传视频的场景，云点播提供 Java SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

Maven 依赖引入

在项目的 pom.xml 文件添加点播 SDK 依赖即可：

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>vod_api</artifactId>
<version>2.1.4</version>
</dependency>
```

jar 包导入

如果项目没有采用 Maven 的方式进行依赖管理，可采用如下方式，下载各个所需的 jar 包，导入项目即可：

jar 文件	说明
vod_api-2.1.4.jar	云点播 SDK。
jackson-annotations-2.9.0.jar,jackson-core-2.9.7.jar,jackson-databind-2.9.7.jar,gson-2.2.4.jar	开源的 JSON 相关库。
cos_api-5.4.10.jar	腾讯云对象存储服务 COS SDK。
tencentcloud-sdk-java-3.1.2.jar	腾讯云 API SDK。
commons-codec-1.10.jar,commons-logging-1.2.jar,log4j-1.2.17.jar,slf4j-api-1.7.21.jar,slf4j-log4j12-1.7.21.jar	开源日志相关库。
httpclient-4.5.3.jar,httpcore-4.4.6.jar,okhttp-2.5.0.jar,okio-1.6.0.jar	开源的 HTTP 处理库。
joda-time-2.9.9.jar	开源时间处理库。
jaxb-api-2.3.0.jar	开源 XML 处理库。
bcprov-jdk15on-1.59.jar	开源加密处理库。

单击 Java SDK 关联 jar 包，将下载的 jar 包导入项目中，即可使用。

简单上传

初始化一个上传客户端对象

使用云 API 密钥初始化 VodUploadClient 实例。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
```

构造上传请求对象

设置媒体本地上传路径。

```
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
```

调用上传

调用上传方法，传入接入点地域及上传请求。

```
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

> 说明：
> 上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setCoverFilePath("/data/videos/Wildlife.jpg");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以使用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setProcedure("Your Procedure Name");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setSubAppId(101);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setStorageRegion("ap-chongqing");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

指定分片并发数

分片并发数是针对大文件，拆分成多个分片同时进行上传。分片并发上传的优势在于可以快速完成单个文件的上传，SDK 会根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。而文件的分片并发数通过 `ConcurrentUploadNumber` 参数进行指定。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setConcurrentUploadNumber(5);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

使用临时证书上传

传入临时证书的相关密钥信息，使用临时证书验证身份并进行上传。

```
VodUploadClient client = new VodUploadClient("Credentials TmpSecretId", "Credenti
als TmpSecretKey", "Credentials Token");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
```

```
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

设置代理上传

设置上传代理，涉及协议及数据都会经过代理进行处理，开发者可以借助代理在自己公司内网上传文件到腾讯云。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
HttpProfile httpProfile = new HttpProfile();
httpProfile.setProxyHost("your proxy ip");
httpProfile.setProxyPort(8080); //your proxy port
client.setHttpProfile(httpProfile);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

自适应码流文件上传

本 SDK 支持上传的自适应码流格式包括 HLS 和 DASH，同时要求 manifest（M3U8 或 MPD）所引用的媒体文件必须为相对路径（即不可以是 URL 和绝对路径），且位于 manifest 的同级目录或者下级目录（即不可以使用 `../` ）。在调用 SDK 上传接口时， `MediaFilePath` 参数填写 manifest 路径，SDK 会解析出相关的媒体文件列表一并上传。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/prog_index.m3u8");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// 业务方进行异常处理
logger.error("Upload Err", e);
}
```

接口描述

上传客户端类 `VodUploadClient`

属性名称	属性描述	类型	必填
secretId	云 API 密钥 ID。	String	是
secretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest`

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径，不支持 URL。	String	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	Integer	否
MediaType	待上传的媒体文件类型，可选类型请参见 视频上传综述，若 MediaFilePath 路径带后缀可不填。	String	否
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String	否
CoverFilePath	待上传的封面文件路径。必须为本地路径，不支持 URL。	String	否
CoverType	待上传的封面文件类型，可选类型请参见 视频上传综述，若 CoverFilePath 路径带后缀可不填。	String	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String	否
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口，创建分类，获得分类 ID。	Integer	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String	否

属性名称	属性描述	类型	必填
ConcurrentUploadNumber	分片并发数，针对大文件分片时有效。	Integer	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
FileId	媒体文件的唯一标识。	String
MediaUrl	媒体播放地址。	String
CoverUrl	媒体封面地址。	String
RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String

上传方法 `VodUploadClient.upload(String region, VodUploadRequest request)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是
request	上传请求。	VodUploadRequest	是

错误码表

状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

C# SDK

最近更新时间：2022-08-03 11:06:51

对于在服务端上传视频的场景，云点播提供 C# SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

通过 nuget 安装

1. 通过命令行安装：

```
dotnet add package VodSDK --version 1.0.1
```

2. 通过 Visual Studio 的 nuget 包管理工具，搜索 VodSDK 并安装。

通过源码包安装

如果项目中没有使用 nuget 工具，可以直接下载源码导入项目中使用：

- 从 Github 访问
- 单击下载 C# SDK

下载最新代码，解压后安装到项目工作目录下，使用 Visual Studio 2017 打开编译。因为此 SDK 还依赖外部的包，所以需要同时安装如下 SDK：

- 云 API SDK
- 对象存储 SDK

简单上传

初始化一个上传客户端对象

使用云 API 密钥初始化 VodUploadClient 实例。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
```

构造上传请求对象

设置媒体本地上传路径。

```
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
```

调用上传

调用上传方法，传入接入点地域及上传请求。

```
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// 打印媒体 FileId
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// 业务方进行异常处理
Console.WriteLine(e);
}
```

说明：
上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.CoverFilePath = "/data/videos/Wildlife.jpg";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// 打印媒体 FileId
```

```
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// 业务方进行异常处理
Console.WriteLine(e);
}
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以使用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.Procedure = "Your Procedure Name";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// 打印媒体 FileId
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// 业务方进行异常处理
Console.WriteLine(e);
}
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.SubAppId = 101;
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// 打印媒体 FileId
```

```
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// 业务方进行异常处理
Console.WriteLine(e);
}
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.StorageRegion = "ap-chongqing";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// 打印媒体 FileId
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// 业务方进行异常处理
Console.WriteLine(e);
}
```

接口描述

上传客户端类 `VodUploadClient`

属性名称	属性描述	类型	必填
secretId	云 API 密钥 ID。	String	是
secretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest`

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径，不支持 URL。	String	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	Integer	否
MediaType	待上传的媒体文件类型，可选类型请参见 视频上传综述，若 MediaFilePath 路径带后缀可不填。	String	否
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String	否
CoverFilePath	待上传的封面文件路径。必须为本地路径，不支持 URL。	String	否
CoverType	待上传的封面文件类型，可选类型请参见 视频上传综述，若 CoverFilePath 路径带后缀可不填。	String	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String	否
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口创建分类，获得分类 ID。	Integer	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
FileId	媒体文件的唯一标识。	String
MediaUrl	媒体播放地址。	String
CoverUrl	媒体封面地址。	String
RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String

上传方法 `VodUploadClient.Upload(String region, VodUploadRequest request)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是
request	上传请求。	VodUploadRequest	是

错误码表

状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

PHP SDK

最近更新时间：2023-02-16 16:18:31

对于在服务端上传视频的场景，云点播提供 PHP SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

使用 composer 引入

```
{
"require": {
"qcloud/vod-sdk-v5": "v2.4.0"
}
}
```

通过源码包安装

1. 如果项目中没有使用 composer 工具进行依赖管理，从 Github 访问 可以直接下载源码，导入项目中使用。
2. 解压 vod-sdk.zip 文件到项目中，引入 autoload.php 文件即可使用。

简单视频上传

初始化上传对象

使用云 API 密钥初始化 VodUploadClient 实例。

使用 composer 导入

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

use Vod\VodUploadClient;

$client = new VodUploadClient("your secretId", "your secretKey");
```

使用源码导入

```
<?php
require 'vod-sdk-v5/autoload.php';
```

```
use Vod\VodUploadClient;

$client = new VodUploadClient("your secretId", "your secretKey");
```

构造上传请求对象

```
use Vod\Model\VodUploadRequest;

$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
```

调用上传

调用上传方法，传入接入点地域及上传请求。

```
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

说明：
上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
```

```
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->CoverFilePath = "/data/videos/Wildlife-Cover.png";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
echo "CoverUrl -> ". $rsp->CoverUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->Procedure = "Your Procedure Name";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;
```

```
$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->SubAppId = 101;
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->StorageRegion = "ap-chongqing";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

使用临时证书上传

传入临时证书的相关密钥信息，使用临时证书验证身份并进行上传。

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

```php
use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKe
y", "Credentials Token");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

设置代理上传

设置上传代理，涉及协议及数据都会经过代理进行处理，开发者可以借助代理在自己公司内网上传文件到腾讯云。

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;
use Vod\Model\VodUploadHttpProfile;

$client = new VodUploadClient("your secretId", "your secretKey");
$uploadHttpProfile = new VodUploadHttpProfile("your proxy addr");
$client->setHttpProfile($uploadHttpProfile);
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

自适应码流文件上传

本 SDK 支持上传的自适应码流格式包括 HLS 和 DASH，同时要求 manifest（M3U8 或 MPD）所引用的媒体文件必须为相对路径（即不可以是 URL 和绝对路径），且位于 manifest 的同级目录或者下级目录（即不可以使

用 `../` ）。在调用 SDK 上传接口时， `MediaFilePath` 参数填写 manifest 路径，SDK 会解析出相关的媒体文件列表一并上传。

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

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/prog_index.m3u8";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// 处理上传异常
echo $e;
}
```

接口描述

上传客户端类 `VodUploadClient`

属性名称	属性描述	类型	必填
secretId	云 API 密钥 ID。	String	是
secretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest`

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径，不支持 URL。	String	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	Integer	否
MediaType	待上传的媒体文件类型，可选类型请参见 视频上传综述，若 MediaFilePath 路径带后缀可不填。	String	否

属性名称	属性描述	类型	必填
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String	否
CoverFilePath	待上传的封面文件路径。必须为本地路径，不支持 URL。	String	否
CoverType	待上传的封面文件类型，可选类型请参见 视频上传综述，若 CoverFilePath 路径带后缀可不填。	String	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String	否
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口，创建分类，获得分类 ID。	Integer	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
FileId	媒体文件的唯一标识。	String
MediaUrl	媒体播放地址。	String
CoverUrl	媒体封面地址。	String
RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String

上传方法 `VodUploadClient.upload(String region, VodUploadRequest request)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是
request	上传请求。	VodUploadRequest	是

错误码表

状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

Python SDK

最近更新时间：2022-06-24 16:07:16

对于在服务端上传视频的场景，云点播提供 Python SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

使用 pip 安装

```
pip install vod-python-sdk
```

通过源码包安装

如果项目中没有使用 pip 工具，可以直接下载源码导入项目中使用：

- 从 Github 访问
- 单击下载 Python SDK

下载最新代码，解压后：

```
$ cd vod-python-sdk
$ python setup.py install
```

简单视频上传

初始化上传对象

使用云 API 密钥初始化 VodUploadClient 实例。

```
from qcloud_vod.vod_upload_client import VodUploadClient
client = VodUploadClient("your secretId", "your secretKey")
```

构造上传请求对象

```
from qcloud_vod.model import VodUploadRequest
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
```

调用上传

调用上传方法，传入接入点地域及上传请求。

```
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

说明：
上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.CoverFilePath = "/data/file/Wildlife-Cover.png"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
print(response.CoverUrl)
except Exception as err:
# 处理业务异常
print(err)
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.Procedure = "Your Procedure Name"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.SubAppId = 101
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.StorageRegion = "ap-chongqing"
try:
```

```
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

指定分片并发数

分片并发数是针对大文件，拆分成多个分片同时进行上传。分片并发上传的优势在于可以快速完成单个文件的上传，SDK 会根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。而文件的分片并发数通过 `ConcurrentUploadNumber` 参数进行指定。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.ConcurrentUploadNumber = 5
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

使用临时证书上传

传入临时证书的相关密钥信息，使用临时证书验证身份并进行上传。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKey",
"Credentials Token")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

自适应码流文件上传

本 SDK 支持上传的自适应码流格式包括 HLS 和 DASH，同时要求 manifest（M3U8 或 MPD）所引用的媒体文件必须为相对路径（即不可以是 URL 和绝对路径），且位于 manifest 的同级目录或者下级目录（即不可以使用 `../` ）。在调用 SDK 上传接口时， `MediaFilePath` 参数填写 manifest 路径，SDK 会解析出相关的媒体文件列表一并上传。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest
client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/prog_index.mp4"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
# 处理业务异常
print(err)
```

接口描述

上传客户端类 `VodUploadClient` ：

属性名称	属性描述	类型	必填
secretId	云 API 密钥 ID。	String	是
secretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest` ：

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径，不支持 URL。	String	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	Integer	否
MediaType	待上传的媒体文件类型，可选类型请参见 视频上传综述，若 MediaFilePath 路径带后缀可不填。	String	否

属性名称	属性描述	类型	必填
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String	否
CoverFilePath	待上传的封面文件路径。必须为本地路径，不支持 URL。	String	否
CoverType	待上传的封面文件类型，可选类型请参见 视频上传综述，若 CoverFilePath 路径带后缀可不填。	String	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String	否
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口，创建分类，获得分类 ID。	Integer	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String	否
ConcurrentUploadNumber	分片并发数，针对大文件分片时有效。	Integer	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
FileId	媒体文件的唯一标识。	String
MediaUrl	媒体播放地址。	String
CoverUrl	媒体封面地址。	String
RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String

上传方法 `VodUploadClient.upload(String region, VodUploadRequest request)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是

参数名称	参数描述	类型	必填
request	上传请求。	VodUploadRequest	是

错误码表

状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

Node.js SDK

最近更新时间：2022-06-24 15:57:42

对于在服务端上传视频的场景，云点播提供 Node.js SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

使用 npm 安装

```
npm i vod-node-sdk --save
```

通过源码包安装

如果项目中没有使用 npm 工具进行依赖管理，可以直接下载源码导入项目中使用：

- 从 Github 访问
- 单击下载 Node.js SDK

简单视频上传

初始化上传对象

使用云 API 密钥初始化 VodUploadClient 实例。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
```

构造上传请求对象

```
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
```

调用上传

调用上传方法，传入接入点地域及上传请求，通过回调方法获取返回的信息。

```
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
```

```
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

说明：
上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.CoverFilePath = "/data/file/Wildlife-cover.png";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
console.log(data.CoverUrl);
}
});
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
```

```
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.Procedure = "Your Procedure Name";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.SubAppId = 101;
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.StorageRegion = "ap-chongqing";
client.upload("ap-guangzhou", req, function (err, data) {
```

```
if (err) {
// 处理业务异常
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

使用临时证书上传

传入临时证书的相关密钥信息，使用临时证书验证身份并进行上传。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKe
y", "Credentials Token");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
console.log(err)
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

自适应码流文件上传

本 SDK 支持上传的自适应码流格式包括 HLS 和 DASH，同时要求 manifest（M3U8 或 MPD）所引用的媒体文件必须为相对路径（即不可以是 URL 和绝对路径），且位于 manifest 的同级目录或者下级目录（即不可以使用 `../` ）。在调用 SDK 上传接口时， `MediaFilePath` 参数填写 manifest 路径，SDK 会解析出相关的媒体文件列表一并上传。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');
client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/prog_index.m3u8";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// 处理业务异常
console.log(err)
```

```
} else {
// 获取上传成功后的信息
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

接口描述

上传客户端类 `VodUploadClient`

属性名称	属性描述	类型	必填
secretId	云 API 密钥 ID。	String	是
secretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest`

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径（用户服务器上的路径），不支持 URL。	String	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	Integer	否
MediaType	待上传的媒体文件类型，可选类型请参见 视频上传综述，若 MediaFilePath 路径带后缀可不填。	String	否
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String	否
CoverFilePath	待上传的封面文件路径。必须为本地路径（用户服务器上的路径），不支持 URL。	String	否
CoverType	待上传的封面文件类型，可选类型请参见 视频上传综述，若 CoverFilePath 路径带后缀可不填。	String	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String	否

属性名称	属性描述	类型	必填
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口，创建分类，获得分类 ID。	Integer	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
FileId	媒体文件的唯一标识。	String
MediaUrl	媒体播放地址。	String
CoverUrl	媒体封面地址。	String
RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String

上传方法 `VodUploadClient.upload(String region, VodUploadRequest request, function callback)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是
request	上传请求。	VodUploadRequest	是
callback	上传完成回调函数。	function	是

上传完成回调函数 `function(err, data)`

参数名称	参数描述	类型	必填
err	错误信息。	Exception	是
data	上传响应结果。	VodUploadResponse	是

错误码表

状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

Go SDK

最近更新时间：2022-06-24 15:40:13

对于在服务端上传视频的场景，云点播提供 Go SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

使用 go get 引入

```
go get -u github.com/tencentcloud/tencentcloud-sdk-go
go get -u github.com/tencentyun/cos-go-sdk-v5
go get -u github.com/tencentyun/vod-go-sdk
```

通过源码包安装

如果项目中需要直接引用源码，可以直接下载源码导入项目中使用：

- 从 Github 访问
- 单击下载 Go SDK

简单视频上传

初始化上传对象

使用云 API 密钥初始化 VodUploadClient 实例。

```
import (
"github.com/tencentyun/vod-go-sdk"
)
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"
```

构造上传请求对象

```
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
)
req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
```

调用上传

调用上传方法，传入接入点地域及上传请求。

```
rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
```

说明：
上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

高级功能

携带封面

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.CoverFilePath = common.StringPtr("/data/video/Wildlife-cover.png")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
```

```
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
fmt.Println(*rsp.Response.CoverUrl)
}
```

指定任务流

首先 创建任务流模板 并为模板命名，发起任务流时，可以用任务流模板名设置 `Procedure` 参数，上传成功后会自动执行任务流。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.Procedure = common.StringPtr("Your Proceducre Name")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

子应用上传

传入 子应用 ID，上传成功后资源只属于具体的子应用。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
```

```
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.SubAppId = common.Uint64Ptr(101)

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

指定存储地域

在 控制台 确认已经开通目标存储地域，若没有开通可以参考 上传存储设置，最后通过 `StorageRegion` 属性设置存储地域的 英文简称。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.StorageRegion = common.StringPtr("ap-chongqing")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

指定分片并发数

分片并发数是针对大文件，拆分成多个分片同时进行上传。分片并发上传的优势在于可以快速完成单个文件的上传，SDK 会根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。而文件的分片并发数通过 `ConcurrentUploadNumber` 参数进行指定。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.ConcurrentUploadNumber = common.Uint64Ptr(5)

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

使用临时证书上传

传入临时证书的相关密钥信息，使用临时证书验证身份并进行上传。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "Credentials TmpSecretId"
client.SecretKey = "Credentials TmpSecretKey"
client.Token = "Credentials Token"
```

```
req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

设置代理上传

设置上传代理，涉及协议及数据都会经过代理进行处理，开发者可以借助代理在自己公司内网上传文件到腾讯云。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
"net/http"
"net/url"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"
proxyUrl, _ := url.Parse("your proxy url")
client.Transport = &http.Transport{
Proxy: http.ProxyURL(proxyUrl),
}

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

自适应码流文件上传

本 SDK 支持上传的自适应码流格式包括 HLS 和 DASH，同时要求 manifest（M3U8 或 MPD）所引用的媒体文件必须为相对路径（即不可以是 URL 和绝对路径），且位于 manifest 的同级目录或者下级目录（即不可以使用 `../` ）。在调用 SDK 上传接口时， `MediaFilePath` 参数填写 manifest 路径，SDK 会解析出相关的媒体文件列表一并上传。

```
package main
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/prog_index.m3u8")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
fmt.Println(*rsp.Response.CoverUrl)
}
```

接口描述

上传客户端类 `VodUploadClient`

属性名称	属性描述	类型	必填
SecretId	云 API 密钥 ID。	String	是
SecretKey	云 API 密钥 Key。	String	是

上传请求类 `VodUploadRequest`

属性名称	属性描述	类型	必填
MediaFilePath	待上传的媒体文件路径。必须为本地路径，不支持 URL。	String 指针	是
SubAppId	云点播 子应用 ID。如果要访问子应用中的资源，则将该字段填写为子应用 ID，否则无需填写该字段。	uint64 指针	否
MediaType	待上传的媒体文件类型，可选类型请参见 媒体上传综述，若 MediaFilePath 路径带后缀可不填。	String 指针	否
MediaName	上传后的媒体名称，若不填默认采用 MediaFilePath 的文件名。	String 指针	否
CoverFilePath	待上传的封面文件路径。必须为本地路径，不支持 URL。	String 指针	否
CoverType	待上传的封面文件类型，可选类型请参见 媒体上传综述，若 CoverFilePath 路径带后缀可不填。	String 指针	否
Procedure	上传后需要自动执行的任务流名称，该参数在创建任务流（API 方式 或 控制台方式）时由用户指定。具体请参考 任务流综述。	String 指针	否
ExpireTime	媒体文件过期时间，格式按照 ISO 8601 标准表示，详见 ISO 日期格式说明。	String 指针	否
ClassId	分类 ID，用于对媒体进行分类管理，可通过 创建分类 接口，创建分类，获得分类 ID。	int64 指针	否
SourceContext	来源上下文，用于透传用户请求信息，上传回调接口将返回该字段值，最长250个字符。	String 指针	否
StorageRegion	存储地域，指定预期希望存储的地域，该字段填写为存储地域的 英文简称。	String 指针	否
ConcurrentUploadNumber	分片并发数，针对大文件分片时有效。	Integer	否

上传响应类 `VodUploadResponse`

属性名称	属性描述	类型
Response	上传返回结果信息。	struct
Response.FileId	媒体文件的唯一标识。	String 指针

属性名称	属性描述	类型
Response.MediaUrl	媒体播放地址。	String 指针
Response.CoverUrl	媒体封面地址。	String 指针
Response.RequestId	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。	String 指针

上传方法 `VodUploadClient.Upload(region string, request *VodUploadRequest)`

参数名称	参数描述	类型	必填
region	接入点地域，即请求到哪个地域的云点播服务器，不同于存储地域，具体参考支持的 地域列表。	String	是
request	上传请求。	VodUploadRequest 指针	是

错误码表

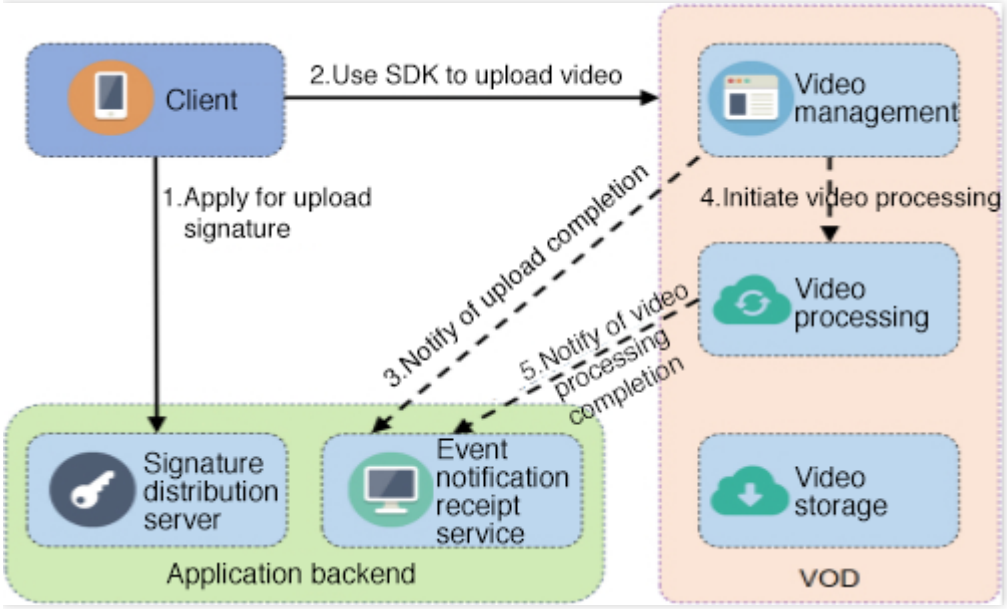
状态码	含义
InternalError	内部错误。
InvalidParameter.ExpireTime	参数值错误：过期时间。
InvalidParameterValue.CoverType	参数值错误：封面类型。
InvalidParameterValue.MediaType	参数值错误：媒体类型。
InvalidParameterValue.SubAppId	参数值错误：子应用 ID。
InvalidParameterValue.VodSessionKey	参数值错误：点播会话。
ResourceNotFound	资源不存在。

客户端上传
客户端上传指引

最近更新时间：2021-10-29 11:13:43

操作场景

客户端视频上传是指 App 的最终用户将本地视频上传到云点播平台，其流程图如下。本文将为您介绍如何使用客户端上传视频。



前提条件

1. 开通服务

开通云点播服务。

2. 获取云 API 密钥

获取调用服务端 API 所需的安全凭证，即 SecretId 和 SecretKey，具体步骤如下：

1. 登录控制台，选择【云产品】>【访问管理】>【API密钥管理】，进入“API 密钥管理”页面。
2. 获取云 API 密钥。如果您尚未创建密钥，则单击【新建密钥】即可创建一对 SecretId 和 SecretKey。

操作步骤

1. 申请上传签名

客户端需要向 App 的签名派发服务器申请上传签名，签名生成步骤请参见 客户端上传签名。多语言签名生成示例请参见：

- PHP 签名示例
- Java 签名示例
- Node.js 签名示例
- C# 签名示例
- Python 签名示例

> 说明：
>
> - 由于客户端上传是直接将视频文件从客户端上传到云点播平台，无需 App 服务端中转，所以云点播必须对发起请求的客户端进行鉴权。
> - 由于 SecretKey 权限过大，App 不能将该信息泄露到客户端，否则会造成严重的安全问题，所以客户端发起上传前需要申请上传签名。

2. 使用 SDK 上传视频

为了方便客户端的视频上传，云点播提供多平台 SDK 方便客户接入，详细请参见：

- Android 上传 SDK
- iOS 上传 SDK
- Web 上传 SDK

高级功能

- **上传时指定任务流**
  如果开发者需要在视频上传完成后自动发起 视频处理任务流（例如转码、截图等），可以在生成 上传签名 时通过 `procedure` 参数来实现，参数值为任务流模板名，云点播支持 创建任务流模板 并为模板命名，发起任务流时，可以用任务流模板名来表示要发起的任务。
- **上传时指定存储地域**
  云点播默认提供的存储地域设置在新加坡，如果需要存储到其他区域，可以在控制台上自助添加其他存储地域，详细请参见 上传存储设置。设置完成后，在生成 上传签名 时通过 `storageRegion` 参数来指定，参数值为存储地域的 英文简称。
- **上传时附带封面**
  云点播允许在视频上传过程中，携带封面上传，即在上传 SDK 接口中填写相关的封面路径，详细请参见：

- Android 上传 SDK
- iOS 上传 SDK
- Web 上传 SDK

- **单次有效签名**

  在视频上传过程中，App 后台派发的签名默认在有效期内是可以多次使用的。如果 App 对视频上传安全性要求很高，可以指定签名单次有效。

  单次有效签名的使用方式：在 App 后台派发签名时，指定参数 `oneTimeValid` 为1即可，详细请参见 客户端上传签名。

  > 说明：
  > 单次有效签名有且只能被使用一次，该签名方式虽然更加安全，但是需要 App 做额外的异常处理，例如，上传出错时，不能简单地重复使用 SDK 上传视频，还需要重新申请上传签名。

- **断点续传**

  在视频上传过程中，如果上传意外终止，用户再次上传该文件，则可以从中断处继续上传。

  > 说明：
  > 断点续传的有效时间为1天，即同一个视频上传被中断，那么1天内再次上传可以直接从断点处上传，超过1天则默认会重新上传完整视频。

  App 开启断点续传功能的方式如下：
  - Android 上传 SDK，上传时设置 `enableResume` 字段为 True 即可。
  - iOS 上传 SDK，上传时设置 `enableResume` 字段为 True 即可。
  - Web 上传 SDK，内置断点续传，无需用户操作。

- **暂停/恢复/取消上传**

  在视频上传过程中，云点播 SDK 允许暂停、恢复或取消上传，详细请参见：
  - Android 上传 SDK
  - iOS 上传 SDK
  - Web 上传 SDK

常见问题

- **如何在视频上传完成后自动发起转码？**

  通过客户端上传签名中 `procedure` 参数指定视频上传完成后的处理方式，详细请参见 上传时指定任务流。

- **App 后台收到视频上传完成的通知后，如何识别是哪个客户上传的？**

  在客户端上传签名中增加 `sourceContext` 参数，通过该参数来携带用户身份信息，上传完成通知会将该参数

传递给 App 后台，详细请参见 事件通知。

3. 事件通知

视频上传完成之后，云点播会给 App 后台发起 事件通知 - 视频上传完成，App 后台可通过该事件感知到视频上传行为。如果要接收事件通知，则 App 需要到 控制台 - 回调设置 开启事件通知。事件通知 - 视频上传完成 主要包含如下信息：

- 新视频文件的 FileId 和 URL。
- 云点播支持在视频上传时指定透传字段，事件完成将透传字段通知到 App 后台。在事件通知中有如下字段：
  - `SourceType` ：该字段被腾讯云固定成 `ServerUpload` ，表示上传来源为服务端上传。
  - `SourceContext` ：用户自定义透传字段，App 后台在派发签名时指定的透传内容，对应签名中的 `sourceContext` 参数。
- 云点播支持在视频上传完成时自动发起视频处理，如果上传时指定了 视频处理任务流，则在事件通知内容中也会携带任务 ID，即事件通知中的 `data.procedureTaskId` 字段。

更多信息请参见：

- 任务管理与事件通知
- 事件通知 - 视频上传完成

客户端上传加速

最近更新时间：2022-09-15 17:35:34

客户端上传加速可为客户提供更高质量的上传服务。该功能基于腾讯云全球部署的加速网络，根据用户的请求，智能地选择最佳接入点和最优链路，提高上传的速度和成功率。同时支持使用 QUIC 协议进行数据传输，改善数据传输的效率和在弱网环境中的稳定性。

影响上传质量的主要因素

长距离传输

点播在全球很多地区部署了存储中心，客户可以选择性地启用这些存储中心，并且在上传时就近存储，详情参见 就近上传。但即便如此，还是无法避免一些终端用户与存储中心距离间隔太远，以及部分客户的业务场景中存在需跨地区甚至跨海上传的场景。长距离的数据上传通常就意味着更长的网络链路和更大的传输时延，而且一旦中间某一个环节出现网络抖动、丢包等问题，就会拉低整条链路的上传速度和成功率。

弱网问题

弱网简单来说就是质量较差的网络环境，如时延大、丢包率高的网络。如今移动网络普及甚广，移动端客户的上传量占比非常大。而移动网络的弱网问题十分常见，如用户处在基站信号覆盖较弱的地区，或者行动过程中联网设备频繁切换网络。如何在弱网环境中保持稳定的数据传输，是提高上传质量的一大难题。

网络协议效率不高

点播客户上传的文件大部分是数据量较大的视频文件，而目前在上传中使用最频繁的网络协议依然是 HTTP1.1。该协议本质还是基于串行模型，同时还存在队头阻塞等问题，在大规模数据传输的场景中，很容易触达性能瓶颈。

客户端上传加速的方案

全球链路加速，高保障通道

针对长距离传输场景，网络链路过长导致的上传质量不佳问题，云点播基于腾讯云遍布全球的加速网络和边缘节点，为客户提供一套数据上传的全球链路加速通道。借助腾讯云的智能全局流量管理平台，将用户的上传请求递交给最靠近用户的边缘节点，就近接收用户的数据。再通过腾讯云打磨多年的加速网络，选择最优链路，将数据传送到存储中心。

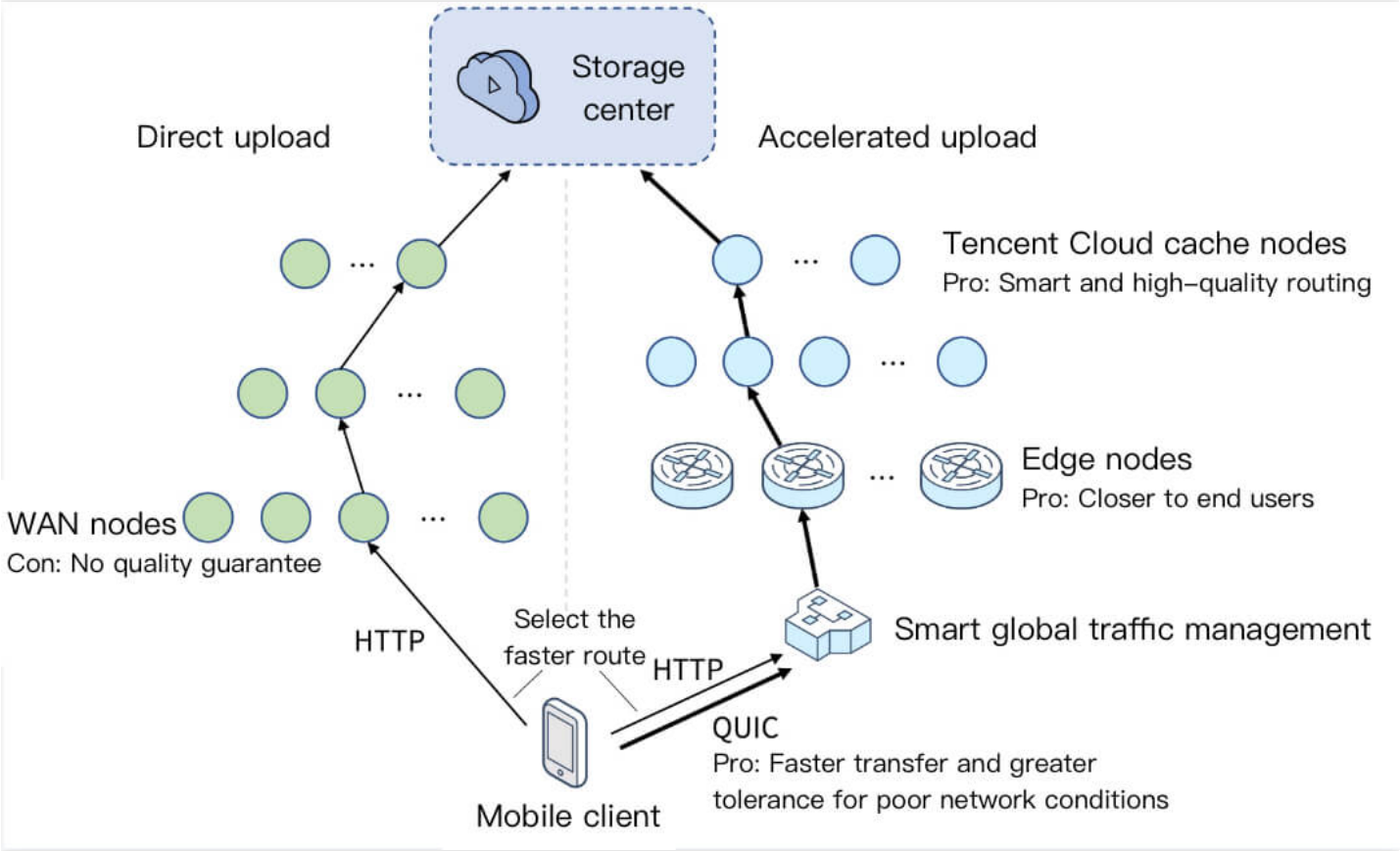
QUIC 协议，更快更稳定

针对弱网环境和网络协议效率不高的问题，云点播提供的客户端上传已支持 QUIC 协议。QUIC 协议是一种基于 UDP 实现的低延迟高可靠通信协议，目前标准的 HTTP3 协议正是基于 QUIC 实现的。QUIC 支持 0-RTT 建立连接

和无队头阻塞的多路复用，可更大限度地利用网络带宽进行实际的数据传输，在丢包率和网络延迟较高的弱网环境也可提供高质量的数据通信。同时，QUIC 还支持连接迁移，在移动端频繁切换网络的场景中，也可平滑过渡，保证网络不中断。

智能择优，使用无门槛

云点播提供的上传加速方案简单易用，只需客户在控制台开启相关功能即可。在使用 SDK 进行上传时，会智能地在普通通道和加速通道中进行竞速和择优，同时会自动探测并决定是否使用 QUIC 协议进行数据上传。



使用方法

下面轻松两步，即可开启客户端上传加速功能：

1. 参照客户端上传加速的 控制台操作 指引，开启“全球链路加速”，并按需开启“QUIC 传输”。
2. Android 和 iOS 平台需确保在 App 启动时调用了 预上传 。若开启“QUIC 传输”，Android 平台需使用版本不低于 9.6 的 SDK，iOS 平台需使用版本不低于 10.4 的 SDK。

> 说明：
>
> - Android、iOS 上传 SDK，可同时支持上传加速和 QUIC 传输；

- Web 端、小程序端上传 SDK，仅支持上传加速，当前暂不支持 QUIC 传输。

费用相关

使用客户端上传加速，将涉及以下费用：

- 全局链路加速费用：使用全局链路加速时，产生的上传加速流量费用
- QUIC 传输费用：使用 QUIC 传输时，产生的上传加速流量费用

以上费用的具体价格，请参见 购买指南。

客户端上传签名

最近更新时间：2024-05-16 14:48:59

客户端在发起上传前，需要向 App 的签名派发服务器申请上传签名。客户端执行上传操作时，必须携带该签名，以便云点播验证客户端的上传是否被授权。

签名生成步骤

1. **获取云 API 密钥**

获取调用服务端 API 所需的安全凭证，即 SecretId 和 SecretKey，具体步骤如下：

2. 登录控制台，选择**云产品** > **访问管理** > API 密钥管理，进入“API 密钥管理”页面。

3. 获取云 API 密钥。如果您尚未创建密钥，则单击**新建密钥**即可创建一对 SecretId 和 SecretKey。

4. **拼接明文串 original。**

按照 URL QueryString 的格式要求拼接签名明文串 original，其格式如下：

```
secretId=[secretId]&currentTimeStamp=[currentTimeStamp]&expireTime=[expireTime]&ran
```

注意：

上述 original 中的 `[secretId]`、`[currentTimeStamp]`、`[expireTime]` 及 `[random]` 需您自行替换成具体的参数值。

original 至少包含 `secretId`、`currentTimeStamp`、`expireTime` 及 `random` 四个必选参数，可包含任意多个选填参数，详细请参见 签名参数。

参数值必须经过 UrlEncode，否则可能导致 QueryString 解析失败。

5. **将明文串转为最终签名**（以部分 Java 代码为例）。

6. 用已获取的 SecretKey 对明文串 original 进行 HMAC-SHA1 加密，得到 signatureTmp：

```
 Mac mac = Mac.getInstance("HmacSHA1");
SecretKeySpec secretKey = new SecretKeySpec(this.secretKey.getBytes("UTF-8"), mac.g
mac.init(secretKey);
 byte[] signatureTmp = mac.doFinal(original.getBytes("UTF-8"));
```

说明：

signatureTmp 是使用 UTF-8 编码、通过 HMAC-SHA1 加密出来的字节数组。

7. 将明文串 original 使用 UTF-8 编码成字节数组，然后把 signatureTmp 与该数组进行合并，最后把合并后的结果进行 Base64 编码，得到最终签名 signature：

```
String signature = base64Encode(byteMerger(signatureTmp, original.getBytes("utf8"))
```

说明：

byteMerger 和 base64Encode 分别是数组合并和 Base64 编码的方法，详细请参见 Java 签名示例代码。

签名生成示例

云点播还提供了**签名生成示例代码**和签名工具，便于您参考和验证：

客户端上传 - 签名生成示例代码

点播客户端上传 - 签名生成工具

点播客户端上传 - 签名校验工具

签名参数说明

参数名称	必选	类型	说明
secretId	是	String	云 API 密钥中的 SecretId，获取方式请参见 客户端上传指引 - 获取云 API 密钥。
currentTimeStamp	是	Integer	当前 Unix 时间戳。
expireTime	是	Integer	签名到期 Unix 时间戳。 `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。十进制数，最大值 `4294967295` （2^32-1，即32位无符号二进制数的最大值）。
classId	否	Integer	视频文件分类，默认为0。
procedure	否	String	视频后续任务处理操作，即完成视频上传后，可自动发起任务流操作。参数值为任务流模板名，云点播支持 创建任务流模板 并为模板命名。
taskPriority	否	Integer	视频后续任务优先级（仅当指定了 procedure 时才有效），取值范围为[-10，10]，默认为0。
taskNotifyMode	否	String	任务流状态变更通知模式（仅当指定了 procedure 时才有效）。 Finish：只有当任务流全部执行完毕时，才发起一次事件通知。 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。 None：不接受该任务流回调。 默认为 Finish。
sourceContext	否	String	来源上下文，用于透传用户请求信息，上传完成回调 将返回该字段

			值，最长250个字符。
oneTimeValid	否	Integer	签名是否单次有效，详细请参见 客户端上传指引 - 单次有效签名。 默认为0，表示不启用；1表示签名单次有效。 相关错误码详见 单次有效签名说明。
vodSubAppId	否	Integer	子应用 ID，如果不填写、填写0或填写开发者的腾讯云 AppId，则操作的子应用为“主应用”。
sessionContext	否	String	会话上下文，用于透传用户请求信息，当指定 procedure 参数后，任务流状态变更回调 将返回该字段值，最长 1000 个字符。
storageRegion	否	String	指定存储地域，可以在控制台上自助添加存储地域，详细请参见 上传存储设置，该字段填写为存储地域的 英文简称。

单次有效签名说明

使用单次有效签名后，签名服务器需要保证每次派发给用户的签名不相同（如保证同一个时间点派发的签名 `random` 不重复），否则会导致重复签名的错误。

签名错误导致的上传失败，如果重试，则需要获取新的签名。

Android 和 Java SDK 签名错误引起的错误状态码是 `1001` 。

签名生成示例

最近更新时间：2020-12-09 19:06:20

PHP 签名示例

```php
<?php
// 确定 App 的云 API 密钥
$secret_id = "XXXXXXXXXXXXXXXXXXX";
$secret_key = "AAAAAAAAAAAAAAAAAAA";

// 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
$original = http_build_query($arg_list);
$signature = base64_encode(hash_hmac('SHA1', $original, $secret_key, true).$origi
nal);

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

Java 签名示例

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

// 签名工具类
class Signature {
private String secretId;
```

```
private String secretKey;
private long currentTime;
private int random;
private int signValidDuration;

private static final String HMAC_ALGORITHM = "HmacSHA1"; //签名算法
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;
}

// 获取签名
public String getUploadSignature() throws Exception {
String strSign = "";
String contextStr = "";

// 生成原始参数字符串
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_CHARS
ET), mac.getAlgorithm());
mac.init(secretKey);

byte[] hash = mac.doFinal(contextStr.getBytes(CONTENT_CHARSET));
byte[] sigBuf = byteMerger(hash, contextStr.getBytes("utf8"));
strSign = base64Encode(sigBuf);
strSign = strSign.replace(" ", "").replace("\n", "").replace("\r", "");
} catch (Exception e) {
throw e;
}
return strSign;
}

private String base64Encode(byte[] buffer) {
BASE64Encoder encoder = new BASE64Encoder();
return encoder.encode(buffer);
}
```

```
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();
// 设置 App 的云 API 密钥
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); // 签名有效期：2天

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

对于 Java1.9 以上的版本，已经移除了 `sun.misc.BASE64Encoder` 相关的包，可以使用 `java.util.Base64` 替换 `base64Encode` 方法中对应的实现，具体请参考如下代码：

```
import java.util.Base64;

private String base64Encode(byte[] buffer) {
Base64.Encoder encoder = Base64.getEncoder();
return encoder.encodeToString(buffer);
}
```

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 = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
var secret_key = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";

// 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());
}
}
```

Python 签名示例

```
#!/usr/local/bin/python3
#coding=utf-8

import time
import random
import hmac
import hashlib
import base64

SecretId = 'IamSecretId'
SecretKey = 'IamSecretKey'
#TimeStamp = int(time.time())
TimeStamp = 1571215095
ExpireTime = TimeStamp + 86400 * 365 * 10
#Random = random.randint(0, 999999)
Random = 220625

Original = "secretId=" + SecretId + "&currentTimeStamp=" + str(TimeStamp) + "&exp
ireTime=" + str(ExpireTime) + "&random=" + str(Random)

Hmac = hmac.new(bytes(SecretKey, 'utf-8'), bytes(Original, 'utf-8'), hashlib.sha
1)
Sha1 = Hmac.digest()
Signature = bytes(Sha1) + bytes(Original, 'utf-8')
Signature2 = base64.b64encode(Signature)

#return str(signature2, 'UTF-8')

print("Original: ", Original)
print("HMAC-SHA1: ", Sha1)
print("Signature before BASE64: ", Signature)
print("Signature after BASE64: ", str(Signature2))
```

Go 签名示例

```
package main

import (
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
"fmt"
"math/rand"
"strconv"
"time"
)

func generateHmacSHA1(secretToken, payloadBody string) []byte {
mac := hmac.New(sha1.New, []byte(secretToken))
sha1.New()
mac.Write([]byte(payloadBody))
return mac.Sum(nil)
}

func main() {
rand.Seed(time.Now().Unix())
secretId := "IamSecretId"
secretKey := "IamSecretKey"
// timestamp := time.Now().Unix()
timestamp := int64(1571215095)
expireTime := timestamp + 86400*365*10
timestampStr := strconv.FormatInt(timestamp, 10)
expireTimeStr := strconv.FormatInt(expireTime, 10)

random := 220625
randomStr := strconv.Itoa(random)
original := "secretId=" + secretId + "&currentTimeStamp=" + timestampStr + "&expi
reTime=" + expireTimeStr + "&random=" + randomStr
signature := generateHmacSHA1(secretKey, original)
signature = append(signature, []byte(original)...)
signatureB64 := base64.StdEncoding.EncodeToString(signature)
fmt.Println(signatureB64)
}
```

Web 端上传 SDK

最近更新时间：2023-03-07 11:20:50

对于浏览器上传音视频的场景，云点播提供了 Web 上传 SDK。如果您需要 SDK 源码，可访问 SDK 源码。

简单视频上传

引入 SDK

script 引入方式

未使用 webpack 的情况下，可通过 script 标签方式引入，该方式会暴露全局的 `TcVod` 变量。script 引入有下面两种方式：

- **下载到本地**
  下载 SDK 源码 到本地，然后按以下方式引入：

```
<script src="./vod-js-sdk-v6.js"></script>
```

> 说明：
> 引入路径请自行调整为您本地保存的路径。

- **使用 CDN 资源**
  使用 CDN 资源，可直接按以下方式引入：

```
<script src="https://cdn-go.cn/cdn/vod-js-sdk-v6/latest/vod-js-sdk-v6.js"></scr
ipt>
```

请 单击此处 查看 script 方式引入的 Demo，请 单击此处 查看 Demo 源码。

npm 引入方式

使用 webpack 的情况下（如使用 Vue 或者 React），可通过 npm 引入：

```
// npm install vod-js-sdk-v6 之后，在页面中直接 import 引入
import TcVod from 'vod-js-sdk-v6'
```

请 单击此处 查看 npm 方式引入的 Demo 源码。

注意：
SDK 依赖 Promise，请在低版本浏览器中自行引入。

定义获取上传签名的函数

```
function getSignature() {
return axios.post(url).then(function (response) {
return response.data.signature;
})
};
```

说明：

- `url` 是您派发签名服务的 URL，更多相关信息请参见 客户端上传指引。
- `signature` 计算规则请参见 客户端上传签名。
- **子应用、视频文件分类、任务流**等信息都包含在上传签名中，更多相关信息请参见 签名参数说明。

上传视频示例

```
// 通过 import 引入的话，new TcVod(opts) 即可
// new TcVod.default(opts) 是 script 引入 的用法
const tcVod = new TcVod.default({
getSignature: getSignature // 前文中所述的获取上传签名的函数
})

const uploader = tcVod.upload({
mediaFile: mediaFile, // 媒体文件（视频或音频或图片），类型为 File
})
uploader.on('media_progress', function(info) {
console.log(info.percent) // 进度
})

// 回调结果说明
// type doneResult = {
// fileId: string,
// video: {
// url: string
// },
// cover: {
// url: string
```

```
// }
// }
uploader.done().then(function (doneResult) {
// deal with doneResult
}).catch(function (err) {
// deal with error
})
```

说明：

- `new TcVod(opts)` 中的 opts 指该接口的相关参数，详细请参见 TcVod 接口描述。
- 上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。
- 如需上传至指定子应用下，请参见 子应用体系 - 客户端上传。

高级功能

同时上传视频和封面

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})

uploader.done().then(function (doneResult) {
// deal with doneResult
})
```

获取上传进度

SDK 支持以回调的形式展示当前的上传进度：

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})
// 视频上传完成时
uploader.on('media_upload', function(info) {
uploaderInfo.isVideoUploadSuccess = true;
```

```
})
// 视频上传进度
uploader.on('media_progress', function(info) {
uploaderInfo.progress = info.percent;
})
// 封面上传完成时
uploader.on('cover_upload', function(info) {
uploaderInfo.isCoverUploadSuccess = true;
})
// 封面上传进度
uploader.on('cover_progress', function(info) {
uploaderInfo.coverProgress = info.percent;
})

uploader.done().then(function (doneResult) {
// deal with doneResult
})
```

`xxx_upload` 与 `xxx_progress` 的回调值请参见 分块上传/复制任务操作。

取消上传

SDK 支持取消正在上传的视频或封面：

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})

uploader.cancel()
```

断点续传

SDK 支持自动断点续传功能，无需做额外操作。当上传意外终止时（如浏览器关闭、网络中断等），您再次上传该文件，可以从中断处继续上传，减少重复上传时间。

接口描述

TcVod

参数名称	必填	类型	参数描述
getSignature	是	Function	获取上传签名的函数。

参数名称	必填	类型	参数描述
appId	否	number	填入后，内置的统计上报会自动带上。
reportId	否	number	填入后，内置的统计上报会自动带上。

TcVod.upload

参数名称	必填	类型	参数描述
mediaFile	否	File	媒体文件（视频或音频或图片）。
coverFile	否	File	封面文件。
mediaName	否	string	覆盖媒体文件元信息中的文件名。
fileId	否	string	当修改封面时传入。
reportId	否	number	填入后，内置的统计上报会自动带上。会覆盖构造函数中的设置。
fileParallelLimit	否	number	同一个实例下上传的文件并发数，默认值3
chunkParallelLimit	否	number	同一个上传文件的分块并发数，默认值6
chunkRetryTimes	否	number	分块上传时，出错重试次数，默认值2（加第一次，请求共3次）
chunkSize	否	number	分块上传时，每片的字节数大小，默认值8388608（8MB）
progressInterval	否	number	上传进度的回调方法 onProgress 的回调频率，单位 ms ，默认值 1000

事件

事件名称	必填	事件描述
media_upload	否	媒体文件上传成功时。
cover_upload	否	封面上传成功时。
media_progress	否	媒体文件上传进度。
cover_progress	否	封面文件上传进度。

常见问题

1. **File 对象怎么获取？**

   使用 `input` 标签，`type` 为 `file` 类型，即可拿到 `File` 对象。

2. **上传的文件是否有大小限制?**

   最大支持60GB。

3. **SDK 支持的浏览器版本有哪些？**

   Chrome、Firefox 等支持 HTML5 的主流浏览器，IE 方面支持的最低版本是 IE10。

4. **如何实现类似暂停上传或恢复上传的功能？**

   SDK 底层已经自动实现了断点续传的功能，因此暂停的本质即是调用 `uploader.cancel()` 这个方法。同理，暂停后的恢复上传也是调用初始的 `tcVod.upload` 方法，区别在于恢复上传时调用该方法的参数，应该是之前缓存起来的参数（例如可以在启动上传时全局变量存储一份参数，上传完成后再清掉）。

5. **Web 端上传 SDK 是否支持使用 https: 域名上传？**

   可以支持。Web 端默认判断当前页面的域名是 http: 时，使用 http: 域名上传。若判断域名非 http: 时，则使用 https: 域名上传。

Android 上传 SDK

最近更新时间：2024-11-15 10:52:16

对于在 Android 平台上传视频的场景，云点播提供了 Android 上传 SDK。上传流程请参见 客户端上传指引。

SDK 名称	云点播 Android 上传 SDK
版本号	V1.1.23.0
SDK 介绍	为 App 的最终用户提供本地视频上传到云点播平台的场景
开发者	腾讯云计算（北京）有限责任公司
下载 SDK	1. 单击下载 Android 上传 Demo 及源码，将下载好的压缩包解压，可以看到 Demo 目录。 2. 上传源码在 `Demo/app/src/main/java/com/tencent/ugcupload/demo/videoupload` 目录下。

集成上传库和源码

1. 拷贝上传源码目录 `Demo/app/src/main/java/com/tencent/ugcupload/demo/videoupload` 到您的工程目录中，需要手动修改一下 package 名。
2. 参考 `Demo/app/build.gradle` 在您的工程中添加依赖：

```
implementation 'com.qcloud.cos:cos-android-nobeacon:5.9.25'
implementation 'com.qcloud.cos:quic:1.5.43'
```

说明：

您也可以参见 手动集成 文档集成对应版本的依赖库。

3. 使用视频上传需要网络、存储等相关访问权限，可在 `AndroidManifest.xml` 中增加如下权限声明：

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

4. 视频上传需要根据网络变化来刷新上传 IP，可按照业务需要动态注册广播，示例如下。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // your code......
```

```
    registerNetReceiver();
}

private void registerNetReceiver() {
  if (null == mNetWorkStateReceiver) {
      mNetWorkStateReceiver = new TVCNetWorkStateReceiver();
      IntentFilter intentFilter = new IntentFilter(ConnectivityManager.CONNECTIVITY
      registerReceiver(mNetWorkStateReceiver, intentFilter);
   }
}

private void unRegisterNetReceiver() {
    if (null != mNetWorkStateReceiver) {
      unregisterReceiver(mNetWorkStateReceiver);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();
    // your code......
    unRegisterNetReceiver();
}
```

简单视频上传

初始化上传对象

```
TXUGCPublish mVideoPublish = new TXUGCPublish(this.getApplicationContext(), "indepe
```

设置上传对象回调

```
mVideoPublish.setListener(new TXUGCPublishTypeDef.ITXVideoPublishListener() {
    @Override
    public void onPublishProgress(long uploadBytes, long totalBytes) {
        mProgress.setProgress((int) (100*uploadBytes/totalBytes));
    }

    @Override
    public void onPublishComplete(TXUGCPublishTypeDef.TXPublishResult result) {
        mResultMsg.setText(result.retCode + " Msg:" + (result.retCode == 0 ? result
    }
});
```

构造上传参数

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam()

param.signature = "xxx";
param.videoPath = "xxx";
```

`signature` 计算规则请参见 客户端上传签名。

调用上传

```
int publishCode = mVideoPublish.publishVideo(param);
```

简单图片上传

初始化上传对象

```
TXUGCPublish mVideoPublish = new TXUGCPublish(this.getApplicationContext(), "indepe
```

设置上传对象回调

```
mVideoPublish.setListener(new TXUGCPublishTypeDef.ITXMediaPublishListener() {
    @Override
    public void onMediaPublishProgress(long uploadBytes, long totalBytes) {
        mProgress.setProgress((int) (100*uploadBytes/totalBytes));
    }
    @Override
    public void onMediaPublishComplete(TXUGCPublishTypeDef.TXMediaPublishResult med
        mResultMsg.setText(result.retCode + " Msg:" + (result.retCode == 0 ? result
    }
});
```

构造上传参数

```
TXUGCPublishTypeDef.TXMediaPublishParam param = new TXUGCPublishTypeDef.TXMediaPubl

param.signature = "xxx";
param.mediaPath = "xxx";
```

`signature` 计算规则请参见 客户端上传签名。

调用上传

```
int publishCode = mVideoPublish.publishMedia(param);
```

说明：

上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

如需上传至指定子应用下，请参见 子应用体系 - 客户端上传。

高级功能

携带封面

在上传参数中带上封面路径即可。

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam()
param.signature = "xxx";
param.videoPath = "xxx";
param.coverPath = "xxx";
```

`signature` 计算规则请参见 客户端上传签名。

取消和恢复上传

取消上传，调用 `TXUGCPublish` 的 `cancleublish()` 接口。

```
mVideoPublish.canclePublish();
```

恢复上传，用相同的上传参数（视频路径和封面路径不变），再调用一次 `TXUGCPublish` 的 `publishVideo` 。

断点续传

在视频上传过程中，云点播支持断点续传，即当上传意外终止时，用户再次上传该文件，可以从中断处继续上传，减少重复上传时间。断点续传的有效时间是1天，即同一个视频上传被中断，那么1天内再次上传可以直接从断点处上传，超过1天默认会重新上传完整视频。

上传参数中的 `enableResume` 为断点续传开关，默认是开启的。

预上传

在实际上传过程中，很大部分的错误是由于网络连接失败或超时导致的，为优化此类问题，我们增加了预上传优化逻辑。预上传包含：HTTPDNS 解析、获取建议上传地域、探测最优上传地域。

建议您在 App 启动的时候调用 `TXUGCPublishOptCenter.getInstance().prepareUpload(signature)` ，预上传模块会把 `<域名，IP>` 映射表和最优上传地域缓存在本地。如果此前动态注册了网络广播，订阅到网络切换时，清空缓存并自动刷新。

`signature` 计算规则请参见 客户端上传签名。

开启 https 上传

将上传参数中 TXPublishParam 中的 enableHTTPS 置为 true 即可，默认 false。

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam()
param.enableHttps = true;
```

关闭日志

关闭日志需要通过 TXUGCPublish 的 `setIsDebug` 方法进行操作，默认开启。开启的情况下，会打印 logcat 日志，同时也会将 log 保存到 app 私有目录下。

```
// false 关闭日志
mTXUGCPublish.setIsDebug(false);
```

视频上传接口描述

初始化上传对象：`TXUGCPublish`

参数名称	参数描述	类型	必填
context	application 上下文。	Context	是
customKey	用于区分不同的用户，建议使用 App 的账号 ID，方便后续定位问题。	String	否

设置点播 appId：`TXUGCPublish.setAppId`

参数名称	参数描述	类型	必填
appId	点播 appId。	int	是

上传视频：`TXUGCPublish.publishVideo`

参数名称	参数描述	类型	必填
param	上传参数。	TXUGCPublishTypeDef.TXPublishParam	是

上传参数：`TXUGCPublishTypeDef.TXPublishParam`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	String	是

videoPath	本地视频文件路径。	String	是
coverPath	本地封面文件路径，默认不带封面文件。	String	否
enableResume	是否启动断点续传，默认开启。	boolean	否
enableHttps	是否启动 HTTPS，默认关闭。	boolean	否
fileName	上传到腾讯云的视频文件名称，不填默认用本地文件名。	String	否
enablePreparePublish	是否开启预上传机制，默认开启。预上传机制可以大幅提升文件的上传质量。	boolean	否
sliceSize	分片大小，支持最小为1M，最大10M，默认为上传文件大小除以10。	long	否
concurrentCount	分片上传最大并发数量，默认为4个。	int	否
trafficLimit	限速值设置范围为819200 ~ 838860800，即100KB/s ~ 100MB/s，如果超出该范围会返回400错误。不建议将该值设置太小，防止超时。-1 表示不限速。	long	否
uploadResumeController	续点控制器，可自行实现续点键值的计算和保存，默认使用 md5 计算文件键值。	IUploadResumeController	否

设置上传回调：`TXUGCPublish.setListener`

参数名称	参数描述	类型	必填
listener	上传进度和结果回调监听。	TXUGCPublishTypeDef.ITXVideoPublishListener	是

进度回调：`TXUGCPublishTypeDef.ITXVideoPublishListener.onPublishProgress`

变量名称	变量描述	类型
uploadBytes	已上传的字节数。	long
totalBytes	总字节数。	long

结果回调：`TXUGCPublishTypeDef.ITXVideoPublishListener.onPublishComplete`

变量名称	变量描述	类型
result	上传结果。	TXUGCPublishTypeDef.TXPublishResult

上传结果：`TXUGCPublishTypeDef.TXPublishResult`

成员变量名称	变量说明	类型
retCode	结果码。	int
descMsg	上传失败的错误描述。	String
videoId	点播视频文件 ID。	String
videoURL	视频存储地址。	String
coverURL	封面存储地址。	String

预上传：`TXUGCPublishOptCenter.prepareUpload`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	String	是

图片上传接口描述

初始化上传对象：`TXUGCPublish`

参数名称	参数描述	类型	必填
context	application 上下文。	Context	是
customKey	用于区分不同的用户，建议使用 App 的账号 ID，方便后续定位问题。	String	否

设置点播 appId：`TXUGCPublish.setAppId`

参数名称	参数描述	类型	必填
appId	点播 appId。	int	是

上传图片：`TXUGCPublish.publishMedia`

参数名称	参数描述	类型	必填
param	上传参数。	TXUGCPublishTypeDef.TXMediaPublishParam	是

上传参数：`TXUGCPublishTypeDef.TXMediaPublishParam`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	String	是
mediaPath	本地图片文件路径。	String	是
enableResume	是否启动断点续传，默认开启。	boolean	否
enableHttps	是否启动 HTTPS，默认关闭。	boolean	否
fileName	上传到腾讯云的视频文件名称，不填默认用本地文件名。	String	否
enablePreparePublish	是否开启预上传机制，默认开启。预上传机制可以大幅提升文件的上传质量。	boolean	否
sliceSize	分片大小，支持最小为1M，最大10M，默认为上传文件大小除以10。	long	否
concurrentCount	分片上传最大并发数量，默认为4个。	int	否
trafficLimit	限速值设置范围为819200 ~ 838860800，即100KB/s ~ 100MB/s，如果超出该范围会返回400错误。不建议将该值设置太小，防止超时。-1 表示不限速。	long	否
uploadResumeController	续点控制器，可自行实现续点键值的计算和保存，默认使用md5计算文件键值。	IUploadResumeController	否

设置上传回调：`TXUGCPublish.setListener`

参数名称	参数描述	类型	必填
listener	上传进度和结果回调订阅。	TXUGCPublishTypeDef.ITXMediaPublishListener	是

进度回调：`TXUGCPublishTypeDef.ITXMediaPublishListener.onPublishProgress`

变量名称	变量描述	类型
uploadBytes	已上传的字节数。	long
totalBytes	总字节数。	long

结果回调：`TXUGCPublishTypeDef.ITXMediaPublishListener.onPublishComplete`

变量名称	变量描述	类型
result	上传结果。	TXUGCPublishTypeDef.TXPublishResult

上传结果：`TXUGCPublishTypeDef.TXMediaPublishResult`

成员变量名称	变量说明	类型
retCode	结果码。	int
descMsg	上传失败的错误描述。	String
mediaId	点播媒体文件 ID。	String
mediaURL	媒体资源存储地址。	String

预上传：`TXUGCPublishOptCenter.prepareUpload`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	String	是

错误码

SDK 通过 `TXUGCPublishTypeDef.ITXVideoPublishListener\\ITXMediaPublishListener` 接口来订阅视频上传相关的状态。因此，可以用 `TXUGCPublishTypeDef.TXPublishResult\\TXMediaPublishResult` 中的 `retCode` 来确认视频上传的情况。

状态码	在 TVCConstants 中所对应的常量	含义
0	NO_ERROR	上传成功。
1001	ERR_UGC_REQUEST_FAILED	请求上传失败，通常是客户端签名过期或者非法，需要 App 重新申请签名。
1002	ERR_UGC_PARSE_FAILED	请求信息解析失败。
1003	ERR_UPLOAD_VIDEO_FAILED	上传视频失败。
1004	ERR_UPLOAD_COVER_FAILED	上传封面失败。
1005	ERR_UGC_FINISH_REQUEST_FAILED	结束上传请求失败。

1006	ERR_UGC_FINISH_RESPONSE_FAILED	结束上传响应错误。
1007	ERR_CLIENT_BUSY	客户端正忙（对象无法处理更多请求）。
1008	ERR_FILE_NOEXIT	上传文件不存在。
1009	ERR_UGC_PUBLISHING	视频正在上传中。
1010	ERR_UGC_INVALID_PARAM	上传参数为空。
1012	ERR_UGC_INVALID_SIGNATURE	视频上传 signature 为空。
1013	ERR_UGC_INVALID_VIDOPATH	视频文件的路径为空。
1014	ERR_UGC_INVALID_VIDEO_FILE	当前路径下视频文件不存在。
1015	ERR_UGC_FILE_NAME	视频上传文件名太长（超过40）或含有特殊字符。
1016	ERR_UGC_INVALID_COVER_PATH	视频文件封面路径不对，文件不存在。
1017	ERR_USER_CANCEL	用户取消上传。
1020	ERR_UPLOAD_SIGN_EXPIRED	签名过期。

iOS 上传 SDK

最近更新时间：2024-11-15 10:57:32

对于在 iOS 平台上传视频的场景，云点播提供了 iOS 上传 SDK 。上传流程请参见 客户端上传指引。

SDK 名称	云点播 iOS 上传 SDK
版本号	V1.1.23.0
SDK 介绍	为 App 的最终用户提供本地视频上传到云点播平台的场景
开发者	腾讯云计算（北京）有限责任公司
下载 SDK	1. 单击下载 iOS 上传 Demo 及源码，将下载好的压缩包解压，可以看到 Demo 目录。 2. 上传源码在 `Demo/app/src/main/java/com/tencent/ugcupload/demo/videoupload` 目录下。

集成上传库和源码

1. 拷贝上传源码目录 `TXUGCUploadDemo/upload` 到您的工程中。
2. 在您的 Podfile 中添加如下依赖：

```
pod 'QCloudQuic','6.3.7'
pod 'QCloudCOSXML/Slim','6.4.4'
// 根据您的工程已经有该依赖可以不用额外添加
pod 'AFNetworking','4.0.1'
```

3. 在 Build Settings 中设置 Other Linker Flags，加入参数 `-ObjC` 。

简单视频上传

初始化上传对象

```
TXUGCPublish *_videoPublish = [[TXUGCPublish alloc] initWithUserID:@"upload_video_u
```

设置上传对象回调

```
_videoPublish.delegate = self;
```

```
#pragma mark - TXVideoPublishListener

- (void)onPublishProgress:(NSInteger)uploadBytes totalBytes:(NSInteger)totalBytes {
    self.progressView.progress = (float)uploadBytes/totalBytes;
    NSLog(@"onPublishProgress [%ld/%ld]", uploadBytes, totalBytes);
}

- (void)onPublishComplete:(TXPublishResult*)result {
    NSString *string = [NSString stringWithFormat:@"上传完成，错误码[%d]，信息[%@]", re
    [self showErrorMessage:string];
    NSLog(@"onPublishComplete [%d/%@]", result.retCode, result.retCode == 0? result
}
```

构造上传参数

```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];

publishParam.signature  = @"由您业务后台产生的签名";
publishParam.videoPath  = @"视频文件路径";
```

`signature` 计算规则请参见 客户端上传签名。

调用上传

```
[_videoPublish publishVideo:publishParam];
```

说明：

上传方法根据用户文件的长度，自动选择普通上传以及分片上传，用户不用关心分片上传的每个步骤，即可实现分片上传。

如需上传至指定应用下，请参见 应用体系 - 客户端上传。

高级功能

携带封面

在上传参数中带上封面图片即可。

```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];
publishParam.signature  = @"由您业务后台产生的签名";
publishParam.coverPath = @"封面图片文件路径";
publishParam.videoPath  = @"视频文件路径";
```

取消和恢复上传

取消上传，调用 `canclePublish` 接口。

```
[_videoPublish canclePublish];
```

恢复上传，用相同的上传参数（视频路径和封面路径不变），再调用一次 `TXUGCPublish` 的 `publishVideo` 。

断点续传

在视频上传过程中，云点播支持断点续传，即当上传意外终止时，用户再次上传该文件，可以从中断处继续上传，减少重复上传时间。断点续传的有效时间是1天，即同一个视频上传被中断，那么1天内再次上传可以直接从断点处上传，超过1天默认会重新上传完整视频。
上传参数中的 `enableResume` 为断点续传开关，默认是开启的。

开启 https 上传

将上传参数中 TXPublishParam 中的 enableHTTPS 置为 true 即可，默认 false。

```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];
publishParam.enableHTTPS  = true;
```

关闭日志

关闭日志需要通过 TXUGCPublish 的 `setIsDebug` 方法进行操作，默认开启。开启的情况下，会打印 logcat 日志，同时也会将 log 保存到 app 私有目录下。

```
// NO 关闭日志
[_videoPublish setIsDebug:NO];
```

图片和媒体上传

```
// 创建对象
TXUGCPublish *_imagePublish = [[TXUGCPublish alloc] initWithUserID:@"upload_image_u

// 设置回调
_imagePublish.mediaDelegate = self;

// 构造上传参数
TXMediaPublishParam *publishParam = [[TXMediaPublishParam alloc] init];
publishParam.signature  = @"由您业务后台产生的签名";
publishParam.mediaPath = @"图片文件路径";

// 上传图片或媒体文件
[_imagePublish publishMedia:publishParam];
```

视频上传接口描述

初始化上传对象：`TXUGCPublish::initWithUserID`

参数名称	参数描述	类型	必填
userID	用户 userID，用于区分不同的用户。	NSString	否

开始上传：`TXUGCPublish.publishVideo`

参数名称	参数描述	类型	必填
param	发布参数。	TXPublishParam	是

上传参数：`TXPublishParam`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	NSString*	是
videoPath	本地视频文件路径。	NSString*	是
coverPath	封面图片本地路径，可不设置。	NSString*	否
fileName	上传到腾讯云的视频文件名称，不填默认用本地文件名。	NSString*	否
enableResume	是否启动断点续传，默认开启。	BOOL	否
enableHttps	是否启动 HTTPS，默认关闭。	BOOL	否
enablePreparePublish	是否开启预上传机制，默认开启。预上传机制可以大幅提升文件的上传质量。	BOOL	否
sliceSize	分片大小，支持最小为1M，最大10M，默认为上传文件大小除以10。	long	否
concurrentCount	分片上传最大并发数量，默认为4个。	int	否
trafficLimit	限速值设置范围为819200 ~ 838860800，即100KB/s ~	long	否

	100MB/s，如果超出该范围会返回400错误。不建议将该值设置太小，防止超时。-1 表示不限速。		
uploadResumController	续点控制器，可自行实现续点键值的计算和保存，默认使用 md5 计算文件键值。	id<IUploadResumeController>	否

设置上传回调：`TXUGCPublish.delegate`

成员变量名称	变量描述	类型	必填
delegate	上传进度和结果回调代理。	TXVideoPublishListener	是

上传进度回调：`onPublishProgress`

变量名称	变量描述	类型
uploadBytes	已经上传的字节数。	NSInteger
totalBytes	总字节数。	NSInteger

上传结果回调：`onPublishComplete`

变量名称	变量描述	类型
result	上传结果。	TXPublishResult

上传事件回调：`onPublishEvent`

变量名称	变量描述	类型
evt	事件，用于调试打印。	NSDictionary

上传结果：`TXPublishResult`

成员变量名称	变量说明	类型
retCode	错误码。	int
descMsg	上传失败的错误描述。	NSString
videoId	点播视频文件 ID。	NSString
videoURL	视频存储地址。	NSString

coverURL	封面存储地址。	NSString

预上传：`TXUGCPublishOptCenter.prepareUpload`

参数名称	参数描述	类型	必填
signature	客户端上传签名	NSString	是

错误码

SDK 通过 `TXVideoPublishListener` 接口来订阅视频上传相关的状态。因此，可以用 `TXPublishResult` 中的 `retCode` 来确认视频发布的情况。

错误码	在 TVCCommon 中所对应的常量	含义
0	TVC_OK	上传成功
1001	TVC_ERR_UGC_REQUEST_FAILED	请求上传失败，通常是客户端签名过期或者非法，需要 App 重新申请签名
1002	TVC_ERR_UGC_PARSE_FAILED	请求信息解析失败
1003	TVC_ERR_VIDEO_UPLOAD_FAILED	上传视频失败
1004	TVC_ERR_COVER_UPLOAD_FAILED	上传封面失败
1005	TVC_ERR_UGC_FINISH_REQ_FAILED	结束上传请求失败
1006	TVC_ERR_UGC_FINISH_RSP_FAILED	结束上传响应错误
1008	TVC_ERR_FILE_NOT_EXIST	传入的文件路径上文件不存在
1009	TVC_ERR_ERR_UGC_PUBLISHING	视频正在上传中
1010	TVC_ERR_UGC_INVALID_PARAME	无效参数
1012	TVC_ERR_INVALID_SIGNATURE	短视频上传签名为空
1013	TVC_ERR_INVALID_VIDEOPATH	视频路径为空
1017	TVC_ERR_USER_CANCLE	用户调用取消上传
1020	TVC_ERR_UPLOAD_SIGN_EXPIRED	签名过期

图片和媒体上传接口描述

初始化上传对象：`TXUGCPublish::initWithUserID`

参数名称	参数描述	类型	必填
userID	用户 userID，用于区分不同的用户。	NSString	否

开始上传：`TXUGCPublish.publishMedia`

参数名称	参数描述	类型	必填
param	发布参数。	TXMediaPublishParam	是

上传参数：`TXMediaPublishParam`

参数名称	参数描述	类型	必填
signature	客户端上传签名。	NSString*	是
mediaPath	本地图片/媒体文件路径。	NSString*	是
fileName	上传到腾讯云的图片/媒体文件名称，不填默认用本地文件名。	NSString*	否
enableResume	是否启动断点续传，默认开启。	BOOL	否
enableHttps	是否启动 HTTPS，默认关闭。	BOOL	否
enablePreparePublish	是否开启预上传机制，默认开启。预上传机制可以大幅提升文件的上传质量。	BOOL	否
sliceSize	分片大小，支持最小为1M，最大10M，默认为上传文件大小除以10。	long	否
concurrentCount	分片上传最大并发数量，默认为4个。	int	否
trafficLimit	限速值设置范围为819200 ~ 838860800，即100KB/s ~ 100MB/s，如果超出该范围会返回400错误。不建议将该值设置太小，防止超时。-1 表示不限速。	long	否
uploadResumController	续点控制器，可自行实现续点键值的计算和保存，默认使用 md5 计算文件键值。	id<IUploadResumeController>	否

设置上传回调：`TXUGCPublish.TXMediaPublishListener`

成员变量名称	变量描述	类型	必填
mediaDelegate	上传进度和结果回调代理。	TXMediaPublishListener	是

上传进度回调：`onMediaPublishProgress`

变量名称	变量描述	类型
uploadBytes	已上传的字节数。	NSInteger
totalBytes	总字节数。	NSInteger

上传结果回调：`onMediaPublishComplete`

变量名称	变量描述	类型
result	上传结果。	TXMediaPublishResult

上传事件回调：`onMediaPublishEvent`

变量名称	变量描述	类型
evt	事件，用于调试打印。	NSDictionary

上传结果：`TXMediaPublishResult`

成员变量名称	变量说明	类型
retCode	错误码。	int
descMsg	上传失败的错误描述。	NSString
mediaId	图片/媒体文件 ID。	NSString
mediaURL	图片/媒体存储地址。	NSString

预上传：`TXUGCPublishOptCenter.prepareUpload`

参数名称	参数描述	类型	必填
signature	客户端上传签名	NSString	是

错误码

SDK 通过 `TXMediaPublishListener` 接口来订阅图片/媒体上传相关的状态。因此，可以用 `TXMediaPublishResult` 中的 `retCode` 来确认图片/媒体发布的情况。

错误码	在 TVCCommon 中所对应的常量	含义
0	TVC_OK	上传成功
1001	TVC_ERR_UGC_REQUEST_FAILED	请求上传失败，通常是客户端签名过期或者非法，需要 App 重新申请签名。
1002	TVC_ERR_UGC_PARSE_FAILED	请求信息解析失败。
1003	TVC_ERR_VIDEO_UPLOAD_FAILED	上传视频失败。
1004	TVC_ERR_COVER_UPLOAD_FAILED	上传封面失败。
1005	TVC_ERR_UGC_FINISH_REQ_FAILED	结束上传请求失败。
1006	TVC_ERR_UGC_FINISH_RSP_FAILED	结束上传响应错误。
1008	TVC_ERR_FILE_NOT_EXIST	传入的文件路径上文件不存在。
1009	TVC_ERR_ERR_UGC_PUBLISHING	视频正在上传中。
1010	TVC_ERR_UGC_INVALID_PARAME	无效参数。
1012	TVC_ERR_INVALID_SIGNATURE	短视频上传签名为空。
1013	TVC_ERR_INVALID_VIDEOPATH	视频路径为空。
1017	TVC_ERR_USER_CANCLE	用户调用取消上传。
1020	TVC_ERR_UPLOAD_SIGN_EXPIRED	签名过期。

Flutter 上传 SDK

最近更新时间：2024-11-15 10:56:54

对于在 Flutter 平台上传视频的场景，云点播提供了 Flutter 上传 SDK 。上传流程请参见 客户端上传指引。

SDK 名称	云点播 Flutter 上传 SDK
版本号	V1.0.0
SDK 介绍	为 App 的最终用户提供本地视频上传到云点播平台的场景
开发者	腾讯云计算（北京）有限责任公司
下载 SDK	1. 单击下载 Flutter 上传 SDK 及源码，将下载好的压缩包解压，可以看到 vod_upload 目录。 2. 上传源码在 `vod_upload/lib` 目录下。

环境准备

Flutter：
Flutter 2.5.0及更高版本。
Dart 2.19.2及更高版本但低于3.0版本。
Android：
Android Studio 3.5及更高版本。
Android 4.1及更高版本。
iOS：
Xcode 11.0及更高版本。
iOS 9.0及更高版本。
请确保您的项目已设置有效的开发者签名。

快速集成

引入依赖

1. 将 SDK 源码复制到项目的目录下。
2. 在 `pubspec.yaml` 中引入 `SDK` 。

```
vod_upload_flutter:
  path: ./vod_upload
```

3. 项目根目录下运行 `flutter pub get` 命令刷新依赖。

注意：

1. 最好在 `项目根目录` 、 `SDK目录` 、 `SDK Example目录` 下分别运行 `flutter pub get` 命令，不然有可能报错。
2. `SDK Example目录` 为 `SDK` 的测试项目，如无需要可以删除。

添加原生配置

Android

在 `AndroidManifest.xml` 中增加如下配置：

```
<!--网络权限-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

iOS

在 `iOS` 的 `Info.plist` 中增加如下配置：

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

注意：

如需运行 `SDK` 中提供的 `Demo` ，还应声明相册使用权限。

使用

1. 引入文件。

```
import 'package:vod_upload_flutter/txugc_publish.dart';
```

2. 创建对象。

```
var uploader = TXUGCPublish(
    id: "",
);
```

注意：

`id` 可以为任意字符串，只要**保证不重复**即可，主要目的是将 Flutter 对象与原生层对象进行映射。

相关 API

上传视频

```
uploader.publishVideo(TXPublishParam(
      signature: "",
      videoPath: "",
      fileName: "",
));
```

取消上传视频

```
uploader.cancelUploadVideo();
```

恢复上传视频

```
uploader.resumeUploadVideo(TXPublishParam(
      signature: "",
      videoPath: "",
      fileName: "",
));
```

上传媒体文件

```
uploader.publishMedia(TXMediaPublishParam(
      signature: "",
      mediaPath: "",
      fileName: "",
));
```

取消上传媒体文件

```
uploader.cancelUploadMedia();
```

恢复上传媒体文件

```
uploader.resumeUploadMedia(TXMediaPublishParam(
      signature: "",
      mediaPath: "",
      fileName: "",
));
```

预上传

```
TXUGCPublish.prepareUpload(signature, callback);
```

注意：

预上传为**静态方法。**

获取上传信息

```
// android端只能在上传过程中获取信息，iOS端全程都可以获取信息
uploader.getStatusInfo();
```

上报 AppId

```
uploader.setAppId(appId);
```

设置视频上传回调

```
uploader.setVideoListener(listener);
```

设置媒体上传回调

```
uploader.setMediaListener(listener);
```

回调接口及参数解释

视频上传参数

`TXPublishParam`

字段名	类型	是否必填	解释	默认值
signature	string	是	签名	null
videoPath	string	是	视频路径	null
fileName	string	是	文件名	null
enableResume	boolean	否	是否启用续点	true
enableHttps	boolean	否	是否启用 https	false

coverPath	string	否	封面图	null
enablePreparePublish	boolean	否	是否启用预上传(关闭后可手动预上传)	true
sliceSize	integer	否	分片大小。 支持最小为1M，最大10M，默认0。代表上传文件大小除以10。	0
concurrentCount	integer	否	分片上传并发数量(若 ≤ 0，则取 SDK 内部默认值2)。	-1

媒体上传参数

`TXMediaPublishParam`

字段名	类型	是否必填	解释	默认值
signature	string	是	签名	null
mediaPath	string	是	视频路径	null
fileName	string	是	文件名	null
enableResume	boolean	否	是否启用续点	true
enableHttps	boolean	否	是否启用 https	false
enablePreparePublish	boolean	否	是否启用预上传(关闭后可手动预上传)	true
sliceSize	integer	否	分片大小。 支持最小为1M，最大10M，默认0。代表上传文件大小除以10。	0
concurrentCount	integer	否	分片上传并发数量(若 ≤ 0，则取 SDK 内部默认值2)。	-1

视频上传回调

`ITXVideoPublishListener`

方法名	参数	解释
onPublishProgress	void	上传进度回调

onPublishComplete	void	上传完成回调

参数解释

`onPublishProgress`

参数名	类型	解释
uploadBytes	integer	上传的字节数
totalBytes	integer	总计字节数

`onPublishComplete`

参数名	类型	解释
result	TXPublishResult	上传结果

`TXPublishResult`

字段名	类型	解释
retCode	integer	错误码
descMsg	string	错误描述信息
videoId	string	视频文件 Id
videoURL	string	视频播放地址
coverURL	string	封面存储地址

媒体文件上传回调

`ITXMediaPublishListener`

方法名	参数	解释
onMediaPublishProgress	void	上传进度回调
onMediaPublishComplete	void	上传完成回调

参数解释：

`onMediaPublishProgress`

参数名	类型	解释

uploadBytes	integer	上传的字节数
totalBytes	integer	总计字节数

`onMediaPublishComplete`

参数名	类型	解释
result	TXPublishResult	上传结果

`TXMediaPublishResult`

字段名	类型	解释
retCode	integer	错误码
descMsg	string	错误描述信息
mediaId	string	视频文件 Id
mediaURL	string	视频播放地址

预上传回调

`IPrepareUploadCallback`

方法名	返回值	解释
onLoading	void	开始预上传回调
onFinish	void	预上传完成回调

上传状态信息

`ReportInfo`

字段名	类型	解释
reqType	string	请求类型，标识是在哪个步骤。
errCode	string	错误码
cosErrCode	string	COS 上传错误码
errMsg	string	错误信息
reqTime	string	当前步骤的开始时间

reqTimeCost	string	当前步骤的耗时
fileSize	string	文件大小
fileType	string	文件类型
fileName	string	文件名
fileId	string	文件 Id
appId	string	使用 TXUGCPublish 设置进来的点播 appId
reqServerIp	string	当前正在进行步骤所访问的 ip
reportId	string	客户自定义上报 id，可通过 TXUGCPublish 构造方法传入。
reqKey	string	请求键值，一般由文件最后修改时间和本次上传开始时间组成。
vodSessionKey	string	点播服务器会话键值，从申请上传接口获得。
cosRegion	string	当前上传所访问的区域
requestId	string	当前 COS 上传的请求 id
cosVideoPath	string	当前 COS 视频上传的路径
vodErrCode	integer	信令请求错误码
useHttpDNS	integer	是否使用 httpDns 进行域名解析
useCosAcc	integer	是否开启了 COS 域名加速
tcpConnTimeCost	integer	当前步骤连接服务器耗时
recvRespTimeCost	integer	当前步骤收到服务器响应耗时

媒体加工处理
视频处理
视频即时处理

最近更新时间：2024-05-07 19:19:12

传统的离线转码技术需要对整个音视频进行解码和编码，这是一种异步处理模式，导致用户的等待时间较长。然而，即时转码技术可以立即播放，无需等待，无论视频的长度，都能实现秒级开播，为用户提供全新的播放体验。

说明：

目前即时转码还处于白名单体验阶段，如需开通请联系商务申请激活。

即时转码模板

即时转码模板包含了分辨率、码率等参数。云点播使用即时转码模板表示转码参数集合，通过模板，可以指定以下转码相关参数。

分类	参数	说明
视频编码	分辨率	支持的宽度范围：128px - 1920px 支持的高度范围：128px - 1920px
	码率	支持的视频码率范围：128kbps - 10000kbps
水印	水印图片	水印图片的 base64
	水印位置	水印的位置
	水印分辨率	水印展示的分辨率

针对常见的使用场景，云点播提供了以下预置即时转码模板。

即时转码模板名	视频分辨率	码率	水印
hls_avc_540_preset	540P	1000kbps	无
hls_avc_720_preset	720P	1800kbps	无
hls_avc_1080_preset	1080P	2500kbps	无

另外，您还可以通过 服务端 API 创建和管理自定义即时转码模板。

即时转码

1. 将视频上传到云点播后，通过控制台或者服务端 API 得到视频的 URL 地址，如下：

```
http://example.com/dir1/dir2/myVideo.mp4
```

2. 拼接即时转码参数，并将 URL 中的 templateName 替换为模板名，得到即时转码的播放 URL。

```
http://example.com/dir1/dir2/myVideo.mp4$JM!Transcode,Template={templateName}/index
```

3. 下面使用预置模板名 `hls_avc_720_preset` 为例，展示如何拼接得到即时转码的播放 URL。

```
http://example.com/dir1/dir2/myVideo.mp4$JM!Transcode,Template=hls_avc_720_preset/i
```

视频处理综述

最近更新时间：2021-10-29 11:18:21

视频处理是对原始视频进行分析或加工，并输出处理结果的过程。

类别	名称	说明
视频编辑	直播推流	• 剪辑：从视频中截取一段，生成一个新视频。 • 拼接：将多个视频拼接，生成一个新视频。
	视频编辑	对媒体裁剪、拼接、重叠、翻转等操作，实现混音、声音提取、画中画等效果。
视频转换	转码	将视频转码成指定格式和分辨率的新视频。
	截图	按照指定时间点或采样间隔，对视频截图。
	水印	视频转码的同时，打上文字或图片水印。
	转动图	将视频中的一段转成 GIF 或 WEBP 格式的动图。
	转自适应码流	将视频转成 HLS 或 Dash 格式的自适应码流。
	视频加密	使用商业级 DRM（FairPlay 和 Widevine）对视频做加密。
视频 AI	视频内容智能识别	对视频内容进行智能识别（令人反感的信息、令人不适宜的信息）。
	视频内容分析	对视频内容进行智能分析（分类、标签、封面等）
	视频内容识别	对视频内容进行智能识别。

以上是云点播提供的视频处理功能清单，除了提供转码、截图、水印等一系列基础处理能力，还提供了以下两个特色能力：

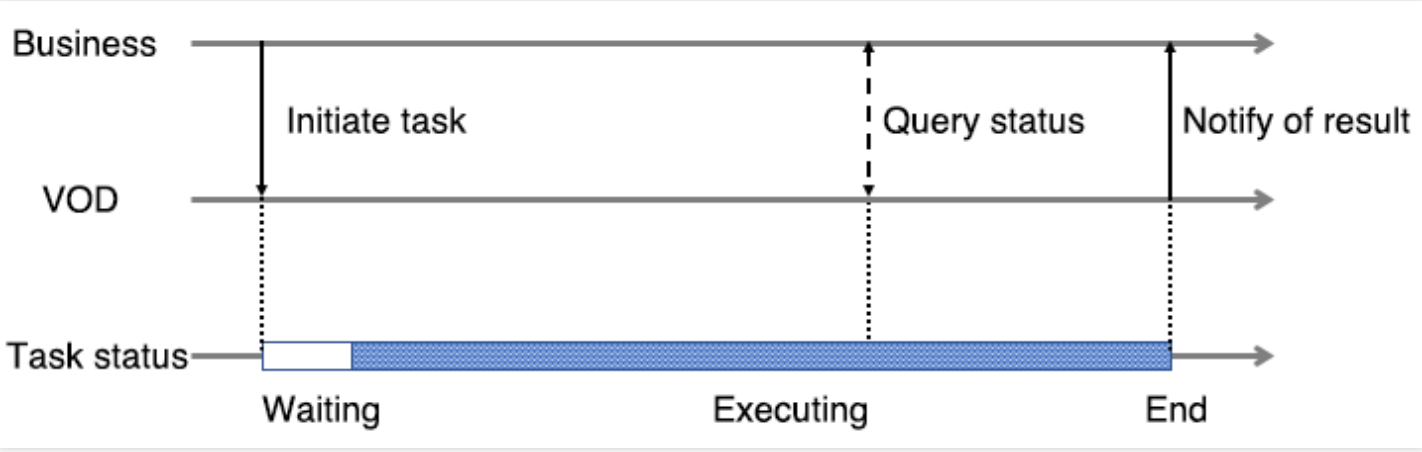
- 借助腾讯云强大的 AI，对视频进行智能识别和分析。
- 集成商业级 DRM，对视频进行高级别的加密。

发起视频处理后，处理结果不能立即输出（即无法同步地获取结果），所以视频处理都是以离线任务的方式进行的。关于如何发起视频处理任务并获取任务结果，请参见 视频处理任务体系。

视频处理任务体系

最近更新时间：2021-10-29 11:22:06

视频处理任务发起后需要一定的时间才能执行完成并输出结果（几分钟到几小时），它本质上属于一种离线任务。针对视频处理任务的特点，云点播提供了一套“同步发起 + 异步感知”的任务体系，业务方可以同步地发起任务，并异步接收结果通知，感知任务的执行结果。



- 发起任务：业务方提交一个视频处理任务后，云点播立即向业务方返回一个任务 ID，任务提交后，等待一段时间后开始执行。
- 结果通知：云点播在任务执行成功后，会立即向业务方发起结果通知，通知中包含任务 ID 和执行结果。
- 查询任务：业务方提交任务后，可以在任意时间内通过任务 ID，同步查询任务的执行状态和执行结果。

参数模板

视频处理的参数通常较复杂，例如，视频转码参数包括封装格式、编码格式、码率、分辨率以及帧率等数十个参数。为了简化视频处理任务参数，云点播将各类视频处理的参数集合固化成参数模板（如 转码模板），并用模板 ID 表示。

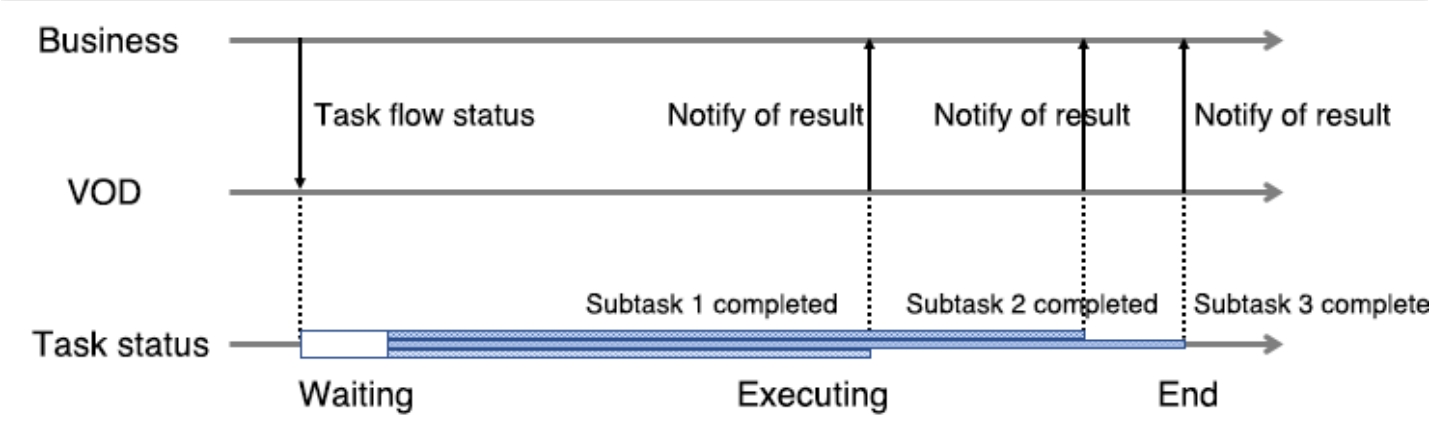
- 预置参数模板：对于常见的视频处理参数集合，云点播预置了一批参数模板，称为预置参数模板，模板清单请参见 预置参数模板列表。
- 自定义参数模板：可以通过控制台或服务端 API 自定义参数模板。

任务流

云点播中，下列视频处理操作都属于独立任务：

- 将视频转码出一路流畅的 MP4
- 将视频转码出一路标清的 MP4
- 对视频按照10s为间隔做采样截图
- 对视频做智能识别
- 对视频做智能分类

如果希望同时处理几个独立任务，则会产生多个任务 ID，并需要接收和处理多个任务的结果通知。为了简化多个任务的发起与感知，云点播提出了任务流。任务流本质上是一个包含多个子任务的“父任务”，发起一个任务流等同于发起任务流中包含的所有子任务。



如图所示，任务流中包含了3个子任务，任务流在最后一个子任务（子任务3）完成时结束。任务流的结果通知不仅在任务流结束时触发，也会在各个子任务完成时触发，因此业务方可以实时感知到任意一个子任务的执行结果。

云点播中的视频处理任务大多以任务流的方式执行，“任务流”可以被当做一种特殊的“任务”。云点播还支持 创建任务流模板，并为模板命名。发起任务流时，可以用任务流模板名字来表示要发起的任务。

任务发起

发起视频处理任务，主要有“通过服务端 API 发起”，“通过控制台发起”和“上传时指定要执行的任务”三种方式。

通过服务端 API 发起

通过服务端 API，可以直接对云点播中的视频发起任务，也可以对视频做编辑，并指定编辑生成的新视频要执行的任务。

- 视频处理
- 对指定 URL 的视频发起视频处理
- 编辑视频

通过控制台发起

通过控制台，可以对云点播中的视频发起任务，发起方式请参见 处理视频。

上传时指定要执行的任务

云点播提供了客户端上传、服务端上传和控制台上传三种视频上传方式，这几种上传方式都可以指定视频文件上传后要执行的任务。

- 客户端上传：通过 客户端上传签名 中的 `procedure` 参数，指定视频上传后要执行的任务。
- 服务端上传：通过 申请上传 中的 `procedure` 参数，指定视频上传后要执行的任务。
- 控制台上传：通过控制台上传视频，选择【上传的同时对视频进行处理操作】并指定视频上传后要执行的任务，具体操作请参考 上传视频。

结果通知

业务方发起视频处理后，需要通过以“结果通知”的方式异步感知任务的执行结果。
视频处理的结果通知主要有以下类型：

- 任务流状态变更
- 视频编辑完成

视频处理的结果通知属于云点播的“事件通知”，有“HTTP 普通回调”和“可靠回调”两种接收类型，事件通知的接收方式等信息请参见 事件通知。

任务查询

业务方除了以结果通知的方式感知任务的执行结果，还可以通过任务 ID 定时轮询任务的执行状态，即任务查询。云点播目前仅提供 获取任务列表 和 查询任务详情 两种服务端 API 查询任务的执行状态和执行结果。

预置参数模板列表

最近更新时间：2024-11-04 10:39:07

云点播支持使用参数模板代替复杂的参数集合，发起视频处理。针对不同视频处理的场合，云点播预置了一批参数模板。

视频转换类

视频转换类的预置参数模板包含了以下几种类型：

预置转码模板

预置转封装模板

预置转动图模板

预置指定时间点截图模板

预置采样截图模板

预置雪碧图模板

预置转自适应码流模板

预置转码模板

转码视频格式

规格等级	模板 ID	封装格式（Format）	视频参数				音频参数
			分辨率（Resolution）	码率（Bitrate）	帧率（FPS）	编码（Codec）	码率（Bitrate
流畅（FLU）	100010	MP4	按比例缩放 × 360	400kbps	25	H.264	64 kbps
	100210	HLS					
标清（SD）	100020	MP4	按比例缩放 × 540	1000kbps			
	100220	HLS					
高清（HD）	100030	MP4	按比例缩放 × 720	1800kbps			128kbps
	100230	HLS					
全高清（FHD）	100040	MP4	按比例缩放 × 1080	2500kbps			
	100240	HLS					
2K	100070	MP4	按比例缩放 ×	3000kbps			160kbps

	100270	HLS	1440				
4K	100080	MP4	按比例缩放 × 2160	6000kbps			
	100280	HLS					

转码音频格式

模板 ID	封装格式（Format）	音频码率（Bitrate）	编码（Codec）	声道数（SoundSystem）	采样频率（SampleRate）
1100	M4A	24kbps	AAC	双通道（Stereo）	44100Hz
1110		48kbps			
1120		96kbps			
1130		192kbps			
1140		256kbps			
1010	MP3	128kbps	MP3		
1020		320kbps			

预置极速高清模板

规格等级	模板 ID	封装格式（Format）	视频参数				音频参
			分辨率（Resolution）	最大码率（Bitrate）	帧率（FPS）	编码（Codec）	码率（Bitra
同源（SAME）	100800	MP4	同源	无限制	25	H.264	同源
流畅（FLU）	100810		按比例缩放 × 360				64 kbp
标清（SD）	100820		按比例缩放 × 540				
高清（HD）	100830		按比例缩放 × 720				128kb
全高清（FHD）	100840		按比例缩放 × 1080				

预置转封装模板

模板 ID	转封装目标格式（Format）
875	MP4
876	HLS

预置转动图模板

模板 ID	图片格式（Format）	分辨率（Resolution）	帧率（FPS）
20000	GIF	同源	2
20001	WEBP	同源	2

预置指定时间点截图模板

模板 ID	输出格式（Format）	宽度（Width）	高度（Height）	填充方式（FillType）
10	JPG	同源	同源	拉伸

预置采样截图模板

模板 ID	输出格式（Format）	宽度（Width）	高度（Height）	采样方式（SampleType）	截图间隔（Interval）	填充方式（FillType）
10	JPG	同源	同源	按百分比	10%	拉伸

预置雪碧图模板

模板 ID	输出格式（Format）	小图宽度（Width）	小图高度（Height）	小图行数（Rows）	小图列数（Columns）	采样方式（SampleType）	截图间（Inter
10	JPG	142	80	10	10	按时间间隔	10秒

预置转自适应码流模板

模板信息

模板 ID	打包类型（PackageType）	加密类型（EncryptionType）	子流信息（SubstreamInfo）	过滤“低分辨率转高分辨率”（DisableHigherResolution）

10	HLS	不加密	包含从“流畅”到“4K”共6个规格的子流	是
12	HLS	SimpleAES	包含从“流畅”到“4K”共6个规格的子流	是
20	MPEG-DASH	不加密	包含从“流畅”到“4K”共6个规格的子流	否

子流信息

子流规格	视频参数				音频参数		
	分辨率（Resolution）	码率（Bitrate）	帧率（FPS）	编码（Codec）	码率（Bitrate）	采样频率（SampleRate）	音频声（Sou
流畅	按比例缩放 x 240	256kbps	24	H.264	48kbps	44100Hz	双声道
标清	按比例缩放 x 480	512kbps	24	H.264	48kbps	44100Hz	双声道
高清	按比例缩放 x 720	1024kbps	24	H.264	48kbps	44100Hz	双声道
全高清	按比例缩放 x 1080	2500kbps	24	H.264	48kbps	44100Hz	双声道
2K	按比例缩放 x 1440	3072kbps	24	H.264	48kbps	44100Hz	双声道
4K	按比例缩放 x 2160	6144kbps	24	H.264	48kbps	44100Hz	双声道

媒体 AI 类

媒体 AI 类的预置参数模板包含了以下几种类型：

预置音视频审核模板

预置视频内容分析模板

预置视频内容识别模板

预置音视频审核模板

模板 ID	色情（Porn）	暴恐（Terror）	娇喘（Moan）
10	是	是	是

预置音视频内容分析模板

模板 ID	智能分类（Classification）	智能标签（Tag）	智能封面（Cover）	智能按帧标签（FrameTag）
10	是	是	是	否
20	是	是	是	是

预置音视频内容识别模板

模板 ID	人脸识别（Face）	语音全文识别（AsrFullText）	语音翻译识别（AsrTranslate）	语音关键词识别（AsrWords）	文本全文识别（OcrFullText）	文本关键词识别（OcrWor
10	是（使用默认人物库）	否	否	否	否	否
111	否	是（音频源语言中文，生成 vtt 字幕）	否	否	否	否
112	否	是（音频源语言英文，生成 vtt 字幕）	否	否	否	否
113	否	是（音频源语言日文，生成 vtt 字幕）	否	否	否	否
121	否	否	是（音频源语言中文，翻译目标语言英文，生成 vtt 字幕）	否	否	否

122	否	否	是（音频源语言英文，翻译目标语言中文，生成 vtt 字幕）	否	否	否
123	否	否	是（音频源语言马来语，翻译目标语言英文，生成 vtt 字幕）	否	否	否
124	否	否	是（音频源语言英文，翻译目标语言马来语，生成 vtt 字幕）	否	否	否
125	否	否	是（音频源语言中文，翻译目标语言日文，生成 vtt 字幕）	否	否	否
126	否	否	是（音频源语言中文，翻译目标语言韩文，生成 vtt 字幕）	否	否	否

历史转码类

历史预置转码模板

转码视频格式

规格等级	模板 ID	封装格式（Format）	视频参数				音频参数
			分辨率（Resolution）	码率（Bitrate）	帧率（FPS）	编码（Codec）	编码（Codec）
流		MP4	320 ×				

畅（FLU）	10		按比例缩放	256kbps	24	H.264	AAC
	510	MP4	按比例缩放 × 240	250kbps	15	H.265	AAC
	210	HLS	320 × 按比例缩放	256kbps	24	H.264	AAC
	610	HLS	按比例缩放 × 240	250kbps	15	H.265	AAC
	10046	FLV	320 × 按比例缩放	256kbps	24	H.264	MP3
	710	FLV	按比例缩放 × 240	250kbps	15	H.265	AAC
标清（SD）	20	MP4	640 × 按比例缩放	512kbps	24	H.264	AAC
	520	MP4	按比例缩放 × 480	600kbps	24	H.265	AAC
	220	HLS	640 × 按比例缩放	512kbps	24	H.264	AAC
	620	HLS	按比例缩放 × 480	600kbps	24	H.265	AAC
	10047	FLV	640 × 按比例缩放	512kbps	24	H.264	MP3
	720	FLV	按比例缩放 × 480	600kbps	24	H.265	AAC
高清（HD）	30	MP4	1280 × 按比例缩放	1024kbps	24	H.264	AAC
	530	MP4	按比例缩放 × 720	800kbps	25	H.265	AAC
	230	HLS	1280 × 按比例缩放	1024kbps	24	H.264	AAC
	630	HLS	按比例缩放 × 720	800kbps	25	H.265	AAC
		FLV	1280 ×				

	10048		按比例缩放	1024kbps	24	H.264	MP3
	730	FLV	按比例缩放 × 720	800kbps	25	H.265	AAC
全高清（FHD）	40	MP4	1920 × 按比例缩放	2500kbps	24	H.264	AAC
	540	MP4	按比例缩放 × 1080	1400kbps	30	H.265	AAC
	240	HLS	1920 × 按比例缩放	2500kbps	24	H.264	AAC
	640	HLS	按比例缩放 × 1080	1400kbps	30	H.265	AAC
	10049	FLV	1920 × 按比例缩放	2500kbps	24	H.264	MP3
	740	FLV	按比例缩放 × 1080	1400kbps	30	H.265	AAC
2K	70	MP4	按比例缩放 × 1440	3072kbps	30	H.264	AAC
	570	MP4	按比例缩放 × 1440	2048kbps	30	H.265	AAC
	270	HLS	按比例缩放 × 1440	3072kbps	30	H.264	AAC
	670	HLS	按比例缩放 × 1440	2048kbps	30	H.265	AAC
	370	FLV	按比例缩放 × 1440	3072kbps	30	H.264	MP3
	770	FLV	按比例缩放 × 1440	2048kbps	30	H.265	AAC
4K	80	MP4	按比例缩放 × 2160	6144kbps	30	H.264	AAC
	580	MP4	按比例缩放 × 2160	4096kbps	30	H.265	AAC
		HLS	按比例				

	280		缩放 × 2160	6144kbps	30	H.264	AAC
	680	HLS	按比例 缩放 × 2160	4096kbps	30	H.265	AAC
	380	FLV	按比例 缩放 × 2160	6144kbps	30	H.264	MP3
	780	FLV	按比例 缩放 × 2160	4096kbps	30	H.265	AAC

以上转视频模板中未注明的参数全部相同，分别是：

分类	参数/能力项	说明
视频参数	编码档次	使用 H.264 编码时，编码档次为 High 使用 H.265 编码的，编码档次为 Main
	GOP 长度	240帧
	颜色空间	YUV420P
	码率控制方法	动态比特率编码（VBR）
音频参数	采样率	44100Hz
	码率	48kbps
	声道数	双通道（Stereo）

视频编制
视频编辑

最近更新时间：2020-12-09 15:24:43

视频编辑，是对云点播中的视频进行剪辑和拼接的过程，是一种离线任务。视频编辑的功能包括以下几种：

- **视频剪辑**：对云点播中的一个文件进行剪辑，生成一个新的视频。
- **视频拼接**：对云点播中的多个文件进行拼接，生成一个新的视频。
- **视频剪辑后拼接**：对云点播中的多个文件进行剪辑，然后再拼接，生成一个新的视频。
- **直播流转视频**：对云点播中的一个流进行处理，生成一个新的视频。
- **直播流剪辑**：对云点播中的一个流进行剪辑，生成一个新的视频。
- **直播流拼接**：对云点播中的多个流进行拼接，生成一个新的视频。
- **直播流剪辑后拼接**：对云点播中的多个流进行剪辑，然后拼接，生成一个新的视频。

> ⚠ **注意：**
>
> 当对直播流做剪辑、拼接等操作时，请确保流结束后再操作，否则生成的视频可能不完整。

编辑后生成的新视频封装格式是 MP4。发起编辑时，可以指定是否对生成的新视频执行 任务流。

任务发起

视频编辑任务，通过 服务端 API 方式发起。调用 API 的返回结果中包含任务 ID，用于关联 结果获取 时对应的任务结果。

结果获取

发起任务后，您可以通过异步等待 结果通知 或同步进行 任务查询 的方式获取编辑的执行结果。下面是发起视频编辑任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：

```
{
"EventType":"EditMediaComplete",
"EditMediaCompleteEvent":{
"TaskId":"EditMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status":"FINISH",
"ErrCode":0,
"Message":"SUCCESS",
```

```
"Input":{
"InputType":"File",
"FileInfoSet":[
{
"FileId":"24961954183381008",
"StartTimeOffset":0,
"EndTimeOffset":300
},
{
"FileId":"24961954183381009",
"StartTimeOffset":0,
"EndTimeOffset":300
},
{
"FileId":"24961954183381010",
"StartTimeOffset":0,
"EndTimeOffset":300
}
]
},
"Output":{
"FileType":"mp4",
"FileId":"24961954183923290",
"FileUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/f0.mp4"
},
"ProcedureTaskId":""
}
}
```


回调结果中，`Input.InputType` 为 `File`，表示编辑的视频是文件类型。`Input.FileInfoSet` 包含三个元素，其中 `StartTimeOffset` 为 `0`，`EndTimeOffset` 为 `300`，表示对三个视频各剪辑前5分钟的片段后再做拼接，拼接后的视频时长为15分钟。`Output.FileId` 是视频编辑后生成的新视频的 FileId，视频的播放 URL 是 `FileUrl` 中的值。

视频合成

最近更新时间：2022-12-30 16:30:24

视频合成，是对云点播中的视频进行裁剪、拼接、重叠以及翻转等复杂操作，是一种离线任务。使用视频合成，可以达到以下效果：

- **画面旋转**：对视频、图片的画面旋转一定角度，或按照某个方向翻转。
- **声音控制**：升高降低视频、音频中声音的音量，或者对视频静音。
- **画面叠加**：将视频、图片中的画面依序叠加在一起，如实现“画中画”的效果。
- **声音混合**：将视频、音频中的声音混合在一起（混音）。
- **声音提取**：将视频中的音频提取出来（不保留画面）。
- **裁剪**：对视频、音频裁剪出指定时间段。
- **拼接**：对视频、音频、图片按时间顺序前后拼接。
- **转场**：将多段视频或图片拼接时，可以在段落之间添加转场效果。
- **倍速**：将视频或音频素材做快进或者慢放处理。

合成后生成的媒体封装格式是 MP4（视频）或 MP3（音频）。

任务发起

视频合成任务，通过 服务端 API 方式发起。调用 API 的返回结果中包含任务 ID，用于关联 结果获取 时对应的任务结果。

结果获取

发起任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取视频合成的执行结果。下面是发起视频合成任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：


```
{
"EventType": "ComposeMediaComplete",
"ComposeMediaCompleteEvent": {
"TaskId": "ComposeMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status": "FINISH",
"ErrCode": 0,
"Message": "SUCCESS",
"Input": {
"Tracks": [{
"Type": "Video",
```

```
"TrackItems": [{
"Type": "Video",
"SourceMedia": "5285485487985271487",
"AudioOperations": [{
"Type": "Volume",
"VolumeParam": {
"Mute": 1
}
}]
}]
},
{
"Type": "Audio",
"TrackItems": [{
"Type": "Empty",
"EmptyItem": {
"Duration": 5
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271488",
"Duration": 15
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271489",
"SourceMediaStartTime": 2,
"Duration": 14
}
}
]
}
],
"Output": {
"FileName": "视频合成效果测试",
"Container": "mp4"
}
},
"Output": {
"FileType": "mp4",
"FileId": 5285485487985271490,
"FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4"
}
```

```
  }
  }
```

回调结果中，`Input.Tracks` 包含两个元素，`Type` 分别是 Video 和 Audio，表示合成的视频包含一个视频轨和音频轨。

- 视频轨道：来源视频 ID 是5285485487985271487，对其做了静音处理。
- 音频轨道：包含5秒的静音，以及15秒和14秒的两个配音。

`Output.FileId` 是视频合成后生成的新视频 FileId，视频播放 URL 是 `FileUrl` 中的值。

视频转换
转码

最近更新时间：2024-09-03 11:04:36

转码是将视频码流转换成另一个视频码流的过程，是一种离线任务。通过转码，可以改变原始码流的编码格式、分辨率和码率等参数，从而适应不同终端和网络环境的播放。使用转码功能可以实现：
适配更多终端：将原始视频转码成拥有更强终端适配能力的格式（如 MP4），使视频资源能够在更多设备上播放。
适配不同带宽：将视频转换成流畅、标清、高清以及超清等输出，用户可以根据当前网络环境选择合适码率的视频播放。
改善播放效率：转码可以将 MP4 位于尾部的元信息 MOOV 提前到头部，播放器无需下载完整视频即可立即播放。
为视频打水印：为视频打上水印标识视频的归属或版权，更多信息请参见 水印。
节省带宽：采用更先进的编码方式（如 H.265）转码，在不损失原始画质的情况下显著降低码率，节省播放带宽。
视频转码后，根据 结果获取 可以获得转码后视频播放的 URL。您可以使用自有播放器或第三方播放器，播放转码后的视频。

注意：

转码功能主要适用于**短视频**的场景。对于**长视频**（视频网站、在线教育等），使用 转自适应码流 将为您和您的用户带来更好的体验。

转码模板

转码的目标规格包含了编码格式、分辨率和码率等参数。云点播使用转码模板表示转码参数集合，通过转码模板，可以指定以下转码相关参数。

说明：

更多音视频转码格式，请参见 转码支持格式。

分类	参数	说明
封装	封装格式	可以转以下视频和纯音频封装格式： 视频：MP4、TS、HLS、FLV 纯音频：MP3、M4A、FLAC、OGG
	删除视频流	如果开启“删除视频流”，转码出来的视频将不包含视频流（仅保留音频流）
	删除音频流	如果开启“删除音频流”，转码出来的视频将不包含音频流（仅保留视频流）
视频编码	编码方式（Codec）	支持 H.264 和 H.265 两种编码方式

	码率（Bitrate）	支持的视频码率范围：10kbps - 35Mbps
	帧率（Frame Rate）	支持的帧率范围：1fps - 60fps，常见的有24fps、25fps和30fps
	分辨率（Resolution）	支持宽度范围：128px - 4096px 支持高度范围：128px - 4096px
	GOP 长度	支持 GOP 长度范围：1秒 - 10秒
	编码档次（Profile）	当视频编码方式为 H.264 时，支持 Baseline、Main 和 High 的编码档次 当视频编码方式为 H.265 时，仅支持 Main 编码档次
	颜色空间（Color Space）	支持 YUV420P
音频编码参数	编码方式（Codec）	支持 MP3、AAC、AC3 和 FLAC 的编码方式
	采样率（Sample Rate）	支持下列音频采样率： 34000Hz 44100Hz 48000Hz
	码率（Bitrate）	支持码率在26kbps - 256kbps，包括： 48kbps 64kbps 128kbps
	声道（Channel）	单声道 双声道 立体道

针对常见的转码规格，云点播提供了 预置转码模板。另外，您还可以通过控制台（具体操作请参见 模板设置）或调用 服务端 API 创建和管理自定义转码模板。

任务发起

发起转码任务，有“通过服务端 API 直接发起”，“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。

以下是各种方式发起转码任务的说明：

调用服务端 API ProcessMedia 发起任务：在请求中的 `MediaProcessTask.TranscodeTaskSet` 参数指定 转码模板 的模板 ID。

通过控制台对视频发起任务：在控制台 添加任务流，任务流中设置目标转码规格；在控制台使用该任务流 发起视频处理。
服务端上传时指定任务：在控制台 添加任务流，任务流中设置目标转码规格；申请上传 中的 `procedure` 参数指定为该任务流。
客户端上传时指定任务：在控制台 添加任务流，任务流中设置目标转码规格；在 客户端上传签名 中的 `procedure` 指定该任务流。
控制台上传：在控制台 添加任务流，任务流中设置目标转码规格；通过控制台上传视频，选择 上传的同时对视频进行处理操作 并指定视频上传后执行该任务流。

结果获取

发起转码任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取转码的执行结果。下面是发起转码任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：


```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784246869930",
        "FileName":"动物世界",
        "FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
        "MetaData":{
            "AudioDuration":60,
            "AudioStreamSet":[
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
            "VideoStreamSet":[
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
```

```
                "Height":480,
                "Width":640
            }
        ],
        "Width":640
    },
    "MediaProcessResultSet":[
        {
            "Type":"Transcode",
            "TranscodeTask":{
                "Status":"SUCCESS",
                "ErrCode":0,
                "Message":"",
                "Input":{
                    "Definition":220
                },
                "Output":{
                    "Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/v.f20.m3
                    "Size":63120997,
                    "Container":"mov,mp4,m4a,3gp,3g2,mj2",
                    "Height":480,
                    "Width":640,
                    "Bitrate":513402,
                    "Md5":"084d403c73930ca2f835679af1f37bd3",
                    "Duration":60,
                    "VideoStreamSet":[
                        {
                            "Bitrate":473101,
                            "Codec":"h264",
                            "Fps":24,
                            "Height":480,
                            "Width":640
                        }
                    ],
                    "AudioStreamSet":[
                        {
                            "Bitrate":48581,
                            "Codec":"aac",
                            "SamplingRate":44100
                        }
                    ],
                    "Definition":220
                }
            }
        }
    ],
    "TasksPriority":0,
```

```
        "TasksNotifyMode":""
    }
}
```

回调结果中，`ProcedureStateChangeEvent.MediaProcessResultSet` 有一个 `Type` 为 `Transcode` 类型的转码结果，`Definition` 为220。

水印

最近更新时间：2021-01-22 10:36:00

打水印是在视频转码或截图时，将特定的图片或文字附加在画面指定位置的过程，是一种离线任务。云点播支持以下类型的水印：

- 静态图片水印：PNG 格式的图片水印，可以是版权方的 LOGO、台标等，常用于表明视频的版权归属。
- 动态图片水印：APNG 格式的动态图片水印，可以实现水印图片动态变化的效果。
- 文字水印：多语言的文字形式水印，可以是用户昵称等，常用于短视频中标识视频制作者。

云点播支持为视频或截图打多个水印，并可以指定各个水印在画面中的大小和位置。

水印模板

水印的目标规格包含了水印类型、宽高和位置等参数。云点播使用水印模板表示水印参数集合，通过水印模板，可以指定以下水印相关参数。

参数	说明
水印类型（Type）	可以打图片和文字两种类型的水印： • 图片水印：静态图片和动态图片。 • 文字水印：支持多种语言的文字。
水印位置（Position）	水印在视频画面中的相对位置。
图片大小（ImageSize）	图片水印占视频画面的大小。
图片内容（ImageContent）	图片水印中图片的二进制内容。
字体大小（FontSize）	文字水印中字体的大小。
字体类型（FontType）	文字水印中文字的字体类型（如宋体）。
字体颜色（FontColor）	文字水印中文字的颜色（0xRRGGBB）。
文字透明度（FontAlpha）	文字水印中文字的透明度（0 - 100%）。

您可以通过控制台（具体操作请参见 模板设置）或调用 服务端 API 创建和管理自定义水印模板。

任务发起

打水印需要发起转码任务，有“通过服务端 API 直接发起”，“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。

以下是各种方式发起带水印的转码任务说明：

- 调用服务端 API ProcessMedia 发起任务：在请求中的 `MediaProcessTask.TranscodeTaskSet` 参数指定 水印模板 的模板 ID。
- 通过控制台对视频发起任务：在控制台 添加任务流，任务流中设置转码时所打水印的目标规格；在控制台使用该任务流 发起视频处理。
- 服务端上传时指定任务：在控制台 添加任务流，任务流中设置转码时所打水印的目标规格；申请上传 中的 `procedure` 参数指定为该任务流。
- 客户端上传时指定任务：在控制台 添加任务流，任务流中设置转码时所打水印的目标规格；在 客户端上传签名 中的 `procedure` 指定该任务流。
- 控制台上传：在控制台 添加任务流，任务流中设置转码时所打水印的目标规格；通过控制台上传视频，选择【上传的同时对视频进行处理操作】并指定视频上传后执行该任务流。

结果获取

发起打水印的转码任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取截图任务的执行结果。下面是发起带有水印的转码任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：


```
{
"EventType":"ProcedureStateChanged",
"ProcedureStateChangeEvent":{
"TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
"Status":"FINISH",
"FileId":"5285890784246869930",
"FileName":"动物世界",
"FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
"MetaData":{
"AudioDuration":60,
"AudioStreamSet":[
{
"Bitrate":383854,
"Codec":"aac",
"SamplingRate":48000
}
],
"Bitrate":1021028,
"Container":"mov,mp4,m4a,3gp,3g2,mj2",
"Duration":60,
"Height":480,
```

```
"Rotate":0,
"Size":7700180,
"VideoDuration":60,
"VideoStreamSet":[
{
"Bitrate":637174,
"Codec":"h264",
"Fps":23,
"Height":480,
"Width":640
}
],
"Width":640
},
"MediaProcessResultSet":[
{
"Type":"Transcode",
"TranscodeTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
"Definition":220,
"WatermarkSet": [
{
"Definition": 23120
}
]
},
"Output":{
"Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/v.f20.m3u8",
"Size":63120997,
"Container":"mov,mp4,m4a,3gp,3g2,mj2",
"Height":1086,
"Width":1920,
"Bitrate":513402,
"Md5":"084d403c73930ca2f835679af1f37bd3",
"Duration":60,
"VideoStreamSet":[
{
"Bitrate":473101,
"Codec":"h264",
"Fps":24,
"Height":480,
"Width":640
}
],
```

```
"AudioStreamSet":[
{
"Bitrate":48581,
"Codec":"aac",
"SamplingRate":44100
}
],
"Definition":220
}
}
}
],
"TasksPriority":0,
"TasksNotifyMode":""
}
}
```

回调结果中，`ProcedureStateChangeEvent.MediaProcessResultSet` 有一个 `Type` 为 `Transcode` 类型的转码结果：转码的规格 `Definition` 为220，转码的同时打了一个水印，水印的规格 `Definition` 为23120。

截图

最近更新时间：2023-03-07 11:20:50

截图是截取视频特定位置的图像并生成图片的过程，是一种离线任务。云点播提供以下类型的截图：

- 指定时间点截图：指定一组时间点，截取视频在这些时间点的图像。
- 采样截图：按相同的时间间隔对视频截取多张图。
- 截取一张图作封面：指定一个时间点截图，将其 URL 作为媒资系统中该视频的封面。
- 截雪碧图：按相同的时间间隔对视频截取多张小图，然后组装成若干大图（即雪碧图）。

使用截图功能，可以满足如下应用场景：

- 为视频生成封面：使用视频的截图作为视频的封面。
- 缩略图：雪碧图是嵌套了多个小图（即缩略图）的大图，常用来表示一个视频的概要。
- 播放预览：雪碧图配合 VTT 文件，可以用来实现播放器进度条上的预览效果。

截图模板

截图的目标规格，包含截图文件格式、截图宽高等参数。云点播使用截图模板表示截图参数集合，通过截图模板，可以指定以下截图相关参数。

时间点截图模板

时间点截图模板，用于“指定时间点截图”和“截取一张图作封面”两种任务。

参数	说明
格式（Format）	截图文件的输出格式，目前仅支持 JPG。
宽度（Width）	截图宽度，范围是128px - 4096px。
高度（Height）	截图高度，范围是128px - 4096px。
填充方式（FillType）	当截图的宽高比与原始视频的宽高比不一致时，对截图的处理方式，即为“填充”。一般有以下几种填充方式： • 拉伸：对图片进行拉伸，填满整个图片，可能导致图片被“压扁”或者“拉长”。 • 留黑：保持图片宽高比不变，边缘剩余部分使用黑色填充。 • 留白：保持图片宽高比不变，边缘剩余部分使用白色填充。 • 高斯模糊：保持图片宽高比不变，边缘剩余部分使用高斯模糊化后填充。

针对常见的规格，云点播提供 预置时间点截图模板。另外，您还可以通过控制台创建和管理自定义截图模板，具体操作请参见 模板设置。

采样截图模板

采样截图模板，用于“采样截图”任务。

参数	说明
格式（Format）	截图文件的输出格式，目前仅支持 JPG。
宽度（Width）	截图宽度，范围是128px - 4096px。
高度（Height）	截图高度，范围是128px - 4096px。
采样方式（SampleType）	采样方式分为两种： • 按百分比采样：例如按照5%为间隔采样，生成截图张数将为20张。 • 按时间间隔采样：例如按照10s为间隔采样，截图张数取决于视频的时长。
采样间隔（Interval）	采样的间隔长度： • 如果按百分比采样，间隔是百分比。 • 如果按时间间隔采样，间隔是多少秒。
填充方式（FillType）	当截图的宽高比与原始视频的宽高比不一致时，对截图的处理方式，即为“填充”。一般有以下几种填充方式： • 拉伸：对图片进行拉伸，填满整个图片，可能导致图片被“压扁”或者“拉长”。 • 留黑：保持图片宽高比不变，边缘剩余部分使用黑色填充。 • 留白：保持图片宽高比不变，边缘剩余部分使用白色填充。 • 高斯模糊：保持图片宽高比不变，边缘剩余部分使用高斯模糊化后填充。

针对常见的规格，云点播提供了 预置采样截图模板。另外，您还可以通过控制台创建和管理自定义截图模板，具体操作请参见 模板设置。

雪碧图模板

雪碧图模板，用于“截雪碧图”任务。

参数	说明
格式（Format）	雪碧图文件的输出格式，目前仅支持 JPG。
小图宽度（Width）	雪碧图中小图的宽度。
小图高度（Height）	雪碧图中小图的高度。
小图行数（Rows）	一张大图中有多少行小图。

参数	说明
小图列数（Columns）	一张大图中有多少列小图。
采样方式（SampleType）	小图采样方式，目前仅支持按照时间间隔采样。
采样间隔（Interval）	小图采样的间隔，即隔多久采样一张小图。

注意：

- Width × Columns 需要在128px - 4096px之间（即大图宽度在128px - 4096px之间）。
- Height × Rows 需要在128px - 4096px之间（即大图高度在128px - 4096px之间）。

针对常见的规格，云点播提供了 预置雪碧图模板。另外，您还可以通过控制台创建和管理自定义截图模板，具体操作请参见 模板设置。

任务发起

发起截图任务，有“通过服务端 API 直接发起”，“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。

以下是各种方式发起截图任务的说明：

- 调用服务端 API ProcessMedia 发起任务：在请求中的 `MediaProcessTask.SnapshotByTimeOffsetTaskSet` 参数指定 截图模板 的模板 ID。
- 通过控制台对视频发起任务：在控制台 添加任务流，任务流中设置目标截图规格；在控制台使用该任务流 发起视频处理。
- 服务端上传时指定任务：在控制台 添加任务流，任务流中设置目标截图规格 申请上传 中的 `procedure` 参数指定为该任务流。
- 客户端上传时指定任务：在控制台 添加任务流，任务流中设置目标截图规格；在 客户端上传签名 中的 `procedure` 指定该任务流。
- 控制台上传：在控制台 添加任务流，任务流中设置目标截图规格；通过控制台上传视频，选择 上传的同时对视频进行处理操作 并指定视频上传后执行该任务流。

结果获取

发起截图任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取截图任务的执行结果。下面是发起截图任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：


```
{
"EventType":"ProcedureStateChanged",
"ProcedureStateChangeEvent":{
"TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
"Status":"FINISH",
"FileId":"5285890784246869930",
"FileName":"动物世界",
"FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
"MetaData":{
"AudioDuration":60,
"AudioStreamSet":[
{
"Bitrate":383854,
"Codec":"aac",
"SamplingRate":48000
}
],
"Bitrate":1021028,
"Container":"mov,mp4,m4a,3gp,3g2,mj2",
"Duration":60,
"Height":480,
"Rotate":0,
"Size":7700180,
"VideoDuration":60,
"VideoStreamSet":[
{
"Bitrate":637174,
"Codec":"h264",
"Fps":23,
"Height":480,
"Width":640
}
],
"Width":640
},
"MediaProcessResultSet":[
{
"Type":"SnapshotByTimeOffset",
"SnapshotByTimeOffsetTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
```

```
"Definition":10,
"Definition":[3, 6, 9]
},
"Output":{
"Definition":10,
"PicInfoSet": [
{
"TimeOffset":3,
"Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx1.jpg"
},
{
"TimeOffset":6,
"Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx2.jpg"
},
{
"TimeOffset":9,
"Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx3.jpg"
}
]
}
}
},
{
"Type":"SampleSnapshot",
"SampleSnapshotTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
"Definition":10
},
"Output":{
"Definition":10,
"SampleType":"Percent",
"Interval": 10,
"WaterMarkDefinition": [],
"ImageUrlSet": [
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx1.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx2.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx3.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx4.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx5.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx6.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx7.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx8.jpg",
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx9.jpg"
]
```

```
}
}
},
{
"Type":"ImageSprites",
"ImageSpriteTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
"Definition":10
},
"Output":{
"Definition":10,
"Height":80,
"Width":142,
"TotalCount":1,
"ImageUrlSet":[
"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx1.jpg"
],
"WebVttUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.vtt"
}
}
},
{
"Type":"CoverBySnapshot",
"CoverBySnapshotTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
"Definition":10,
"PositionType":"Time",
"PositionValue":0
},
"Output":{
"CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.jpg"
}
}
}
],
"TasksPriority":0,
"TasksNotifyMode":""
}
}
```

回调结果
中，`ProcedureStateChangeEvent.MediaProcessResultSet` 有 `Type` 为 `SnapshotByTimeOffset` 、`SampleSnapshot` 、`ImageSprites` 和 `CoverBySnapshot` 的结果，分别代表指定时间点截图、采样截图、截雪碧图和截取一张图作封面几种截图任务。

转动图

最近更新时间：2020-10-20 15:48:51

转动图是选取视频片段生成动图（GIF 和 WEBP 等）的过程，是一种离线任务。动图是一组连续帧的无缝循环，以较小的体积实现动画效果。

说明：

支持转动图时，指定动图在原始视频中的起始和结束时间，即「截取视频的一段」转成动图。

转动图模板

转动图的目标规格，包含动图格式、宽高和帧率等参数。云点播使用转动图模板表示转动图参数集合，通过转动图模板，可以指定以下动图相关参数。

参数	说明
格式（Format）	动图文件的输出格式，目前仅支持 GIF 和 WEBP。
宽度（Width）	动图宽度，范围是128px - 4096px。
高度（Height）	动图高度，范围是128px - 4096px。
帧率（FPS）	支持的帧率范围是1fps - 60fps。

针对常见的规格，云点播提供了 预置转动图模板。另外，您还可以通过控制台创建和管理自定义转动图模板，具体操作请参考 模板设置。

任务发起

发起转动图任务，有“通过服务端 API 直接发起”，“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。

以下是各种方式发起转动图任务的说明：

- 调用服务端 API ProcessMedia 发起任务：在请求中的 `MediaProcessTask.AnimatedGraphicTaskSet` 参数指定 转动图模板 的模板 ID。
- 通过控制台对视频发起任务：在控制台 添加任务流，任务流中设置目标转动图规格；在控制台使用该任务流 发起视频处理。

- 服务端上传时指定任务：在控制台 添加任务流，任务流中设置目标转动图规格；申请上传 中的 `procedure` 参数指定为该任务流。
- 客户端上传时指定任务：在控制台 添加任务流，任务流中设置目标转动图规格；在 客户端上传签名 中的 `procedure` 指定该任务流。
- 控制台上传：在控制台 添加任务流，任务流中设置目标转动图规格；通过控制台上传视频，选择【上传的同时对视频进行处理操作】并指定视频上传后执行该任务流。

结果获取

发起转动图任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取转动图的执行结果。下面是发起转动图任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：


```
{
"EventType":"ProcedureStateChanged",
"ProcedureStateChangeEvent":{
"TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
"Status":"FINISH",
"FileId":"5285890784246869930",
"FileName":"动物世界",
"FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
"MetaData":{
"AudioDuration":60,
"AudioStreamSet":[
{
"Bitrate":383854,
"Codec":"aac",
"SamplingRate":48000
}
],
"Bitrate":1021028,
"Container":"mov,mp4,m4a,3gp,3g2,mj2",
"Duration":60,
"Height":480,
"Rotate":0,
"Size":7700180,
"VideoDuration":60,
"VideoStreamSet":[
{
"Bitrate":637174,
"Codec":"h264",
"Fps":23,
"Height":480,
"Width":640
```

```
}
],
"Width":640
},
"MediaProcessResultSet":[
{
"Type":"AnimatedGraphics",
"AnimatedGraphicTask":{
"Status":"SUCCESS",
"ErrCode":0,
"Message":"",
"Input":{
"Definition":20001,
"StartTimeOffset":2,
"StartTimeOffset":5
},
"Output":{
"Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/v.f20001.webp",
"Definition":20001,
"Container":"webp",
"Height":480,
"Width":640,
"Bitrate":324271,
"Size":121601,
"Md5":"084d403c73930ca2f835679af1f37bd3",
"StartTimeOffset":3,
"EndTimeOffset":5
}
}
}
],
"TasksPriority":0,
"TasksNotifyMode":""
}
}
```


回调结果中，`ProcedureStateChangeEvent.MediaProcessResultSet` 有一个 `Type` 为 `AnimatedGraphics` 类型的转动图结果，`Definition` 为20001。

转自适应码流

最近更新时间：2024-02-20 11:58:51

转自适应码流是指将视频转码并打包生成自适应码流输出文件的过程。它的特点是包含多个码率的音视频文件和一个描述性文件（manifest），播放器能够根据当前带宽，动态选择最合适的码率播放。目前应用最广泛的自适应码流格式，是 Master Playlist 格式下的 HLS。
云点播支持将视频转出 HLS 和 MPEG-DASH 格式的自适应码流，使用该功能您将获得：
播放器根据当前带宽动态选择合适的码率播放，为观看者带来良好的体验。
主流播放器原生支持 HLS 自适应码流，无需定制播放器。
云点播提供了 播放器 SDK，集成后可以快速便利地播放自适应码流。

转自适应码流模板

通过转自适应码流参数，可以控制自适应码流中各个子流的“视频转码参数”、“音频转码参数”等参数。云点播使用转自适应码流模板表示参数集合，通过转自适应码流模板，可以指定以下相关参数。

参数	说明
打包类型	自适应码流的格式，目前支持 HLS 和 MPEG-DASH
加密类型	加密类型目前仅 HLS 格式支持 SimpleAES 加密，DASH 不支持加密
子流规格	控制输出多少个子流，以及各个子流的视频转码参数和音频转码参数： 视频转码参数：分辨率、码率、帧率、编码格式等 音频转码参数：采样频率、声道数、编码格式等
是否过滤“低分辨率转高分辨率”	通常来说，低分辨率的原始视频转码高分辨率无法获得画质和音质的提升。开启过滤“低分辨率转高分辨率”，可以避免不必要的转码

针对常见的参数组合，云点播提供了 预置转自适应码流模板，同时也支持自定义转自适应码流模板。

任务发起

发起转自适应码流任务，有“通过服务端 API 直接发起”，“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。
以下是各种方式发起转自适应码流任务的说明：
调用服务端 API ProcessMedia 发起任务：在请求中
的 `MediaProcessTask.AdaptiveDynamicStreamingTaskSet` 参数指定 转自适应码流模板 的模板 ID。

通过控制台对视频发起任务：调用 服务端 API 创建任务流，任务流中配置转自适应码流任务（`MediaProcessTask.AdaptiveDynamicStreamingTaskSet` 中指定）；在控制台使用该任务流 发起视频处理。
服务端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置转自适应码流任务（`MediaProcessTask.AdaptiveDynamicStreamingTaskSet` 中指定）；申请上传 中的 `procedure` 参数指定为该任务流。
客户端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置转自适应码流任务（`MediaProcessTask.AdaptiveDynamicStreamingTaskSet` 中指定）；在 客户端上传签名 中的 `procedure` 指定该任务流。
控制台上传：调用 服务端 API 创建任务流，任务流中配置转自适应码流任务（`MediaProcessTask.AdaptiveDynamicStreamingTaskSet` 中指定）；通过控制台上传视频，选择【上传的同时对视频进行处理操作】并指定视频上传后执行该任务流。

结果获取

发起转自适应码流任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取转自适应码流任务的执行结果。下面是发起转自适应码流任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：

```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784246869930",
        "FileName":"动物世界",
        "FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
        "MetaData":{
            "AudioDuration":60,
            "AudioStreamSet":[
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
```

```
            "VideoStreamSet":[
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
                    "Height":480,
                    "Width":640
                }
            ],
            "Width":640
        },
        "MediaProcessResultSet":[
            {
                "Type":"AdaptiveDynamicStreaming",
                "AdaptiveDynamicStreamingTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "Definition":10,
                        "Package":"hls",
                        "DrmType":"",
                        "Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/adp.10.m
                    }
                }
            },
            {
                "Type":"AdaptiveDynamicStreaming",
                "AdaptiveDynamicStreamingTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":20
                    },
                    "Output":{
                        "Definition":20,
                        "Package":"dash",
                        "DrmType":"",
                        "Url":"http://1256768367.vod2.myqcloud.com/xxx/xxx/adp.20.m
                    }
                }
            }
        ],
```

```
            "TasksPriority":0,
            "TasksNotifyMode":""
        }
    }
```

回调结果中，`ProcedureStateChangeEvent.MediaProcessResultSet` 有两个 `Type` 为 `AdaptiveDynamicStreaming` 类型的转自适应码流结果，`Definition` 分别为10和20。

图片处理
图片处理综述

最近更新时间：2022-12-01 16:30:24

图片处理是对点播媒资中的原始图片进行分析或加工，输出处理结果的过程。

类别	说明
图片即时处理	支持的操作类型： • 缩放：指定宽高或长短边对原始图片进行缩放生成新图片。 • 裁剪：以原始图片的几何中心为裁剪中心，指定半径裁剪生成圆形图片或者指定宽高裁剪生成矩形图片。

以上是云点播提供的图片处理功能清单，发起图片处理后，将立即得到处理结果（即同步地获取结果）。关于如何发起图片处理并获取结果，请参见 图片即时处理。

图片即时处理

最近更新时间：2022-12-01 16:30:24

对图片修改尺寸、部分裁剪等编辑处理存在较多的应用场景，比如媒资控制台展示缩略图、对用户上传的图片按统一尺寸裁剪生成圆形/方形用户头像等等。在云端图片编辑场景下，传统图片编辑方式（如本地编辑软件、Web 在线编辑）并非最佳实践，实际使用中存在诸多痛点，例如：

- 操作流程复杂（一般要经过下载、编辑、上传回云端等繁琐步骤）
- 手工编辑效率低且易出错。

使用云点播图片即时处理可一举解决上述全部痛点。

维度	传统图片编辑	云点播-图片即时处理
处理流程	需要下载、编辑、上传等多个步骤，操作费时费力。	直接在云端完成图片编辑所有流程，省去下载和上传步骤。
操作方式	图片编辑软件的使用门槛较高，对编辑人员的图片编辑能力有一定的要求。	通过在 URL 实时指定图片编辑参数，零门槛上手操作。
访问速度	通过云存储链接进行访问和下载图片的速度会比较慢，影响用户体验。	通过云点播 CDN 进行全网加速分发，极速获取处理后的图片。

云点播图片即时处理支持缩略和裁剪两种操作，具体如下：

操作类型	详细操作
缩略	指定宽度，高度等比缩放。
	指定高度，宽度等比缩放。
	指定长边，短边等比缩放。
	指定短边，长边等比缩放。
	指定宽度与高度强制缩放。
裁剪	内切圆裁剪，指定裁剪半径。
	矩形裁剪，指定裁剪的宽度和高度。

图片即时处理的具体使用方式，请参见 图片即时处理 。

预置参数模板列表

最近更新时间：2023-03-07 11:47:21

云点播支持使用参数模板代替复杂的参数集合，发起图片处理。针对不同处理场景，云点播预置了一批参数模板。

图片即时处理

一个图片即时处理模板代表了一组有序的操作集合。

预置图片即时处理模板

模板 ID	操作说明	操作参数
10	裁剪	• 类型：矩形裁剪 • 宽度：360px • 高度：200px
20	裁剪	• 类型：矩形裁剪 • 宽度：200px • 高度：400px
30	缩略	• 类型：指定短边，长边等比例缩放 • 短边：320px
40	缩略	• 类型：强制指定宽、高 • 宽度：200px • 高度：200px
50	先缩略，后裁剪	缩略： • 类型：指定短边，长边等比例缩放 • 短边：320px 裁剪： • 类型：矩形裁剪 • 宽度：200px • 高度：200px

图片审核

预置图片审核模板

模板 ID	色情（Porn）	暴恐（Terror）
10	是	是

媒体 AI
音视频内容审核

最近更新时间：2023-10-13 17:30:22

音视频内容审核是借助于 AI 对音视频内容进行智能审核，是一种离线任务。任务的执行结果中，包括审核评分、审核建议和嫌疑视频片段。根据“审核建议”，视频管理者可以决定视频是否允许公开，有效规避违规视频带来的法律风险和品牌伤害。

云点播可以对画面图像、画面中的文本、语音中的文本以及声音内容四种对象进行审核，审核标签包括色情、暴力和娇喘。

对象	审核标签
画面图像	色情（Porn）
	暴力（Terror）
声音	娇喘（Moan）
语音中的文本（ASR）	色情（Porn）
	暴力（Terror）
画面中的文本（OCR）	色情（Porn）
	暴力（Terror）

音视频审核结果部分字段说明：

字段名	类型	含义
Confidence	Float	审核评分（0 - 100），评分越高，嫌疑越大。
Suggestion	String	审核建议，有 pass，review，block 三种： pass：嫌疑度不高，建议直接通过。 review：嫌疑度较高，建议人工复核。 block：嫌疑度很高，建议直接屏蔽。
Form	String	审核形式，有如下几种： Image：画面图像。 Voice：声音。 OCR：画面中的文字。 ASR：语音中的文字。
Label	String	审核标签，有如下几种：

		Porn：色情。 Terror：暴力。 Moan：娇喘。

音视频审核模板

通过音视频审核参数，可以控制审核任务具体检测哪些审核标签。云点播使用视频审核模板来表示审核参数集合，通过视频审核模板，可以指定审核任务中检测哪一项或几项标签：

色情（Porn）

暴力（Terror）

娇喘（Moan）

针对常见的操作组合，云点播提供了 预置音视频审核模板。另外，您还可以调用 服务端 API 创建和管理自定义视频审核模板。

任务发起

发起音视频内容审核任务，有“通过服务端 API 直接发起”、“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。

以下是各种方式发起音视频内容审核任务的说明：

通过服务端 API 直接发起。调用服务端 API ReviewAudioVideo 发起任务。

通过控制台直接发起。请参见控制台指南 音视频审核。

服务端上传时指定任务：在控制台 添加任务流，任务流中开启内容审核；申请上传 中的 `procedure` 参数指定为该任务流。

客户端上传时指定任务：在控制台 添加任务流，任务流中开启内容审核；在 客户端上传签名 中的 `procedure` 指定该任务流。

控制台上传：在控制台 添加任务流，任务流中开启内容审核；通过控制台上传视频，选择 上传的同时对视频进行处理操作 并指定视频上传后执行该任务流。

结果获取

发起视频审核任务后，您可以通过异步等待 音视频审核完成 和同步进行 任务查询 两种方式获取视频审核任务的执行结果。下面是发起审核任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：

```
{
    "EventType": "ReviewAudioVideoComplete",
    "ReviewAudioVideoCompleteEvent": {
        "TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
```

```
        "Status": "FINISH",
        "ErrCodeExt": "",
        "Message": "SUCCESS",
        "Input": {
            "FileId": "387702130626135215"
        },
        "Output": {
            "Suggestion": "block",
            "Label": "Porn",
            "Form": "Image",
            "SegmentSet": [
                {
                    "StartTimeOffset": 0,
                    "EndTimeOffset": 1,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:16.039Z"
                },
                {
                    "StartTimeOffset": 1,
                    "EndTimeOffset": 2,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:17.039Z"
                },
                {
                    "StartTimeOffset": 2,
                    "EndTimeOffset": 3,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
```

```
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:18.039Z"
            },
            {
                "StartTimeOffset": 3,
                "EndTimeOffset": 4,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:19.039Z"
            },
            {
                "StartTimeOffset": 4,
                "EndTimeOffset": 5,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:20.039Z"
            },
            {
                "StartTimeOffset": 5,
                "EndTimeOffset": 6,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:21.039Z"
```

```
            },
            {
                "StartTimeOffset": 6,
                "EndTimeOffset": 7,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:22.039Z"
            },
            {
                "StartTimeOffset": 7,
                "EndTimeOffset": 8,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:23.039Z"
            },
            {
                "StartTimeOffset": 8,
                "EndTimeOffset": 9,
                "Confidence": 99,
                "Suggestion": "block",
                "Label": "Porn",
                "SubLabel": "SexyBehavior",
                "Form": "Image",
                "AreaCoordSet": [],
                "Text": "",
                "KeywordSet": [],
                "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                "PicUrlExpireTime": "2023-01-16T03:06:24.039Z"
            },
            {
                "StartTimeOffset": 9,
                "EndTimeOffset": 10,
                "Confidence": 99,
```

```
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:25.039Z"
                }
            ],
            "SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtra
            "SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
        },
        "SessionContext": "",
        "SessionId": ""
    }
}
```


回调结果中，`ReviewAudioVideoCompleteEvent.Output` 为音视频审核结果的输出，`Output.Suggestion` 表示整体审核建议，这里为 `block` 即建议直接屏蔽。`Output.Label=Porn` 和 `Output.Form=Image` 说明最有可能的违规内容是视频画面包含色情的信息。

一个音视频可能存在多个违规片段，`Output.SegmentSet` 列出其中的前 10 个片段（完整的违规结果可以在链接有效期内通过 `Output.SegmentSetFileUrl` 获取）。

每个违规片段的 `StartTimeOffset` 和 `EndTimeOffset` 标明了片段在原始视频里的起止时间，`SubLabel` 标明了片段具体的违规内容。

对于画面文字或语音文字识别：

`Text` 标明片段识别出来的完整文字内容。

`KeywordSet` 标明命中的违规关键词列表。

对于视频画面（人和物体）以及画面文字识别：

`AreaCoordSet` 标明违规对象的区域坐标。

`Url` 为违规画面截图链接。

`PicUrlExpireTime` 为 `Url` 过期时间，超过后链接不可访问。

音视频内容分析

最近更新时间：2023-03-22 14:46:48

视频内容分析，即借助 AI 对音视频内容进行智能分析，是一种离线任务。使用音视频内容分析，可以对视频分类、打标签及截取封面等给出智能建议，帮助视频平台准确高效地管理视频。
音视频内容分析包括以下功能：

功能名称	说明
智能分类	对视频所属的分类给出建议，目前有十余个类别，包括： 新闻、娱乐、游戏、科技、美食、体育、旅行、动漫、舞蹈、音乐、影视及汽车等。
智能标签	对视频可以打上的标签给出建议，目前共有3000余种标签，例如： 游戏、交通工具、音乐家、赛车、宠物、架子鼓、自行车、魔兽世界、电脑、学校及夹克等。
智能封面	从视频中选出一张或几张截图，作为推荐采用的封面。
智能按帧标签	为视频逐帧画面，给出打标签的建议，目前共有1000余种标签，例如： 现代舞、水上运动、牛排、宝宝、小猫、一年生植物、驱逐舰、漫画、草坪、婚纱、多功能厅及护照等。

音视频内容分析模板

通过音视频内容分析参数，可以控制分析任务具体执行哪几项分析操作。云点播使用音视频内容分析模板来表示智能分析参数集合：
是否启用智能分类。
是否启用智能标签。
是否启用智能封面。
是否启用智能按帧标签。
针对常见的操作组合，云点播提供了 预置音视频内容分析模板。另外，您还可以调用 服务端 API 创建和管理自定义音视频内容分析模板。

任务发起

发起音视频内容分析任务，有“通过服务端 API 直接发起”、“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。
以下是各种方式发起音视频内容分析任务的说明：

调用服务端 API ProcessMedia 发起任务：在请求中的 `AiAnalysisTask` 参数指定 音视频内容分析模板 的模板 ID。
通过控制台对视频发起任务：调用 服务端 API 创建任务流，任务流中配置音视频内容分析任务（`MediaProcessTask.AiAnalysisTask` 中指定）；在控制台使用该任务流 发起视频处理。
服务端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置音视频内容分析任务（`MediaProcessTask.AiAnalysisTask` 中指定）；申请上传 中的 `procedure` 参数指定为该任务流。
客户端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置音视频内容分析任务（`MediaProcessTask.AiAnalysisTask` 中指定）；在 客户端上传签名 中的 `procedure` 指定该任务流。
控制台上传：调用 服务端 API 创建任务流，任务流中配置音视频内容分析任务（`MediaProcessTask.AiAnalysisTask` 中指定）；通过控制台上传视频，选择 上传的同时对视频进行处理 操作 并指定视频上传后执行该任务流。

结果获取

发起音视频内容分析任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取视频内分析任务的执行结果。下面是发起内容分析任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：

```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784246869930",
        "FileName":"动物世界",
        "FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
        "MetaData":{
            "AudioDuration":60,
            "AudioStreamSet":[
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
            "VideoStreamSet":[
```

```
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
                    "Height":480,
                    "Width":640
                }
            ],
            "Width":640
        },
        "AiAnalysisResultSet":[
            {
                "Type":"Classification",
                "ClassificationTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "ClassificationSet":[
                            {
                                "Classification":"动物",
                                "Confidence":80
                            },
                            {
                                "Classification":"旅行",
                                "Confidence":34
                            }
                        ]
                    }
                }
            },
            {
                "Type":"Cover",
                "CoverTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "CoverSet":[
                            {
                                "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
```

```
                            "Confidence":79
                        },
                        {
                            "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
                            "Confidence":70
                        },
                        {
                            "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
                            "Confidence":66
                        }
                    ]
                }
            }
        },
        {
            "Type":"Tag",
            "TagTask":{
                "Status":"SUCCESS",
                "ErrCode":0,
                "Message":"",
                "Input":{
                    "Definition":10
                },
                "Output":{
                    "TagSet":[
                        {
                            "Tag":"马",
                            "Confidence":34
                        },
                        {
                            "Tag":"鸟",
                            "Confidence":27
                        },
                        {
                            "Tag":"植物",
                            "Confidence":13
                        },
                        {
                            "Tag":"海滩",
                            "Confidence":11
                        }
                    ]
                }
            }
        }
    ],
    "TasksPriority":0,
```

```
            "TasksNotifyMode":""
        }
    }
```

回调结果

中，`ProcedureStateChangeEvent.AiAnalysisResultSet` 有 `Type` 为 `Classification` 、 `Cover` 和 `Tag` 三种类型的分析结果，分别代表视频智能分类、视频智能封面和视频智能标签。

`Type` 为 `Classification` 的结果显示， `Output.ClassificationSet` 置信度最高的分类是 `动物` ，其次的分类则是 `旅行` 。

`Type` 为 `Cover` 的结果 `Output.CoverSet` ，给出了3个建议采用的封面， `CoverUrl` 是对应封面的下载地址。

`Type` 为 `Tag` 的结果 `Output.TagSet` ，给出了4个视频建议采用的标签，按照置信度从高到低排列。

音视频内容识别

最近更新时间：2024-11-04 10:11:28

腾讯云点播（VOD）服务于2022年08月01日新增音视频内容识别计费项并对用户发起的音视频内容识别任务将进行正式计费，具体详情请参见 关于音视频内容识别正式商业计费的公告 。

音视频内容识别，即借助 AI 对音视频内容进行智能识别，是一种离线任务。使用音视频内容识别，可以识别出视频画面中的人脸、文字、片头片尾以及语音中的文字。根据音视频内容识别的结果，您可以准确高效地管理视频。音视频内容识别包括以下功能：

功能名称	功能说明	用途举例
人脸识别	识别画面中出现的人脸	标注明星在画面中出现的位置。 排查画面中出现的相关人物。
语音全文识别	识别语音中出现的全部文字	为演讲内容生成字幕。 对视频的语音内容做数据分析。
文本全文识别	识别画面中出现的全部文字	对画面中的文字做数据分析。
语音关键词识别	识别语音中存在的关键词	排查语音中的敏感词。 检索语音中提到的特定关键词。
文本关键词识别	识别画面中存在的关键词	排查画面中的敏感词。 检索画面中出现的特定关键词。
视频片头片尾识别	识别视频的片头和片尾	进度条中标注片头、片尾和正片的位置。 批量对视频掐头去尾。
语音翻译识别	识别语音中出现的全部文字，同时翻译为指定语言	为短剧生成翻译字幕。 对跨国音视频会议录制文件生成多语言字幕。

部分内容识别功能需要依赖素材库，有公共库和自定义库两种：

公共库：点播预置好的素材库。

自定义库：用户自行创建和管理的素材库。

识别类型	公共库	自定义库
人脸识别	支持，素材人物主要涉及娱乐明星、体育明星和相关人物。	支持，调用 服务端 API 管理自定义人脸库。
语音单词识别	暂不支持。	支持，调用 服务端 API 管理关键词库。

文字单词识别	暂不支持。	支持，调用 服务端 API 管理关键词库。

音视频内容识别模板

音视频内容识别集成了多项识别功能，需要通过参数进行精细化控制，控制的目标如：
启用的识别类型：启用内容识别中的哪几项功能。
使用的素材库：对于人脸识别，使用公共库还是自定义库。
指定过滤分数：人脸识别的置信度达到多少分的结果才返回。
指定过滤标签：人脸的标签在什么范围内的结果才返回。
针对常见的操作组合，云点播提供了 预置音视频内容识别模板。另外，您还可以调用 服务端 API 创建和管理自定义音视频内容识别模板。

任务发起

发起音视频内识别任务，有“通过服务端 API 直接发起”、“通过控制台直接发起”和“上传时指定要执行的任务”三种方式。具体请参照视频处理的 任务发起。
以下是各种方式发起音视频内容识别任务的说明：
调用服务端 API ProcessMedia 发起任务：在请求中的 `AiRecognitionTask` 参数指定 音视频内容识别模板 的模板 ID。
通过控制台对视频发起任务：调用 服务端 API 创建任务流，任务流中配置音视频内容识别任务（`MediaProcessTask.AiRecognitionTask` 中指定）；在控制台使用该任务流 发起视频处理。
服务端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置音视频内容识别任务（`MediaProcessTask.AiRecognitionTask` 中指定）；申请上传 中的 `procedure` 参数指定为该任务流。
客户端上传时指定任务：调用 服务端 API 创建任务流，任务流中配置音视频内容识别任务（`MediaProcessTask.AiRecognitionTask` 中指定）；在 客户端上传签名 中的 `procedure` 指定该任务流。
控制台上传：调用 服务端 API 创建任务流，任务流中配置音视频内容识别任务（`MediaProcessTask.AiRecognitionTask` 中指定）；通过控制台上传视频，选择 上传的同时对视频进行处理操作 并指定视频上传后执行该任务流。

结果获取

发起音视频内容识别任务后，您可以通过异步等待 结果通知 和同步进行 任务查询 两种方式获取视频内识别任务的执行结果。下面是发起内容识别任务后，普通回调方式下结果通知的示例（省略了值为 null 的字段）：

```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1400155958-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784363430543",
        "FileName":"集锦",
        "FileUrl":"http://1400155958.vod2.myqcloud.com/xxx/xxx/aHjWUx5Xo1EA.mp4",
        "MetaData":{
            "AudioDuration":243,
            "AudioStreamSet":[
                {
                    "Bitrate":125599,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1459299,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":243,
            "Height":1080,
            "Rotate":0,
            "Size":44583593,
            "VideoDuration":243,
            "VideoStreamSet":[
                {
                    "Bitrate":1333700,
                    "Codec":"h264",
                    "Fps":29,
                    "Height":1080,
                    "Width":1920
                }
            ],
            "Width":1920
        },
        "AiRecognitionResultSet":[
            {
                "Type":"FaceRecognition",
                "FaceRecognitionTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
```

```
                "ResultSet":[
                    {
                        "Id":183213,
                        "Type":"Default",
                        "Name":"张三",
                        "SegmentSet":[
                            {
                                "StartTimeOffset":10,
                                "EndTimeOffset":12,
                                "Confidence":97,
                                "AreaCoordSet":[
                                    830,
                                    783,
                                    1030,
                                    599
                                ]
                            },
                            {
                                "StartTimeOffset":12,
                                "EndTimeOffset":14,
                                "Confidence":97,
                                "AreaCoordSet":[
                                    844,
                                    791,
                                    1040,
                                    614
                                ]
                            }
                        ]
                    },
                    {
                        "Id":236099,
                        "Type":"Default",
                        "Name":"lisi",
                        "SegmentSet":[
                            {
                                "StartTimeOffset":120,
                                "EndTimeOffset":122,
                                "Confidence":96,
                                "AreaCoordSet":[
                                    579,
                                    903,
                                    812,
                                    730
                                ]
                            }
                        ]
```

```
                                }
                            ]
                        }
                    }
                }
            ],
            "TasksPriority":0,
            "TasksNotifyMode":""
        }
    }
```

回调结果

中，`ProcedureStateChangeEvent.AiRecognitionResultSet` 有 `Type` 为 `FaceRecognition` 的识别结果，代表人脸识别。

`Type` 为 `FaceRecognition` 的结果显示，`Output.ResultSet` 中包含了两个识别出的人物，分别是 `张三` 和lisi。`SegmentSet` 表示人脸出现在视频中的时间段（由 `StartTimeOffset` 和 `EndTimeOffset` 确定）和在画面中的坐标（由 `AreaCoordSet` 确定）。

图片审核

最近更新时间：2023-04-17 15:01:47

云点播图片审核借助于 AI 审核出违规信息。审核的结果包括审核评分、审核建议。根据“审核建议”，媒体管理者可以决定图片是否允许公开，有效规避违规图片带来的法律风险和品牌伤害。
云点播可以对画面图像、画面中的文本两种种对象进行审核，审核标签包括色情、暴力、不适宜的信息、违法、谩骂、广告等。

对象	审核标签
画面图像	色情（Porn）
	暴力（Terror）
	不适宜的信息（Polity）
	广告（Ad）
	违法（Illegal）
画面中的文本（OCR）	色情（Porn）
	暴力（Terror）
	不适宜的信息（Polity）
	广告（Ad）
	违法（Illegal）
	谩骂（Abuse）

审核结果如下表所示：

字段名	类型	含义
Confidence	Float	审核评分（0 - 100），评分越高，嫌疑越大。
Suggestion	String	审核建议，有 pass，review，block 三种： pass：嫌疑度不高，建议直接通过。 review：嫌疑度较高，建议人工复核。 block：嫌疑度很高，建议直接屏蔽。

发起处理

有两种方式发起图片审核：通过云点播 控制台 操作或服务端 API 图片审核 调用。

获取结果

不论以何种方式发起处理，审核结果都立即返回（即同步返回）。
通过云点播控制台发起的，输出结果参见云点播 控制台 ；通过服务端 API 图片审核 调用发起的，输出结果数据格式参见 图片审核 - 3. 输出参数。

事件通知
事件通知综述

最近更新时间：2021-10-29 11:40:44

对云点播中的视频发起的上传、删除、视频处理等操作，都可以被称为一个事件。事件的执行需要一段时间才能完成，云点播在事件结束时，会立即通知 App 服务操作的执行结果，即事件通知。

云点播支持以下几种事件通知：

归类	事件通知
上传删除类	视频上传完成
	URL 拉取视频上传完成
	视频删除完成
视频处理类	任务流状态变更
	视频编辑完成
	视频合成完成

事件通知方式分为“普通回调”和“可靠回调”，您可以登录 云点播控制台 设置回调模式，选择您需要接收回调的事件，具体操作请参见 回调设置。

- 普通回调：在控制台上配置一个回调 URL，系统在事件完成后向该 URL 发送 HTTP 请求，请求体中包含通知内容。
- 可靠回调：在事件完成后，云点播系统将通知内容放入内置的队列，App 服务通过服务端 API 消费队列中的通知。

普通回调

普通回调是 App 服务被动接收事件通知的模式。配置回调 URL 并选择普通回调模式后，云点播会在事件完成后，向回调 URL 发起回调。

云点播发起的普通回调的形式是 HTTP 请求，请求体为 JSON 格式，内容为不含 EventHandle 参数的 EventContent 结构。
以 任务状态变更通知 为例，回调中的 `EventType` 参数为 `ProcedureStateChanged`，信息由 `ProcedureStateChangeEvent` 参数表示（ProcedureTask 结构）。

可靠回调

可靠回调是 App 服务主动向云点播拉取事件通知的模式。选择可靠回调模式后，云点播系统将把事件通知放入队列中，App 服务通过服务端 API 从队列中依次消费事件通知。
App 服务通过 拉取事件通知 API 获取消息后，需要调用 确认事件通知 API 进行确认。消息必须被确认之后，才会从云点播中的队列中删除，所以“可靠回调”的可靠性高于“普通回调”。**如果对事件通知的可靠性要求高，建议使用“可靠回调”模式**。

事件通知入门教程

最近更新时间：2022-05-31 11:47:08

本教程将详细地指导您如何使用云点播的事件通知，包括“普通回调”和“可靠回调”两种方式。

前提条件

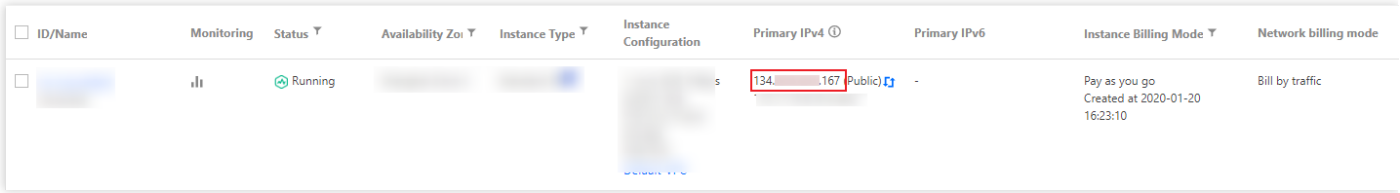
- 注册腾讯云 账号，并完成 实名认证，未进行实名认证的用户无法购买中国境内的云点播实例。
- 普通回调下，需要 Python 2.7 运行环境。

普通回调

部署回调接收服务

以 普通回调 的方式获取事件通知，需要在有公网 IP 的服务器上，部署回调接收服务。这里以 CVM 为例，介绍如何部署该服务：

1. 进入云服务器 CVM 控制台的 实例列表 界面，单击【新建】。
2. 选择【快速配置】菜单，【镜像】选择“**Ubuntu Server**”或“**CentOS**”，【公网带宽】选择“**1Mbps**”并勾选“**分配免费公网 IP**”，然后单击【立即购买】。
3. 再次进入 实例列表 界面，找到您创建成功的 CVM 示例，并拷贝【主IP地址】中的公网 IP（示例中的 IP 是 **134.XXX.XXX.167**）。



4. 登录您购买的 CVM，下载 源码压缩包 并解压到您的工作目录，然后执行以下命令：

```
python NotificationReceiveServer.py
```

命令执行后，CVM 的标准输出打印 `Started httpserver on port 8080` ，表示服务进程已经启动，正在监听8080端口。

5. 在浏览器中输入 `http://134.XXX.XXX.167:8080`，CVM 的标准输出将打印如下 HTTP 请求信息。

```
[root@yanchuxiong notification]# python NotificationReceiveServer.py
Started httpserver on port  8080

----- Request Start ----->

/
Host: [redacted]:8080
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
connection: keep-alive

<----- Request End -----

103.7.29.7 - - [28/May/2019 07:59:56] "GET / HTTP/1.1" 200 -
```

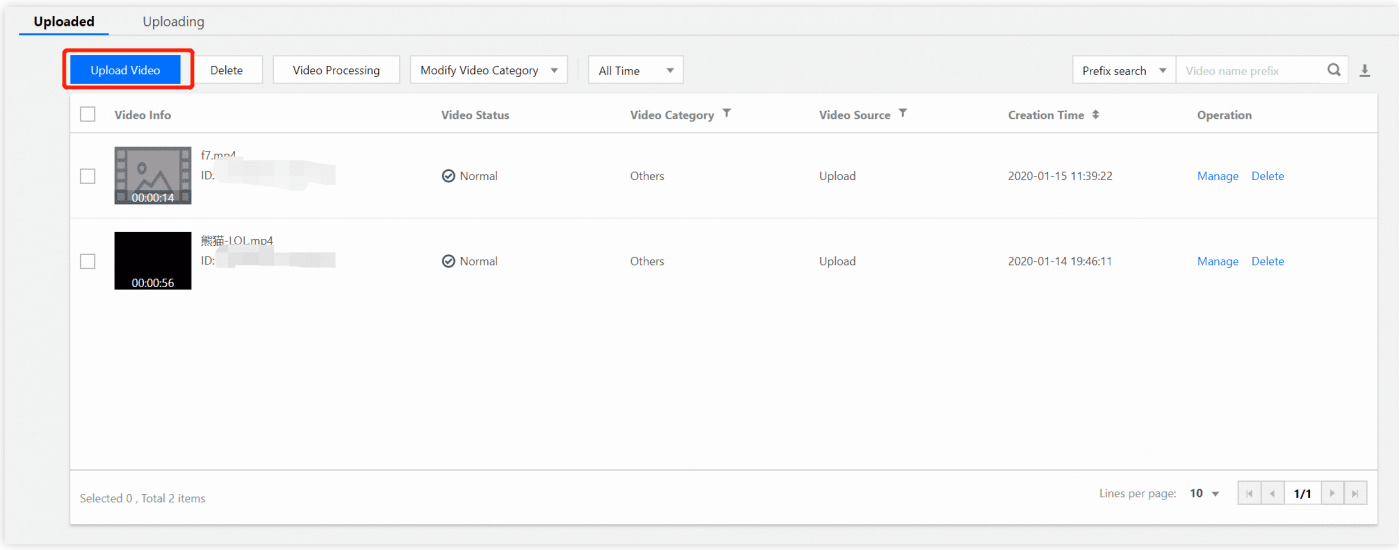
配置普通回调

1. 登录 云点播控制台，单击左侧菜单栏的【回调设置】。
2. 单击【设置】：
   - 事件通知方式：选择“普通回调”。
   - 回调 URL：填入http://134.XXX.XXX.167:8080。
   - 事件通知：勾选“视频上传完成回调”。
3. 单击【确定】完成设置。

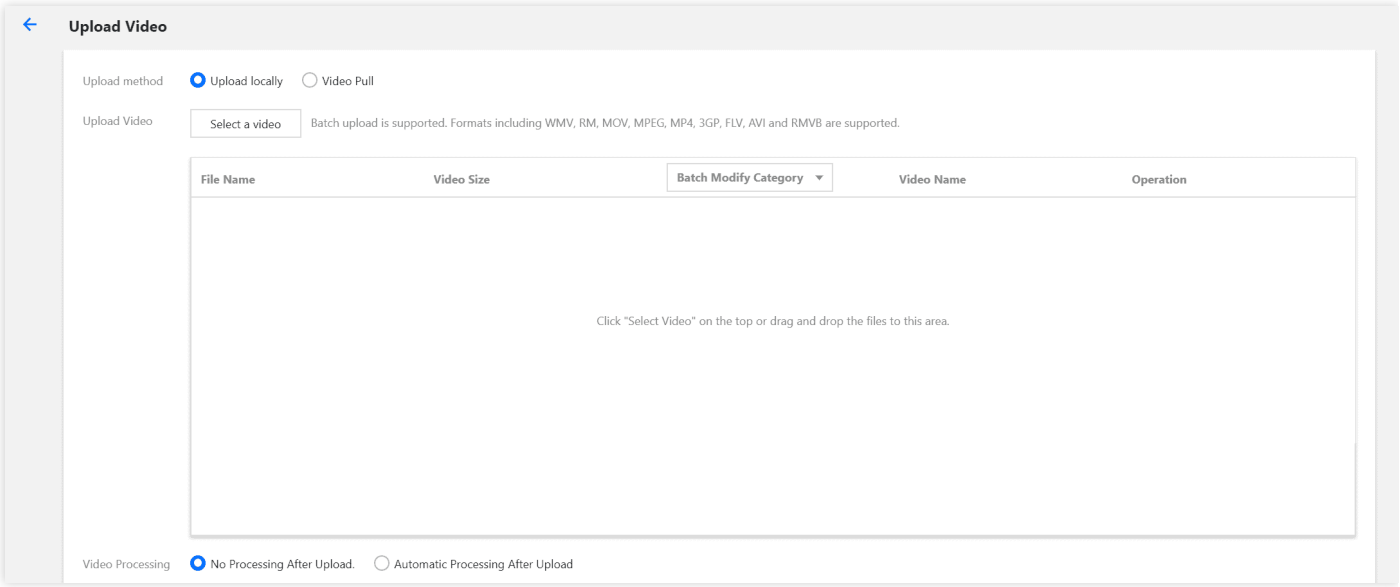
发起与接收普通回调

请下载 演示视频 到您的本地，用于入门教学。

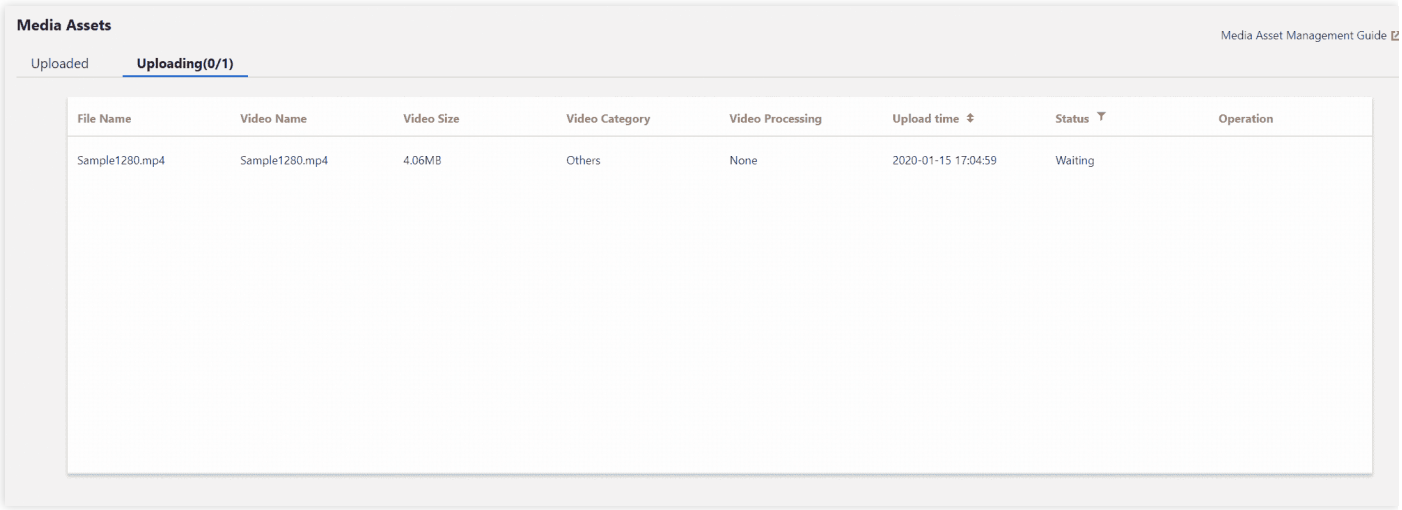
1. 单击左侧菜单栏的【媒资管理】>【视频管理】。



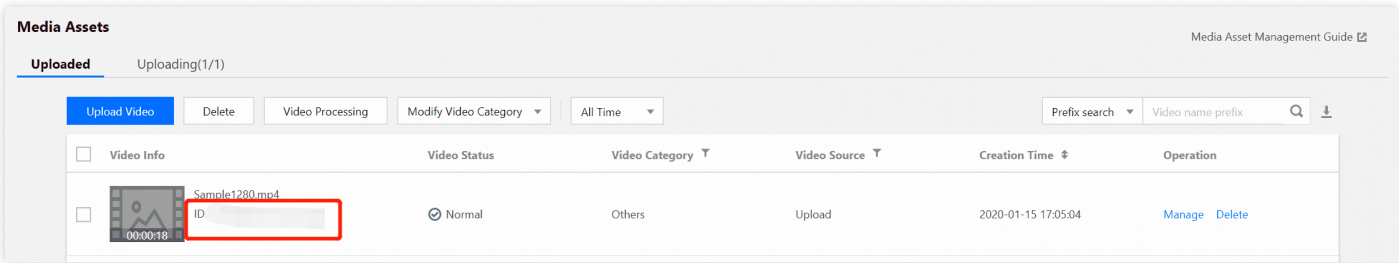
2. 弹出“**上传视频**”的对话框后，选择【本地上传】，单击【选择视频】，将**演示视频**上传到云点播平台。



执行上传之后，您将在“**正在上传**”栏看到视频的上传进度。



上传完成之后，在“**已上传**”栏的视频列表中，将看到上传完成的视频，以及视频对应的 ID（即 FileId）。



3. 检查 CVM，标准输出打印 视频上传完成 通知的内容。

4. 在【媒资管理】的“**已上传**”栏，勾选刚上传成功的视频，单击【视频处理】。【处理类型】选择“**手动选择转码模板**”选项，并在【转码模板】中勾选“**MP4-流畅-FLU (10)**”，保持【视频封面】勾选，单击【确定】。
5. 等待10分钟后，检查 CVM，标准输出打印 任务流状态变更 通知的内容，其中包括了转码（ `Type` 为 `Transcode` ）和时间点截图做封面（ `Type` 为 `CoverBySnapshot` ）的结果。

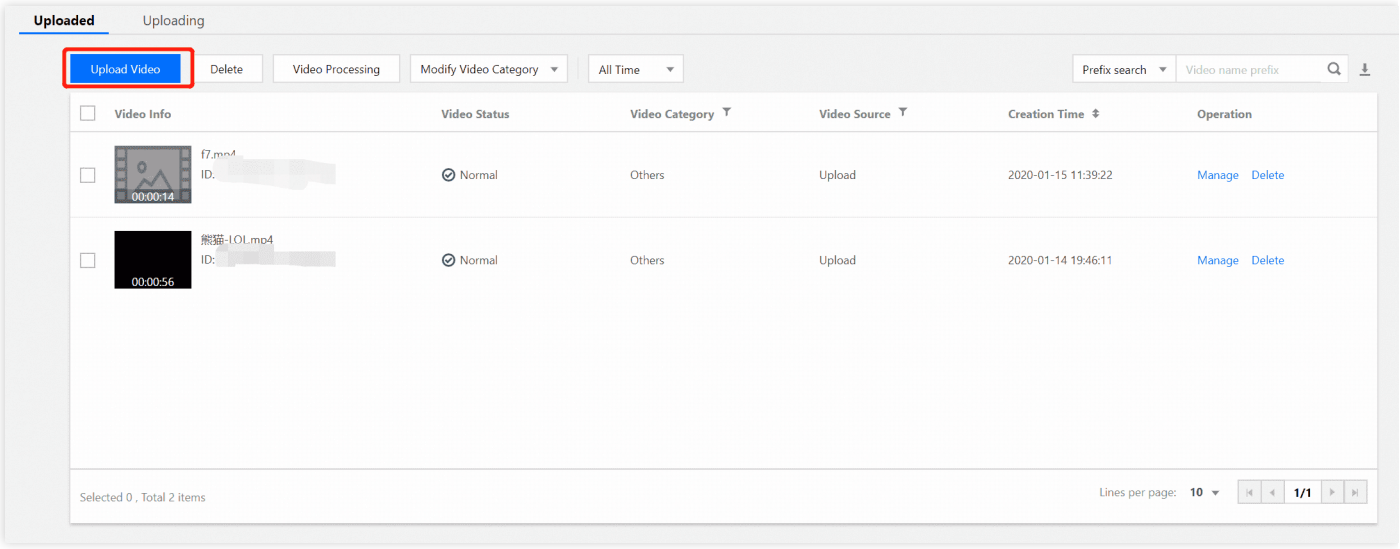
至此，您上传了一个视频并对其执行了转码任务。上传和转码完成后，您的回调接收服务分别接收到了一个“**视频上传完成**”和“**任务流状态变更**”通知。

可靠回调

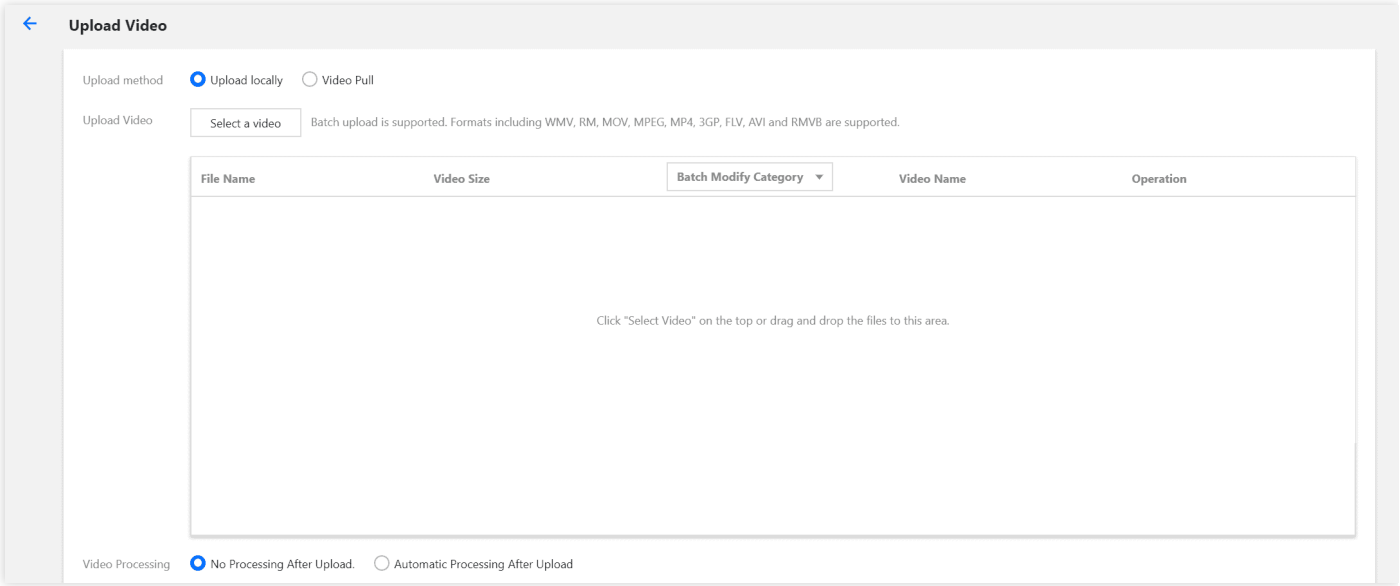
1. 登录 云点播控制台，单击左侧菜单栏的【回调设置】。
2. 单击【设置】：
   - 回调模式：选择“**可靠回调**”。
   - 回调事件：勾选“**视频上传完成回调**”。
3. 单击【确定】完成设置。

发起可靠回调

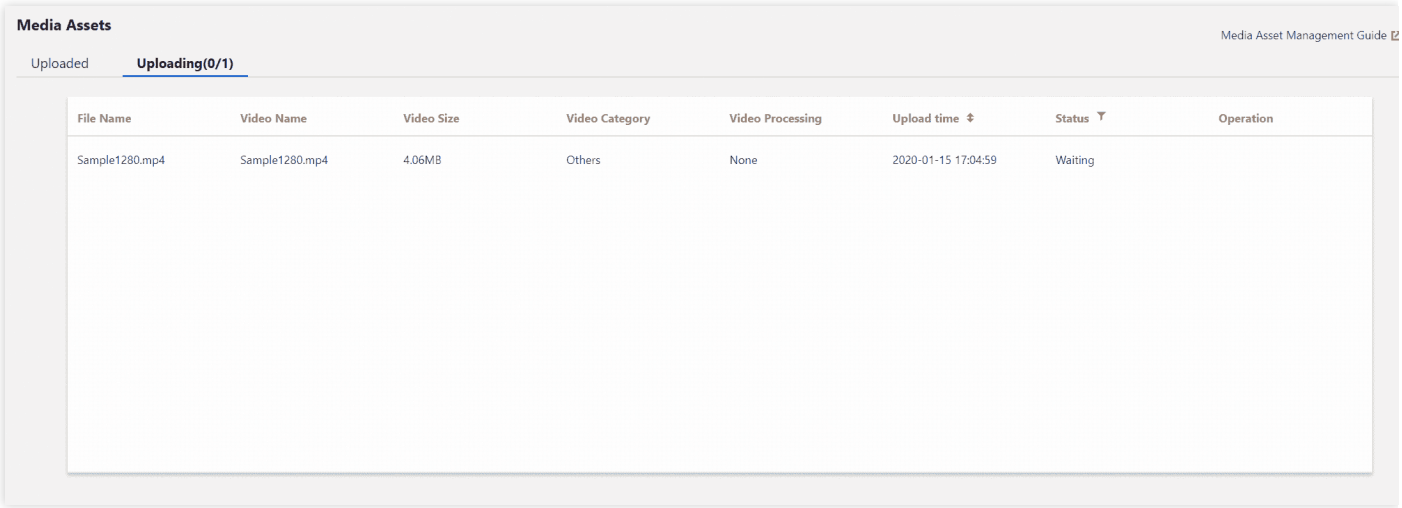
1. 单击左侧菜单栏的【媒资管理】>【视频管理】，选择【已上传】，单击【上传视频】。



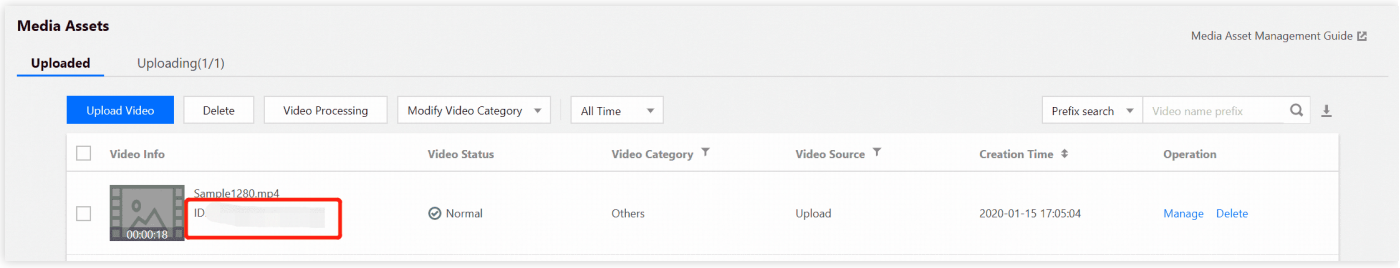
2. 弹出“**上传视频**”的对话框后，选择【本地上传】，单击【选择视频】，将**演示视频**上传到云点播平台。



执行上传之后，您将在“**正在上传**”栏看到视频的上传进度。



上传完成之后，在“**已上传**”栏的视频列表中，将看到上传完成的视频，以及视频对应的 ID（即 FileId）。



3. 在【媒资管理】的“**已上传**”栏，勾选刚上传成功的视频，单击【视频处理】。【处理类型】选择“**手动选择转码模板**”选项，并在【转码模板】中勾选“**MP4-流畅-FLU (10)**”，保持【视频封面】勾选，单击【确定】。

至此，您重新上传了一个视频，并对视频发起了转码任务。这些操作将会触发事件通知。

视频上传完成

最近更新时间：2023-03-13 11:41:15

事件名称

NewFileUpload

事件说明

当 App 配置了事件通知，并且将视频通过客户端或服务端上传后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 FileUploadTask 结构。

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。

```
{
    "EventType":"NewFileUpload",
    "FileUploadEvent":{
        "FileId":"5285890784273533167",
        "MediaBasicInfo":{
            "Name":"动物世界",
            "Description":"",
            "CreateTime":"2019-01-09T16:36:22Z",
            "UpdateTime":"2019-01-09T16:36:24Z",
            "ExpireTime":"9999-12-31T23:59:59Z",
            "ClassId":0,
            "ClassName":"其他",
            "ClassPath":"其他",
            "CoverUrl":"",
            "Type":"mp4",
            "MediaUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/q1BORBPQH1IA.
            "TagSet":[
            ],
            "StorageRegion":"ap-guangzhou-2",
            "SourceInfo":{
                "SourceType":"Upload",
                "SourceContext":""
```

```
            },
            "Vid":"5285890784273533167"
        },
        "ProcedureTaskId":"",
        "ReviewAudioVideoTaskId":""
    }
}
```

可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。

```
{
    "Response": {
        "EventSet": [
            {
                "EventHandle": "EventHandle.N",
                "EventType": "NewFileUpload",
                "FileUploadEvent": {
                    "FileId": "5285890784273533167",
                    "MediaBasicInfo": {
                        "Name": "动物世界",
                        "Description": "",
                        "CreateTime": "2019-01-09T16:36:22Z",
                        "UpdateTime": "2019-01-09T16:36:24Z",
                        "ExpireTime": "9999-12-31T23:59:59Z",
                        "ClassId": 0,
                        "ClassName": "其他",
                        "ClassPath": "其他",
                        "CoverUrl": "",
                        "Type": "mp4",
                        "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/
                        "TagSet": [],
                        "StorageRegion": "ap-guangzhou-2",
                        "SourceInfo": {
                            "SourceType": "Upload",
                            "SourceContext": ""
                        },
                        "Vid": "5285890784273533167"
                    },
                    "ProcedureTaskId": "",
                    "ReviewAudioVideoTaskId":""
                }
            }
```

```
        ],
        "RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

URL 拉取视频上传完成

最近更新时间：2023-03-13 11:41:15

事件名称

PullComplete

事件说明

当 App 配置了事件通知，并且在拉取视频上传完成后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 PullComplete 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。

```
{
    "EventType": "PullComplete",
    "PullCompleteEvent": {
        "TaskId": "125676836723-Pull-f5ac8127b3b6b85cdc13f237c6005d8",
        "Status": "FINISH",
        "ErrCode": 0,
        "Message": "SUCCESS",
        "FileId": "14508071098244959037",
        "MediaBasicInfo": {
            "Name": "动物世界",
            "Description": "",
            "CreateTime": "2019-01-09T16:36:22Z",
            "UpdateTime": "2019-01-09T16:36:24Z",
            "ExpireTime": "9999-12-31T23:59:59Z",
            "ClassId": 0,
            "ClassName": "其他",
            "ClassPath": "其他",
            "CoverUrl": "",
            "Type": "mp4",
            "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
            "TagSet": [ ],
```

```
            "StorageRegion": "ap-guangzhou-2",
            "SourceInfo": {
                "SourceType": "Upload",
                "SourceContext": ""
            },
            "Vid": ""
        },
        "FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
        "ProcedureTaskId": "",
        "ReviewAudioVideoTaskId":"",
        "SessionContext": "",
        "SessionId": ""
    }
}
```

可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。

```
{
    "Response": {
        "EventSet": [
            {
                "EventHandle": "EventHandleX",
                "EventType": "PullComplete",
                "PullCompleteEvent": {
                    "TaskId": "125676836723-Pull-f5ac8127b3b6b85cdc13f237c6005d8",
                    "Status": "FINISH",
                    "ErrCode": 0,
                    "Message": "SUCCESS",
                    "FileId": "14508071098244959037",
                    "MediaBasicInfo": {
                        "Name": "动物世界",
                        "Description": "",
                        "CreateTime": "2019-01-09T16:36:22Z",
                        "UpdateTime": "2019-01-09T16:36:24Z",
                        "ExpireTime": "9999-12-31T23:59:59Z",
                        "ClassId": 0,
                        "ClassName": "其他",
                        "ClassPath": "其他",
                        "CoverUrl": "",
                        "Type": "mp4",
                        "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/
                        "TagSet": [ ],
                        "StorageRegion": "ap-guangzhou-2",
                        "SourceInfo": {
                            "SourceType": "Upload",
```

```
                            "SourceContext": ""
                        },
                        "Vid": ""
                    },
                    "FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.m
                    "ProcedureTaskId": "",
                    "ReviewAudioVideoTaskId":"",
                    "SessionContext": "",
                    "SessionId": ""
                }
            }
        ]
    }
}
```

视频删除完成

最近更新时间：2023-03-13 11:41:15

事件名称

FileDeleted

事件说明

当 App 配置了事件通知，并且在视频删除后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 FileDeleteTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。

```
{
    "EventType":"FileDeleted",
    "FileDeleteEvent":{
        "FileIdSet":[
            "24961954183381008"
        ],
        "FileDeleteResultInfo":[
            {
                "FileId":"24961954183381008",
                "DeleteParts":[
                    {
                        "Type":"TranscodeFiles",
                        "Definition":0
                    }
                ]
            }
        ]
    }
}
```

可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。

```
{
    "Response":{
        "EventSet":[
            {
                "EventHandle":"EventHandle.N",
                "EventType":"FileDeleted",
                "FileDeleteEvent":{
                    "FileIdSet":[
                        "24961954183381008"
                    ],
                    "FileDeleteResultInfo":[
                        {
                            "FileId":"24961954183381008",
                            "DeleteParts":[
                                {
                                    "Type":"TranscodeFiles",
                                    "Definition":0
                                }
                            ]
                        }
                    ]
                }
            }
        ],
        "RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

任务流状态变更

最近更新时间：2023-03-13 11:41:15

事件名称

ProcedureStateChanged

事件说明

当 App 配置了事件通知后，在任务流的状态发生变更后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 ProcedureTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。

```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-475b72xxxcb177t1",
        "Status":"FINISH",
        "ErrCode":0,
        "Message":"",
        "FileId":"5285890784246869930",
        "FileName":"动物世界",
        "FileUrl":"https://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
        "MetaData":{
            "AudioDuration":59.990001678467,
            "AudioStreamSet":[
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
```

```
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
            "VideoStreamSet":[
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
                    "Height":480,
                    "Width":640
                }
            ],
            "Width":640
        },
        "MediaProcessResultSet":[
            {
                "Type":"Transcode",
                "TranscodeTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"SUCCESS",
                    "Input":{
                        "Definition":20
                    },
                    "Output":{
                        "Url":"https://1256768367.vod2.myqcloud.com/xxx/xxx/v.f20.m
                        "Size":4189073,
                        "Container":"mov,mp4,m4a,3gp,3g2,mj2",
                        "Height":480,
                        "Width":640,
                        "Bitrate":552218,
                        "Md5":"eff7031ad7877865f9a3240e9ab165ad",
                        "Duration":60.04700088501,
                        "VideoStreamSet":[
                            {
                                "Bitrate":503727,
                                "Codec":"h264",
                                "Fps":24,
                                "Height":480,
                                "Width":640
                            }
                        ],
                        "AudioStreamSet":[
                            {
                                "Bitrate":48491,
```

```
                                "Codec":"aac",
                                "SamplingRate":44100
                            }
                        ],
                        "Definition":0
                    }
                }
            },
            {
                "Type":"CoverBySnapshot",
                "CoverBySnapshotTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"SUCCESS",
                    "Input":{
                        "Definition":10,
                        "PositionType":"Time",
                        "PositionValue":0
                    },
                    "Output":{
                        "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx
                    }
                }
            }
        ]
    }
}
```


可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。


```
{
    "Response": {
        "EventSet": [
            {
                "EventHandle": "EventHandleX",
                "EventType": "ProcedureStateChanged",
                "ProcedureStateChangeEvent": {
                    "TaskId": "1256768367-Procedure-475b72xxxcb177t1",
                    "Status": "FINISH",
                    "FileId": "5285890784246869930",
                    "FileName": "动物世界",
                    "FileUrl": "https://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.mp
                    "MetaData": {
                        "AudioDuration": 59.990001678467,
                        "AudioStreamSet": [{
```

```
                "Bitrate": 383854,
                "Codec": "aac",
                "SamplingRate": 48000
            }],
            "Bitrate": 1021028,
            "Container": "mov,mp4,m4a,3gp,3g2,mj2",
            "Duration": 60,
            "Height": 480,
            "Rotate": 0,
            "Size": 7700180,
            "VideoDuration": 60,
            "VideoStreamSet": [{
                "Bitrate": 637174,
                "Codec": "h264",
                "Fps": 23,
                "Height": 480,
                "Width": 640
            }],
            "Width": 640
        },
        "MediaProcessResultSet": [{
                "Type": "Transcode",
                "TranscodeTask": {
                    "Status": "SUCCESS",
                    "ErrCode": 0,
                    "Message": "SUCCESS",
                    "Input": {
                        "Definition": 20
                    },
                    "Output": {
                        "Url": "https://1256768367.vod2.myqcloud.com/xx
                        "Size": 4189073,
                        "Container": "mov,mp4,m4a,3gp,3g2,mj2",
                        "Height": 480,
                        "Width": 640,
                        "Bitrate": 552218,
                        "Md5": "eff7031ad7877865f9a3240e9ab165ad",
                        "Duration": 60.04700088501,
                        "VideoStreamSet": [{
                            "Bitrate": 503727,
                            "Codec": "h264",
                            "Fps": 24,
                            "Height": 480,
                            "Width": 640
                        }],
                        "AudioStreamSet": [{
                            "Bitrate": 48491,
```

```
                                    "Codec": "aac",
                                    "SamplingRate": 44100
                                }],
                                "Definition": 0
                            }
                        }
                    },
                    {
                        "Type": "CoverBySnapshot",
                        "CoverBySnapshotTask": {
                            "Status": "SUCCESS",
                            "ErrCode": 0,
                            "Message": "SUCCESS",
                            "Input": {
                                "Definition": 10,
                                "PositionType": "Time",
                                "PositionValue": 0
                            },
                            "Output": {
                                "CoverUrl": "http://1256768367.vod2.myqcloud.co
                            }
                        }
                    }
                ]
            }
        }
    ],
    "RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
  }
}
```

视频编辑完成

最近更新时间：2023-01-05 11:11:43

事件名称

EditMediaComplete

事件说明

当 App 配置了事件通知，并且在编辑视频完成后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 EditMediaTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。


```
{
"EventType":"EditMediaComplete",
"EditMediaCompleteEvent":{
"TaskId":"1256768367-EditMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status":"FINISH",
"ErrCode":0,
"Message":"SUCCESS",
"Input":{
"InputType":"File",
"FileInfoSet":[
{
"FileId":"24961954183381008",
"StartTimeOffset":0,
"EndTimeOffset":0
},
{
"FileId":"24961954183381009",
"StartTimeOffset":0,
"EndTimeOffset":0
},
```

```
{
"FileId":"24961954183381010",
"StartTimeOffset":0,
"EndTimeOffset":0
}
]
},
"Output":{
"FileType":"mp4",
"FileId":"24961954183923290",
"FileUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/f0.mp4"
},
"ProcedureTaskId":"",
"ReviewAudioVideoTaskId":""
}
}
```


可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。


```
{
"Response": {
"EventSet": [
{
"EventHandle": "EventHandle.N",
"EventType": "EditMediaComplete",
"EditMediaCompleteEvent": {
"TaskId": "EditMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status": "FINISH",
"ErrCode": 0,
"Message": "SUCCESS",
"Input": {
"InputType": "File",
"FileInfoSet": [
{
"FileId": "24961954183381008",
"StartTimeOffset": 0,
"EndTimeOffset": 0
},
{
"FileId": "24961954183381009",
"StartTimeOffset": 0,
"EndTimeOffset": 0
},
{
```

```
"FileId": "24961954183381010",
"StartTimeOffset": 0,
"EndTimeOffset": 0
}
]
},
"Output": {
"FileType": "mp4",
"FileId": "24961954183923290",
"FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/f0.mp4"
},
"ProcedureTaskId": "",
"ReviewAudioVideoTaskId": ""
}
}
],
"RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

视频合成完成

最近更新时间：2022-10-26 17:20:28

事件名称

ComposeMediaComplete

事件说明

当 App 配置了事件通知，并且在合成视频完成后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 ComposeMediaTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。

```
{
"EventType": "ComposeMediaComplete",
"ComposeMediaCompleteEvent": {
"TaskId": "1256768367-ComposeMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status": "FINISH",
"ErrCode": 0,
"Message": "SUCCESS",
"Input": {
"Tracks": [{
"Type": "Video",
"TrackItems": [{
"Type": "Video",
"SourceMedia": "5285485487985271487",
"AudioOperations": [{
"Type": "Volume",
"VolumeParam": {
"Mute": 1
}
}]
}]
```

```
},
{
"Type": "Audio",
"TrackItems": [{
"Type": "Empty",
"EmptyItem": {
"Duration": 5
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271488",
"Duration": 15
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271489",
"SourceMediaStartTime": 2,
"Duration": 14
}
}
]
}
],
"Output": {
"FileName": "视频合成效果测试",
"Container": "mp4"
}
},
"Output": {
"FileType": "mp4",
"FileId": 5285485487985271490,
"FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4"
}
}
}
```

可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。

```
{
"Response":{
```

```
"EventSet":[
{
"EventHandle":"EventHandle.N",
"ComposeMediaCompleteEvent":{
"TaskId":"1256768367-ComposeMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status":"FINISH",
"ErrCode":0,
"Message":"SUCCESS",
"Input":{
"Tracks":[
{
"Type":"Video",
"TrackItems":[
{
"Type":"Video",
"SourceMedia":"5285485487985271487",
"AudioOperations":[
{
"Type":"Volume",
"VolumeParam":{
"Mute":1
}
}
]
}
]
},
{
"Type":"Audio",
"TrackItems":[
{
"Type":"Empty",
"EmptyItem":{
"Duration":5
}
},
{
"Type":"Audio",
"AudioItem":{
"SourceMedia":"5285485487985271488",
"Duration":15
}
},
{
"Type":"Audio",
"AudioItem":{
"SourceMedia":"5285485487985271489",
```

```
"SourceMediaStartTime":2,
"Duration":14
}
}
]
}
],
"Output":{
"FileName":"视频合成效果测试",
"Container":"mp4"
}
},
"Output":{
"FileType":"mp4",
"FileId":5285485487985271490,
"FileUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4"
}
}
}
],
"RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

视频取回完成

最近更新时间：2023-03-13 11:41:15

事件名称

RestoreMediaComplete

事件说明

当 App 配置了事件通知，并且将归档或深度归档的媒体文件解冻或取回后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 RestoreMediaTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到如下形式的 HTTP 请求。

```
{
    "EventType":"RestoreMediaComplete",
    "RestoreMediaCompleteEvent":{
        "FileId":"24961954183381008",
        "OriginalStorageClass":"ARCHIVE",
        "TargetStorageClass":"STANDARD",
        "RestoreTier":"Standard",
        "RestoreDay":0,
        "Status":0,
        "Message":"Restore success!"
    }
}
```

可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答。

```
{
    "Response":{
        "EventSet":[
            {
                "EventHandle":"EventHandle.N",
```

```
                "EventType":"RestoreMediaComplete",
                "RestoreMediaCompleteEvent":{
                    "FileId":"24961954183381008",
                    "OriginalStorageClass":"ARCHIVE",
                    "TargetStorageClass":"STANDARD",
                    "RestoreTier":"Standard",
                    "RestoreDay":0,
                    "Status":0,
                    "Message":"Restore success!"
                }
            }
        ],
        "RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

音视频审核完成

最近更新时间：2022-10-13 15:08:47

事件名称

ReviewAudioVideoComplete

事件说明

当 App 配置了事件通知，并且在音视频审核完成后，App 后台即可通过“普通回调”或“可靠回调”的方式获取该事件通知。事件通知内容为 ReviewAudioVideoTask 结构。

示例

普通回调

如果选择普通回调模式，则回调 URL 会接收到来自云点播的 HTTP 请求。请求采用 POST 方法，请求内容在 BODY 中，如下所示（省略了值为 null 的字段）。


```
{
"EventType": "ReviewAudioVideoComplete",
"ReviewAudioVideoCompleteEvent": {
"TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
"Status": "FINISH",
"ErrCodeExt": "",
"Message": "SUCCESS",
"Input": {
"FileId": "387702130626135215"
},
"Output": {
"Suggestion": "block",
"Label": "porn",
"Form": "Image",
"SegmentSet": [
{
"StartTimeOffset": 0,
"EndTimeOffset": 1,
"Confidence": 99,
"Suggestion": "block",
```

```
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 1,
"EndTimeOffset": 2,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 2,
"EndTimeOffset": 3,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 3,
"EndTimeOffset": 4,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 4,
"EndTimeOffset": 5,
"Confidence": 99,
```

```
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 5,
"EndTimeOffset": 6,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 6,
"EndTimeOffset": 7,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 7,
"EndTimeOffset": 8,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 8,
"EndTimeOffset": 9,
```

```
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 9,
"EndTimeOffset": 10,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
}
],
"SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtranssgp251000800/0f9bd2b0-34a8-4642-f481-001894d93019.txt",
"SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
},
"SessionContext": "",
"SessionId": ""
}
}
```


可靠回调

如果选择可靠回调模式，调用 拉取事件通知 API 会接收到如下形式的 HTTP 应答（省略了值为 null 的字段）。


```
{
"Response": {
"EventSet": [
{
"EventHandle": "EventHandle.N",
"EventType": "ReviewAudioVideoComplete",
"ReviewAudioVideoCompleteEvent": {
"TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
"Status": "FINISH",
"ErrCodeExt": "",
"Message": "SUCCESS",
```

```
"Input": {
"FileId": "387702130626135215"
},
"Output": {
"Suggestion": "block",
"Label": "porn",
"Form": "Image",
"SegmentSet": [
{
"StartTimeOffset": 0,
"EndTimeOffset": 1,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 1,
"EndTimeOffset": 2,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 2,
"EndTimeOffset": 3,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 3,
"EndTimeOffset": 4,
```

```
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 4,
"EndTimeOffset": 5,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 5,
"EndTimeOffset": 6,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 6,
"EndTimeOffset": 7,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 7,
```

```
"EndTimeOffset": 8,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 8,
"EndTimeOffset": 9,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 9,
"EndTimeOffset": 10,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
}
],
"SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtranssgp251000800/0f9bd2b0-34a8-4642-f481-001894d93019.txt",
"SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
},
"SessionContext": "",
"SessionId": ""
}
}
],
"RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

媒体分发播放
视频播放综述

最近更新时间：2022-09-15 17:35:35

云点播支持多种方式，播放上传及转码后的视频。其中，播放主要分为短视频播放、长视频播放和视频加密播放几个场景。

短视频播放

短视频，通常指视频时长5分钟以内的视频。主要有：

- 短视频社交平台（微视、快手、抖音）中分享的视频。
- 电商购物平台（京东、拼多多）中的商品宣传视频。
- 微信公众号、自媒体中分享的短片。

长视频播放

长视频通常是由专业机构制作，并通过视频网站发布的视频。主要有：

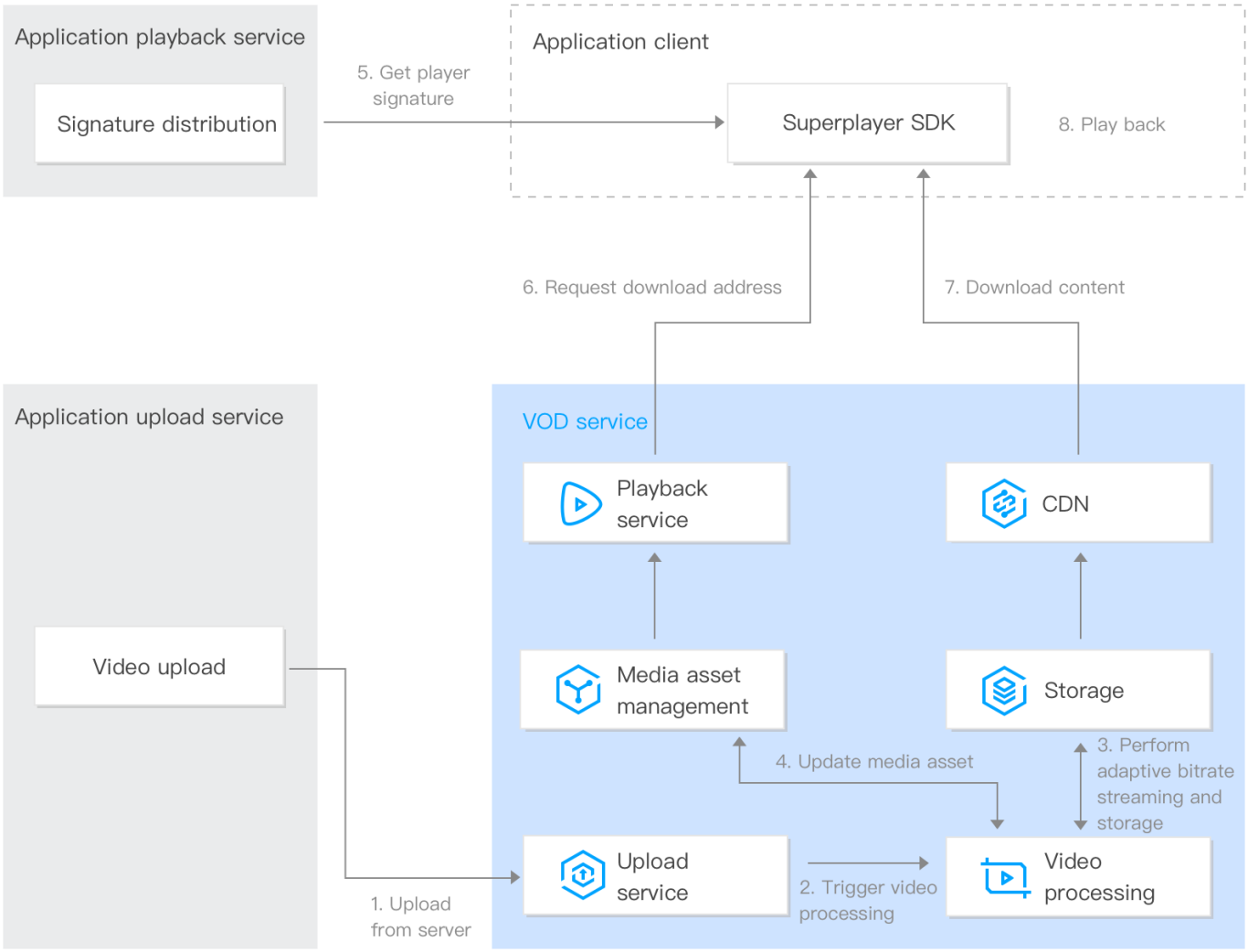
- 视频媒体平台（腾讯视频、优酷、爱奇艺）发布的独播剧、综艺节目。
- 在线教育网站（腾讯课堂、企鹅辅导）的课程视频。
- 网络电视平台（CNTV、芒果 TV）的电视节目回看视频。

视频加密播

视频加密属于长视频场景下的一个细分场景。是视频平台自有的独播剧、网络课程等具有版权的视频，采取加密防止视频被非法下载和传播的措施。

播放架构

针对各类视频播放场景，点播建议使用 **播放器 SDK**，播放转自适应码流后的输出视频。播放的整体架构流程如下：



1. **服务端上传**：业务后台通过控制台、服务端 API 等方式，将视频上传到云点播。
2. **触发视频处理**：上传视频的同时，指定转自适应码流。上传后，视频开始视频处理。
3. **转自适应码流并写入存储**：视频转自适应码流后，输出的视频内容被写入到点播的存储中。
4. **更新媒资**：转自适应码流的视频信息，被写入到媒资管理模块。
5. **派发签名**：业务后台根据播放器签名计算规则，派发播放签名。
6. **请求下载地址**：播放器指定要播放的视频 FileId 后，会从点播的播放服务获取视频的下载地址。
7. **下载内容**：播放器通过下载地址，从点播 CDN 下载内容。
8. **播放**：播放器开始播放自适应码流输出。

文档指引

- 播放器 SDK 支持的功能，请参见 功能说明，集成方式请参见 SDK 下载。
- 为了帮助您快速接入点播的播放器，我们为您提供了播放器 SDK 接入指引，以示例的方式为您讲解接入步骤。

- 对于视频加密播放场景，云点播分别在 视频加密综述 和 视频加密接入指引 中详细介绍了视频加密的原理和接入方式。

添加域名

最近更新时间：2023-08-31 16:07:28

为什么需要添加域名？

当您仍在使用云点播默认分发域名进行媒体内容加速分发时，添加属于您自己的域名用于分发，保障业务更加灵活的同时，还可规避云点播默认分发域名被封禁导致业务分发受影响的风险。

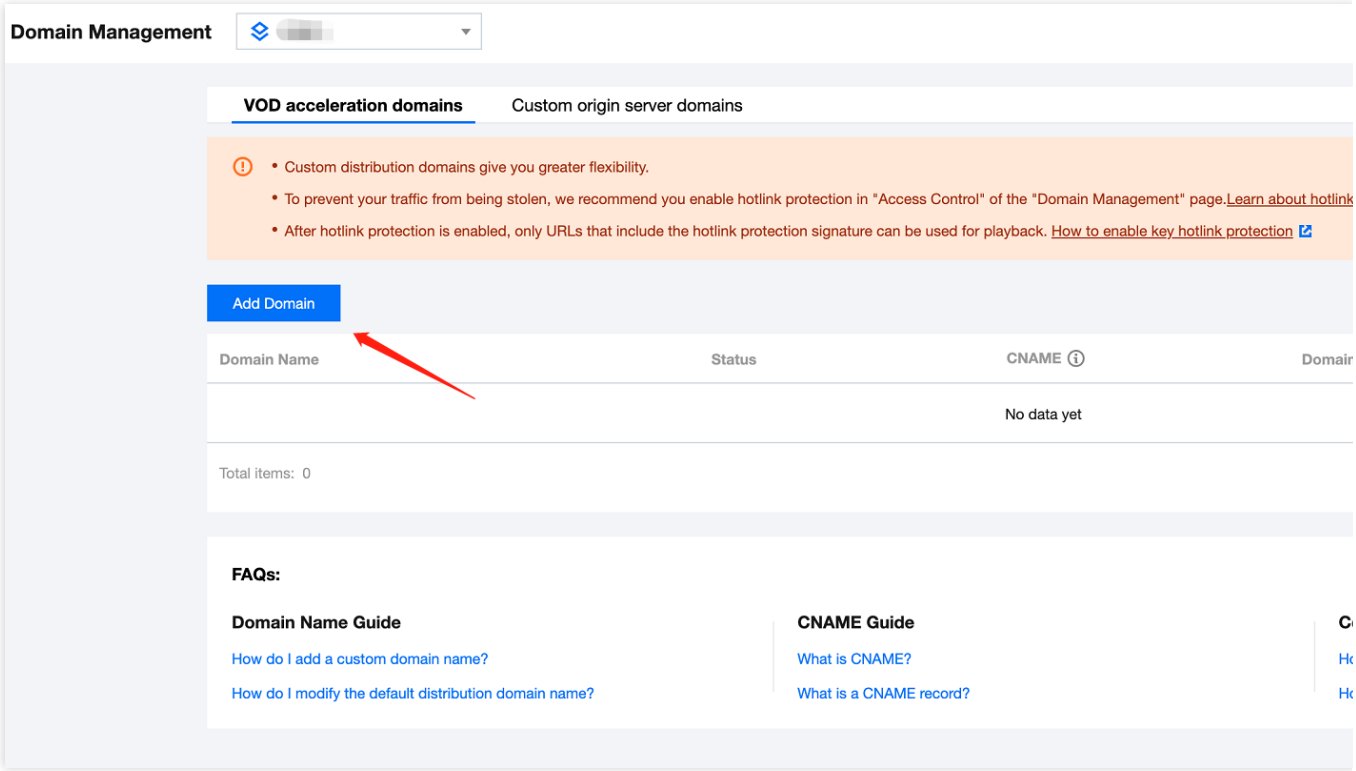
准备工作

准备一个已完成域名注册可用于视频点播加速的域名，例如：example.com

方法一：通过云点播控制台添加域名

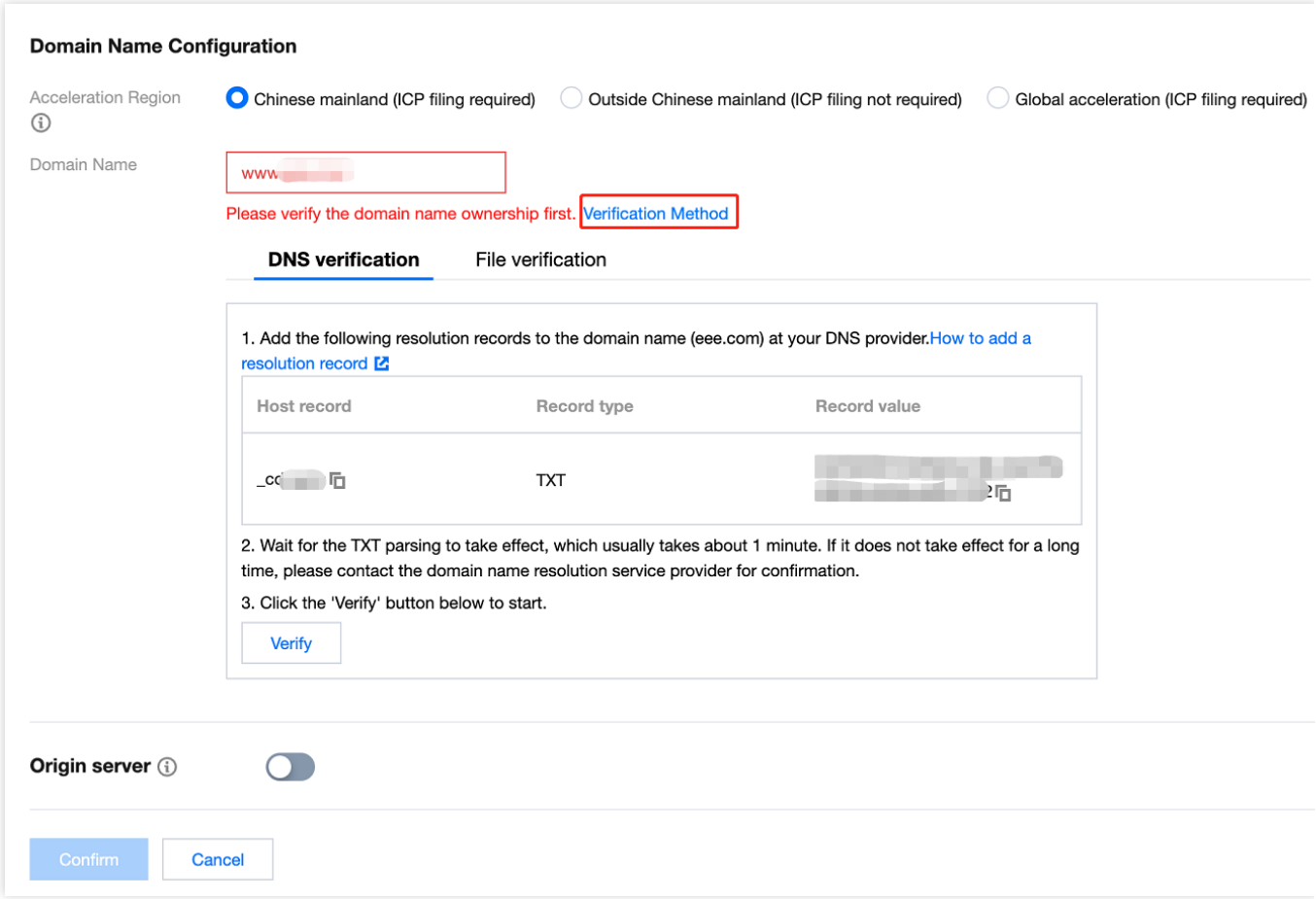
1.添加域名

云点播控制台-分发播放设置-域名管理-添加域名 或 添加自定义源站加速域名，输入域名后，需进行域名归属解析校验。



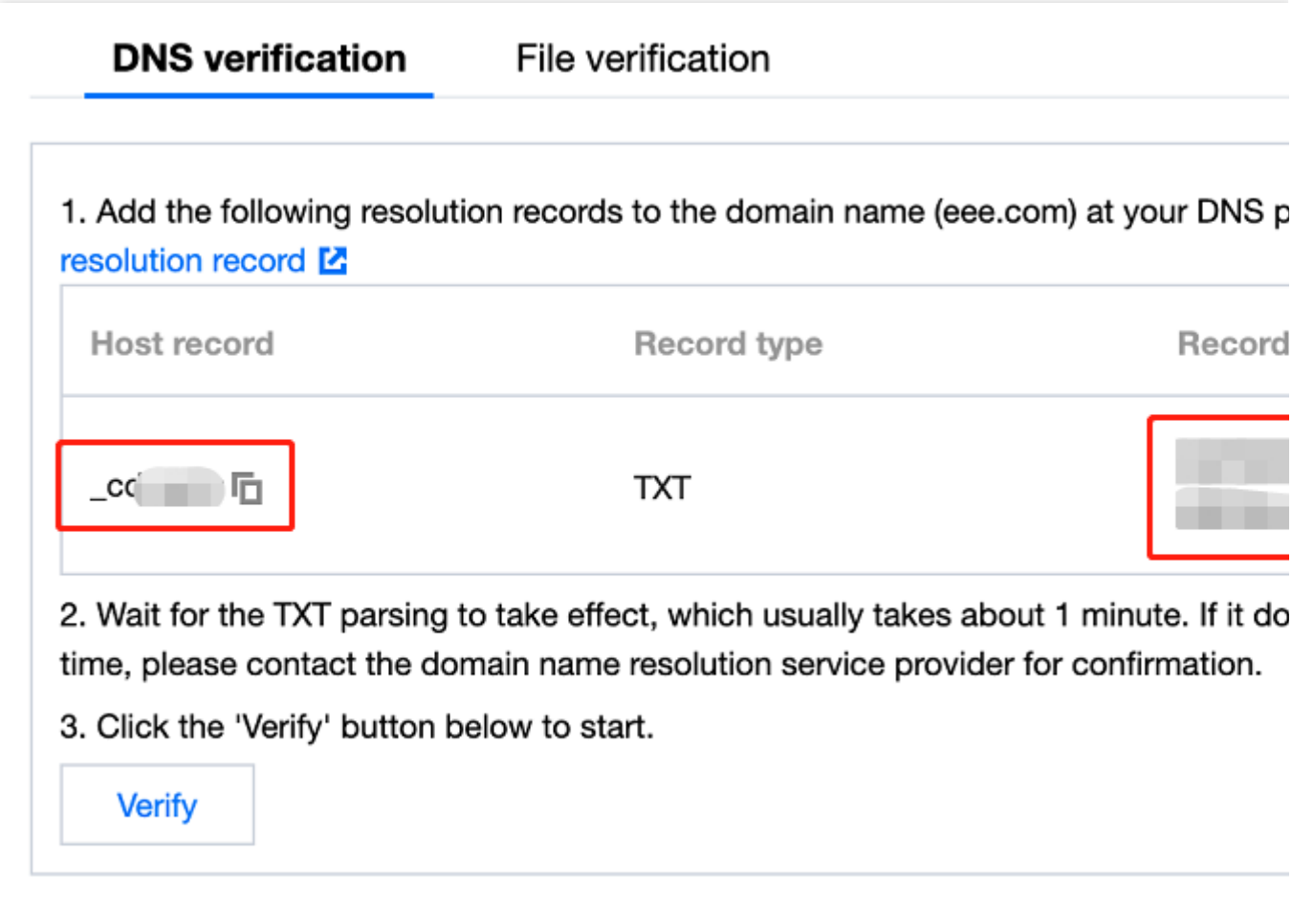
2. DNS 解析验证

（1）单击验证方法



（2）验证方法中，默认为 DNS 解析验证。

使用 DNS 解析验证的方式，需要您前往该域名的解析服务商，在主域名下添加一个主机记录值为 `_cdnauth` 的 TXT 记录。



注意：

无论您需要新增的域名为 `c.b.a.example.com` 、 `*.example.com` 或 `test.example.com` ，多级域名下主机记录值仍应添加在主域名下，例如：添加的域名是 `c.b.a.example.com` ，需要新增一条解析记录为 `_cdnauth.example.com` 即可。

腾讯云 DNS 解析添加方法参考：

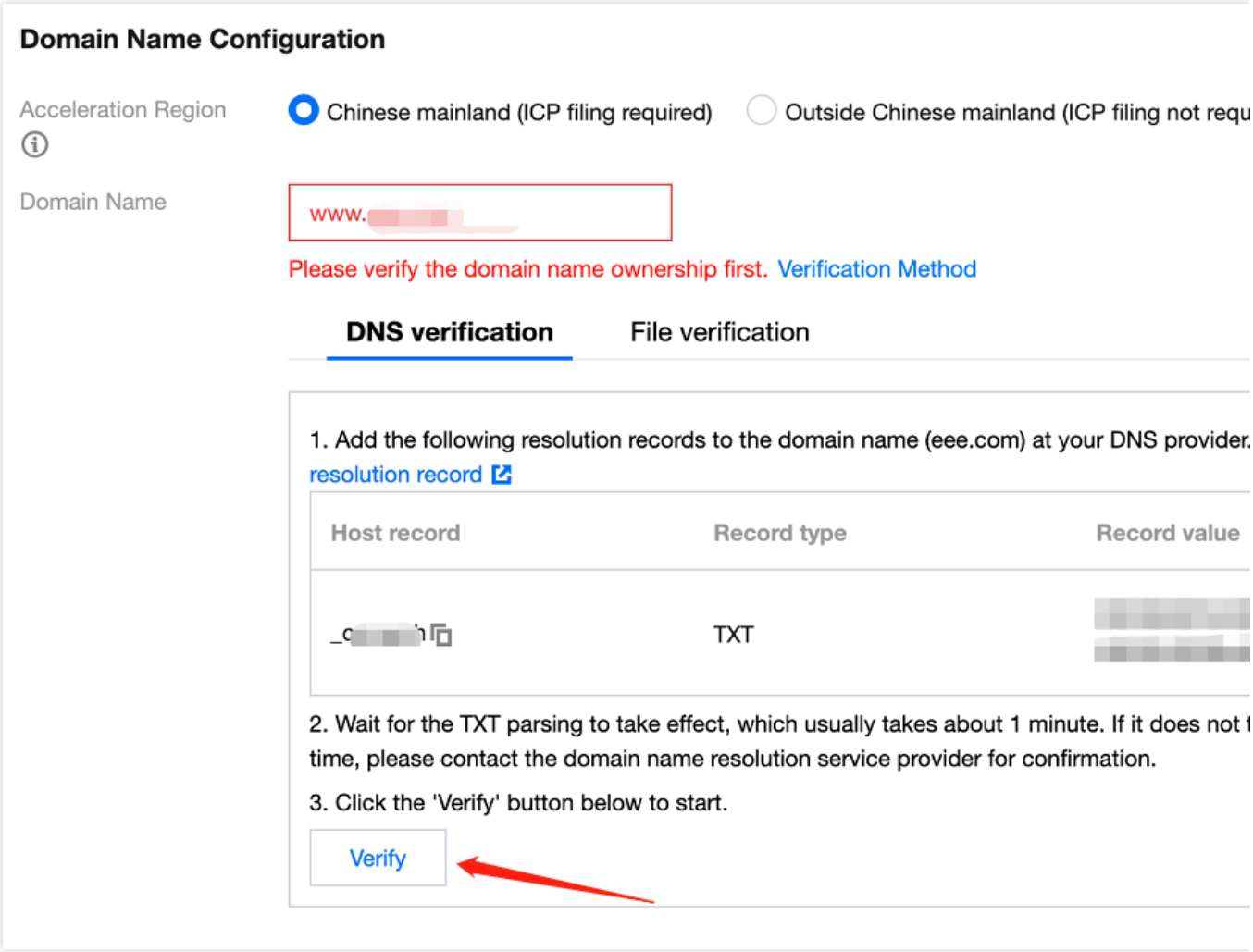
如果您的解析服务商在腾讯云 DNSPod 上，可进入 DNS 解析 DNSPod 控制台，找到该域名并单击解析，添加一条记录类型为 TXT 的 DNS 记录，主机记录填写为 `_cdnauth` ，记录类型选择为 TXT，记录值填写为腾讯云 CDN 提供的记录值，其余选项按照默认参数填写即可。

阿里云 DNS 解析添加方法参考：

如果您的解析服务商在阿里云，可以在阿里云的域名解析控制台内找到该域名，然后点击右侧的解析设置，记录类型选择为TXT，填写腾讯云 CDN 提供的主机记录、记录值，其余保持为默认参数填写即可。

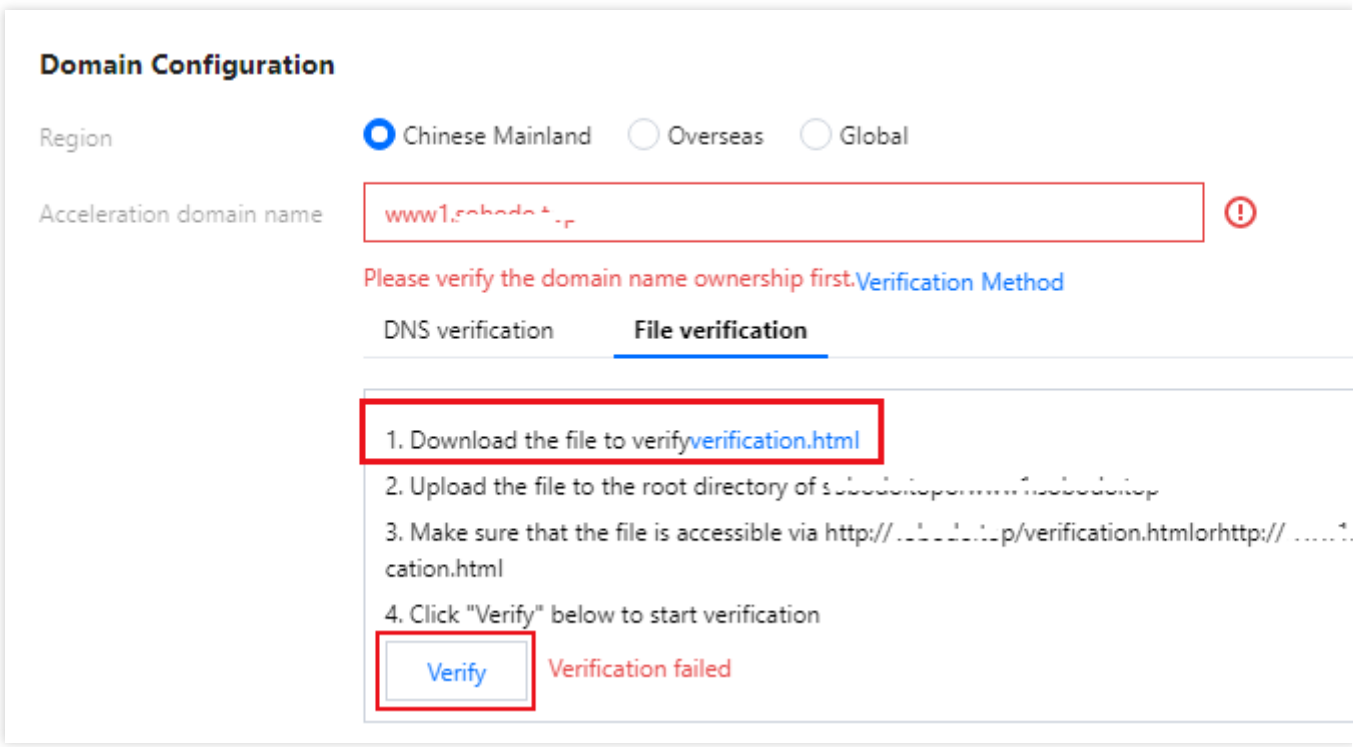
（3）完成域名归属验证

添加完解析记录后，等待 TXT 记录值生效，生效后，您可点击下方的验证按钮，即可完成域名归属校验；如果验证失败，请确认当前 TXT 记录值在域名解析服务商内是否已生效或是否填写了正确的 TXT 记录值；如何检测 TXT 记录是否生效？



3. 文件验证

（1）在验证方法内，选择文件验证的方式



（2）单击下载文件 verification.html

将该文件上传至您主域名的服务器（例如您的 CVM、COS、阿里 ECS、阿里 OSS 等）根目录下，例如：当前添加的域名为 `test.example.com` ，您需要将该文件上传至 `example.com` 的根目录下或 `test.example.com` 的根目录下。

注意：

文件验证的方式已支持通过将文件上传至子域名进行验证，DNS 验证暂不支持，如您需要通过子域名操作域名归属验证，请使用文件验证的方式。

（3）完成域名归属校验

确保可通过 `http://example.com/verification.html` 或 `http://test.example.com/verification.html` 访问到该文件后，即可单击验证按钮进行验证。如果文件内的记录值与我们提供的记录值是一致的，即可验证通过；如果验证失败，请确保上述文件链接可访问，并且您上传的文件为正确文件，可通过访问文件的链接与所下载的文件进行比对是否一致；

具体场景操作示例：

需加速的域名： `a.test.com` ，源站为对象存储 COS：

（1）将生成的验证文件 verification.html 上传到对象存储 COS 的根目录。

（2）在加速域名的解析处增加一条 CNAME 记录，将其指向 COS 源站域名。

（3）完成上述操作后，确认可通过 http(https)://加速域名/verification.html 访问到验证文件。单击验证按钮，即可通过验证。

方法二：通过API添加域名

1.首先，调用 CreateDomainVerifyRecord 接口，生成一条针对该接入域名的 TXT 解析记录。

```
{
  "Response": {
    "DNSVerifyInfo": {
      "Record": "2023082515502104ad6d69c54862dcc99e226349af3440",
      "RecordType": "TXT",
      "SubDomain": "_cdnauth"
    },
    "FileVerifyInfo": {
      "FileVerifyDomains": [
        "123.com"
      ],
      "FileVerifyName": "verification.html",
      "FileVerifyUrl": "http://123.com/verification.html"
    },
    "RequestId": "10645a01-c728-4fb5-baa8-09d21e1090e3"
  }
}
```

2.通过在域名解析处（如DNSPOD）配置上这条TXT解析记录。

3.调用 VerifyDomainRecord 接口，验证该解析是否生效。

```
{
  "Response": {
    "RequestId": "48d4442e-cda6-4404-af2a-467cc5891079",
    "Result": true
  }
}
```

4.验证成功后，即可调用 添加点播加速域名 接口完成域名添加

常见问题

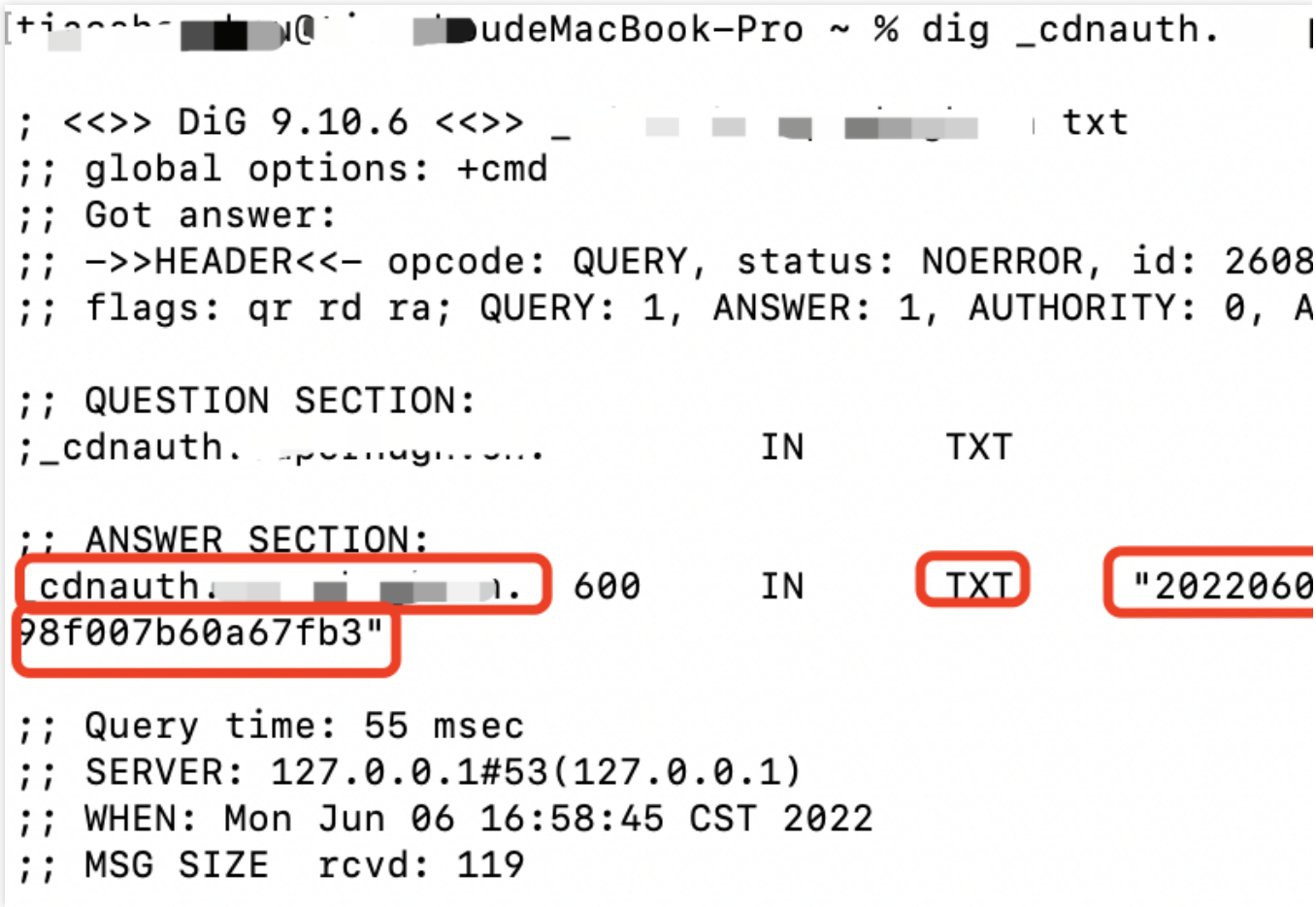
如何手动检测域名归属校验的 TXT 记录值是否生效？

Windows 系统示例：

例如接入域名为test.example.com，可以在系统内打开 cmd 命令界面内，输入nslookup -qt=txt _cdnauth.example.com，根据当前的 TXT 结果，可以查看解析记录是否生效或是否正确。

Linux/Mac 系统示例：

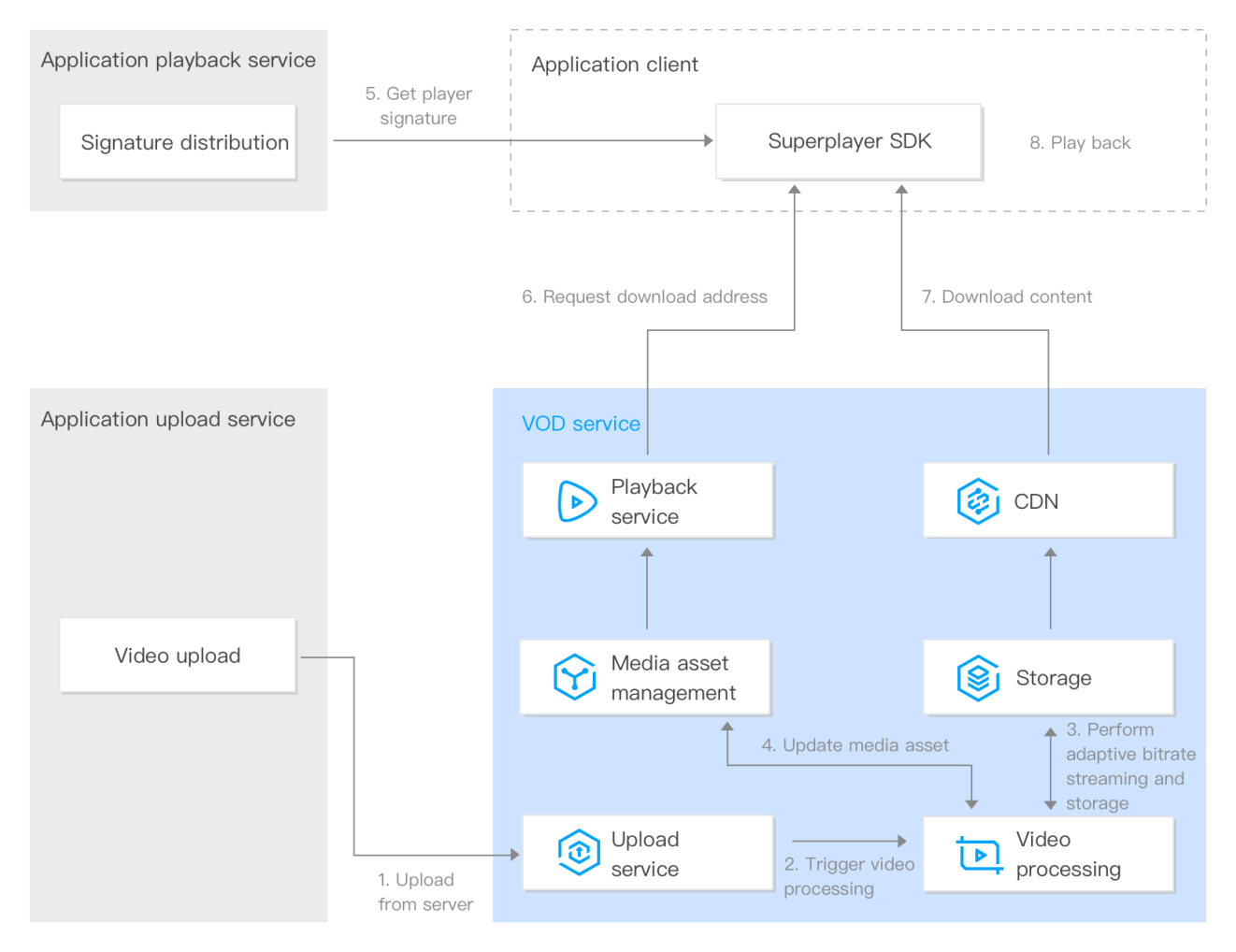
例如接入域名为test.example.com，可以在命令界面内，输入dig _cdnauth.example.com txt，根据当前的 TXT 结果，可以查看解析记录是否生效或是否正确。



播放器签名

最近更新时间：2024-08-26 15:23:26

播放器签名，用于 App 播放服务对终端的授权播放。如下图步骤 6 所示，若 App 播放服务允许终端播放，则派发一个合法的签名。终端在签名有效时间内可以播放视频内容。



下面，将介绍播放器签名的参数组成和生成规则。

签名参数

参数名称	必选	类型	说明
appId	是	Integer	点播应用 appId。
fileId	是	String	点播文件 ID。

contentInfo	是	Object	对应点播文件 ID 播放的具体内容，为 ContentInfo 类型，可播放下列三种中的一种： 转自适应码流 的输出音视频，可以是未加密或加密的。 转码 的输出音视频。 上传 的原始音视频。
currentTimeStamp	是	Integer	派发签名当前 Unix 时间戳。
expireTimeStamp	否	Integer	派发签名到期 Unix 时间戳，不填表示不过期。
urlAccessInfo	否	Object	播放链接访问配置参数，包括 Key 防盗链 配置、播放域名与协议参数，为 UrlAccessInfo 类型。
drmLicenseInfo	否	Object	DRM License 配置参数，为 DrmLicenseInfo 类型 。

ContentInfo 类型

参数名称	必选	类型	说明
audioVideoType	是	String	播放的音视频类型，可选值： RawAdaptive：未加密的 转自适应码流 输出。 ProtectedAdaptive：私有加密或 DRM 保护的 转自适应码流 输出。 Transcode：转码 后输出。 Original：上传 的原始音视频。
rawAdaptiveDefinition	否	Integer	允许输出的未加密的 转自适应码流模板 ID，仅当 audioVideoType 为 RawAdaptive 该参数必填且有效。
drmAdaptiveInfo	否	Object	允许输出的加密保护的 转自适应码流模板 ID，仅当 audioVideoType 为 ProtectedAdaptive 该参数必填且有效，为 DRMAdaptiveInfo 类型。
transcodeDefinition	否	Integer	允许输出的 转码模板 ID，仅当 audioVideoType 为 Transcode 该参数必填且有效。
imageSpriteDefinition	否	Integer	用于进度条预览的 雪碧图模板 ID。
resolutionNames	否	Array of Object	播放器对不于不同分辨率的子流展示名字，为 ResolutionNameInfo 类型 数组。不填或者填空数组则使用默认配置： MinEdgeLength：240，Name：240P。 MinEdgeLength：480，Name：480P。 MinEdgeLength：720，Name：720P。 MinEdgeLength：1080，Name：1080P。 MinEdgeLength：1440，Name：2K。

			MinEdgeLength：2160，Name：4K。 MinEdgeLength：4320，Name：8K。

DRMAdaptiveInfo 类型

参数名称	必选	类型	说明
privateEncryptionDefinition	否	Integer	保护类型 DrmType 为 SimpleAES 的 转自适应码流模板 ID。
widevineDefinition	否	Integer	保护类型 DrmType 为 Widevine 的 转自适应码流模板 ID。
fairPlayDefinition	否	Integer	保护类型 DrmType 为 FairPlay 的 转自适应码流模板 ID。

ResolutionNameInfo 类型

参数名称	必选	类型	说明
MinEdgeLength	是	Integer	视频短边长度，单位：像素。
Name	是	String	展示名字。

UrlAccessInfo 类型

参数名称	必选	类型	说明
t	否	String	16进制字符串，表示链接的过期时间。 具体含义和取值参见 防盗链参数 中的 t 参数。 不填表示不过期。
exper	否	Integer	试看时长，单位为秒，以十进制表示。 如果要指定试看时长，时长必须不小于30秒。 具体含义和取值参见 防盗链参数 中的 exper 参数。
rlimit	否	Integer	最多允许多少个不同 IP 的终端播放，以十进制表示。 具体含义和取值参见 防盗链参数 中的 rlimit 参数。
us	否	String	链接标识，用户增强链接的唯一性。 具体含义和取值参见 防盗链参数 中的 us 参数。
domain	否	String	播放时使用的域名。不填或者填 Default，表示使用 默认分发配置 中的域名。
scheme	否	String	播放时使用的 Scheme。不填或者填 Default，表示使用 默认分发配置

			中的 Scheme。其他可选值： HTTP。 HTTPS。

DrmLicenseInfo 类型

参数名称	必选	类型	说明
persistent	否	String	是否允许终端持久化保存商业级 DRM 播放许可证。取值范围： ON：允许持久化保存。 OFF：不允许持久化保存。 默认取值为 OFF。
rentalDuration	否	Integer	当 persistent 为 ON 时，商业级 DRM 播放许可证允许被持久化保存的时长，单位为秒，不填表示不限时长。
forceL1TrackTypes	否	Array of String	当使用 Widevine 时，要求终端必须使用 L1 安全级别处理的 Track 类型。其中，未指定的 Track 类型默认使用 L3 安全级别处理。取值范围如下： AUDIO：音频子流。 SD：短边小于720的子流。 HD：短边大于等于720并小于2160的子流。 UHD1：短边大于等于2160并小于4320的子流。 UHD2：短边大于等于4320的子流。

说明：

如果您使用了新增 应用，则 appId 参数需要填新增应用的 AppId。

签名参数中的 `t` 、 `exper` 、 `rlimit` 、 `us` 的含义和取值，与 防盗链参数 中的同名参数完全一致。

签名计算

点播播放器签名采用 JWT（JSON Web Token），一种由 Header、PayLoad 和 Key 计算并组合得到的数字令牌。

Header

Header 为 JSON 格式，表示 JWT 使用的算法信息，固定使用如下内容：

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PayLoad

Payload 为 JSON 格式，是播放器签名参数的内容，例如：

```
{
  "appId": 1255566655,
  "fileId": "4564972818519602447",
  "contentInfo": {
    "audioVideoType": "RawAdaptive",
    "rawAdaptiveDefinition": 10,
    "imageSpriteDefinition": 10
  },
  "currentTimeStamp": 1663064276,
  "expireTimeStamp": 1663294210,
  "urlAccessInfo": {
    "t": "6323e6b0",
    "rlimit": 3,
    "us": "72d4cd1101"
  }
}
```

Key

Key 是计算签名时使用的密钥，这里使用 默认分发配置 中的 `播放密钥` 。

计算公式

1. 计算 Signature：

`Signature = HMACSHA256(base64UrlEncode(Header) + "." + base64UrlEncode(Payload), Key)`

2. 计算 Token：

`Token = base64UrlEncode(Header) + '.' + base64UrlEncode(Payload) + '.' + base64UrlEncode(Signature)`

最终得到的 Token，即为点播播放器签名。

说明：

HMACSHA256 请参见 RFC - HMACSHA256。base64UrlEncode 请参见 RFC - base64UrlEncode。

为方便您计算签名以及验证签名，云点播控制台提供了签名生成工具和校验工具：

播放器签名工具 。

计算示例

例如，某用户 appId 是 `1255566655` ，fileId 是 `4564972818519602447` 的视频生成播放器签名，并且：

播放密钥为 `TxtyhLlgo7J3iOADIron` 。

播放器签名的派发时间为 2022-09-13 18:17:56，对应的 Unix 时间是 `1663064276` 。

播放器签名的过期时间为 2022-09-16 10:10:10，对应的 Unix 时间是 `1663294210` 。

防盗链的过期时间为 2022-09-16 11:00:00，对应的 Unix 时间是 `6323e6b0` 。

限制最多允许 3 个不同的 IP 播放 URL。

生成的随机字符串是 `72d4cd1101` 。

则签名步骤如下：

1. Header 的内容为：

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

经过 base64UrlEncode 后的结果是：

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9` 。

2. Payload 的内容为：

```
{
  "appId": 1255566655,
  "fileId": "4564972818519602447",
  "contentInfo": {
    "audioVideoType": "RawAdaptive",
    "rawAdaptiveDefinition": 10,
    "imageSpriteDefinition": 10
  },
  "currentTimeStamp": 1663064276,
  "expireTimeStamp": 1663294210,
  "urlAccessInfo": {
    "t": "6323e6b0",
    "rlimit": 3,
    "us": "72d4cd1101"
  }
}
```

经过 base64UrlEncode 后的结果是：

`eyJhcHBJZCI6MTI1NTU2NjY1NSwiZmlsZUlkIjoiNDU2NDk3MjgxODUxOTYwMjQ0NyIsImNvbnRlbnRJbmZvMSI6eyJhdWRpb1ZpZGVvVHlwZSI6IlJhd0FkYXB0aXZlIiwicmF3QWRhcHRpdmVEZWZpbml0aW9uIjoxMCwiaW1hZ2VTcHJpdGVEZWZpbml0aW9uIjoxMH0sImN1cnJlbnRUaW1lU3RhbXAiOjE2NjMwNjQyNzYsImV4cGlyZVRpbWVTdGFtcCI6MTY2MzI5NDIxMCwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiNjMyM2U2YjAiLCJybGltaXQiOjMsInVzIjoiNzJkNGNkMTEwMSJ9fQ` 。

3. 以播放密钥作为 Key（即 `TxtyhLlgo7J3iOADIron` ）进行 HMAC 计算，Signature 是：

`QFcBX9830ysTzJIyZxoOlRmNb2Gqy2fns9yOfriaDI8` 。

4. 最终 Token 是：

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTI1NTU2NjY1NSwiZmlsZUlkIjoiNDU2NDk3MjgxODUxOTYwMjQ0NyIsImNvbnRlbnRJbmZvMSI6eyJhdWRpb1ZpZGVvVHlwZSI6IlJhd0FkYXB0aXZlIiwicmF3QWRhcHRpdmVEZWZpbml0aW9uIjoxMCwiaW1hZ2VTcHJpdGVEZWZpbml0aW9uIjoxMH0sImN1cnJlbnRUaW1lU3RhbXAiOjE2NjMwNjQyNzYsImV4cGlyZVRpbWVTdGFtcCI6MTY2MzI5NDIxMCwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiNjMyM2U2YjAiLCJybGltaXQiOjMsInVzIjoiNzJkNGNkMTEwMSJ9fQ.QFcBX9830ysTzJIyZxoOlRmNb2Gqy2fns9yOfriaDI8` 。

代码示例

云点播提供了 Python、Java、Go、C#、PHP 和 Node.js 等多种语言的播放器签名的代码示例，具体请参见 播放器签名 - 签名示例。

常见错误

如果您使用播放器签名，播放器 SDK 返回了错误码，较为常见的原因有：

签名计算 KEY 用错，应该使用 默认分发配置 中的 `播放密钥` ，您可自查下是否误用为 KEY 防盗链参数 中的 `KEY` 参数。

签名参数 **填错**，如：

参数类型错误：例如 appId 为整型，错填为 `appId:"125000123"` （字符串型）；又例如 `contentInfo` 里的转码模板参数为整型，错填为 `transcodeDefinition: "14011"` （字符串型）。

参数取值超出有效范围：例如 `contentInfo` 里的播放的音视频类型参数，错填为 `audioVideoType: "Transocde"` （拼写错误，不是有效的枚举值）。

播放器签名示例

最近更新时间：2023-07-14 16:25:21

Python 签名示例

使用 pyjwt 库计算签名，请使用 `pip install pyjwt` 安装。

```
#!/usr/bin/python
#coding=utf-8

import jwt

AppId = 1255566655
FileId = "4564972818519602447"
AudioVideoType = "RawAdaptive"
RawAdaptiveDefinition = 10
ImageSpriteDefinition = 10
CurrentTime = 1546340400
PsignExpire = 1546344000
UrlTimeExpire = "5c2b5640"
PlayKey = "TxtyhLlgo7J3iOADIron"

Original = {
    "appId": AppId,
    "fileId": FileId,
    "contentInfo": {
        "audioVideoType": AudioVideoType,
        "rawAdaptiveDefinition": RawAdaptiveDefinition,
        "imageSpriteDefinition": ImageSpriteDefinition
    },
    "currentTimeStamp": CurrentTime,
    "expireTimeStamp": PsignExpire,
    "urlAccessInfo": {
        "t": UrlTimeExpire
    }
}

Signature = jwt.encode(Original, PlayKey, algorithm='HS256')

print("Original: ", Original)
print("Signature: ", Signature)
```

Java 签名示例

使用 java-jwt 库计算签名。

```
import java.util.*;
import com.auth0.jwt.algorithms.Algorithm;
import com.auth0.jwt.exceptions.JWTCreationException;
import com.auth0.jwt.JWT;

class Main {
    public static void main(String[] args) {
        Integer AppId = 1255566655;
        String FileId = "4564972818519602447";
        String AudioVideoType = "RawAdaptive";
        Integer RawAdaptiveDefinition = 10;
        Integer ImageSpriteDefinition = 10;
        Integer CurrentTime = 1589448067;
        Integer PsignExpire = 1589548067;
        String UrlTimeExpire = "5ebe9423";
        String PlayKey = "TxtyhLlgo7J3iOADIron";
        HashMap<String, Object> urlAccessInfo = new HashMap<String, Object>();
        urlAccessInfo.put("t", UrlTimeExpire);
        HashMap<String, Object> contentInfo = new HashMap<String, Object>();
        contentInfo.put("audioVideoType", AudioVideoType);
        contentInfo.put("rawAdaptiveDefinition", RawAdaptiveDefinition);
        contentInfo.put("imageSpriteDefinition", ImageSpriteDefinition);

        try {
            Algorithm algorithm = Algorithm.HMAC256(PlayKey);
            String token = JWT.create().withClaim("appId", AppId).withClaim("fileId
                    .withClaim("contentInfo", contentInfo)
                    .withClaim("currentTimeStamp", CurrentTime).withClaim("expireTi
                    .withClaim("urlAccessInfo", urlAccessInfo).sign(algorithm);
            System.out.println("token:" + token);
        } catch (JWTCreationException exception) {
            // Invalid Signing configuration / Couldn't convert Claims.
        }
    }
}
```

Go 签名示例

使用 jwt-go 库计算签名，请使用命令 `go get github.com/dgrijalva/jwt-go` 进行安装。

```
package main

import (
        "fmt"
        "time"
        "strconv"
        "github.com/dgrijalva/jwt-go"
)

func main() {
        appId := 1255566655 // 用户 appid
        fileId := "4564972818519602447" // 目标 FileId
        audioVideoType := "RawAdaptive" // 播放的音视频类型
        rawAdaptiveDefinition := 10 // 允许输出的未加密的自适应码流模板 ID
        imageSpriteDefinition := 10 // 做进度条预览的雪碧图模板 ID
        currentTime := time.Now().Unix()
        psignExpire := currentTime + 3600 // 可任意设置过期时间，示例1h
        urlTimeExpire := strconv.FormatInt(psignExpire, 16) // 可任意设置过期时间，16进
        playKey := []byte("TxtyhLlgo7J3iOADIron")

        // Create a new token object, specifying signing method and the claims
        // you would like it to contain.
        token := jwt.NewWithClaims(jwt.SigningMethodHS256, jwt.MapClaims{
                "appId":            appId,
                "fileId":           fileId,
                "contentInfo": {
                        "audioVideoType": audioVideoType,
                        "rawAdaptiveDefinition": rawAdaptiveDefinition,
                        "imageSpriteDefinition": imageSpriteDefinition,
                },
                "currentTimeStamp": currentTime,
                "expireTimeStamp":  psignExpire,
                "urlAccessInfo": map[string]string{
                        "t": urlTimeExpire,
                },
        })

        // Sign and get the complete encoded token as a string using the secret
        tokenString, err := token.SignedString(playKey)
```

```
        fmt.Println(tokenString, err)
}
```

C# 签名示例

使用 jose-jwt 计算签名，请使用 NuGet 命令 `Install-Package jose-jwt` 进行安装。

```
using System;
using System.Text;
using System.Collections.Generic;
using Jose;

public class Program
{
        public static void Main()
        {
                var appId = 1255566655; // 用户 appid
                var fileId = "4564972818519602447"; // 目标 FileId
                var audioVideoType = "RawAdaptive"; // 播放的音视频类型
                var rawAdaptiveDefinition = 10; // 允许输出的未加密的自适应码流模板 ID
                var imageSpriteDefinition = 10; // 做进度条预览的雪碧图模板 ID
                var currentTime = DateTimeOffset.UtcNow.ToUnixTimeSeconds();
                var psignExpire = currentTime + 3600; // 可任意设置过期时间，示例1h
                var urlTimeExpire = psignExpire.ToString("X4"); // 可任意设置过期时间,
                var playKey = "TxtyhLlgo7J3iOADIron";
                var playKeyBytes = Encoding.ASCII.GetBytes(playKey);
                var payload = new Dictionary<string, object>()
                {
                        {"appId", appId},
                        {"fileId", fileId},
                        {"contentInfo", new Dictionary<string, object>()
                                {
                                        {"audioVideoType", audioVideoType},
                                        {"rawAdaptiveDefinition", rawAdaptiveDefini
                                        {"imageSpriteDefinition", imageSpriteDefini
                                }
                        },
                        {"currentTimeStamp", currentTime},
                        {"expireTimeStamp", psignExpire},
                        {"urlAccessInfo", new Dictionary<string, object>()
                                {
                                        {"t", urlTimeExpire}
```

```
                                }
                            }
                    };
                    string token = Jose.JWT.Encode(payload, playKeyBytes, JwsAlgorithm.
                    Console.WriteLine(token);
            }
    }
```

PHP 签名示例

使用 php-jwt 计算签名，请使用命令 `composer require firebase/php-jwt` 进行安装。

```
<?php
require 'vendor/autoload.php';
use \\Firebase\\JWT\\JWT;

$appId = 1255566655; // 用户 appid
$fileId = "4564972818519602447"; // 目标 FileId
$audioVideoType = "RawAdaptive"; // 播放的音视频类型
$rawAdaptiveDefinition = 10; // 允许输出的未加密的自适应码流模板 ID
$imageSpriteDefinition = 10; // 进度条预览的雪碧图模板 ID
$currentTime = time();
$psignExpire = $currentTime + 3600; // 可任意设置过期时间，示例1h
$urlTimeExpire = dechex($psignExpire); // 可任意设置过期时间，16进制字符串形式，示例1h
$playKey = "TxtyhLlgo7J3iOADIron";

$payload = array(
    "appId" => $appId,
    "fileId" => $fileId,
    "contentInfo" => array(
        "audioVideoType"=> $audioVideoType,
                "rawAdaptiveDefinition"=> $rawAdaptiveDefinition,
                "imageSpriteDefinition"=> $imageSpriteDefinition
    ),
    "currentTimeStamp" => $currentTime,
    "expireTimeStamp" => $psignExpire,
    "urlAccessInfo" => array(
        "t" => $urlTimeExpire
    )
);

$jwt = JWT::encode($payload, $playKey, 'HS256');
```

```
print_r($jwt);
?>
```

Node.js 签名示例

使用 jsonwebtoken 计算签名，请使用命令 `npm install jsonwebtoken` 进行安装。

```
var jwt = require('jsonwebtoken');

var appId = 1255566655 // 用户 appid
var fileId = "4564972818519602447" // 目标 FileId
var audioVideoType = "RawAdaptive" // 播放的音视频类型
var rawAdaptiveDefinition = 10 // 允许输出的未加密的自适应码流模板 ID
var imageSpriteDefinition = 10 // 做进度条预览的雪碧图模板 ID
var currentTime = Math.floor(Date.now()/1000)
var psignExpire = currentTime + 3600 // 可任意设置过期时间，示例1h
var urlTimeExpire = psignExpire.toString(16) // 可任意设置过期时间，16进制字符串形式，示例
var playKey = 'TxtyhLlgo7J3iOADIron'

var payload = {
        appId: appId,
        fileId: fileId,
        contentInfo: {
                audioVideoType: audioVideoType,
                rawAdaptiveDefinition: rawAdaptiveDefinition,
                imageSpriteDefinition: imageSpriteDefinition
        },
        currentTimeStamp: currentTime,
        expireTimeStamp: psignExpire,
        urlAccessInfo: {
                t: urlTimeExpire
        }
}
var token = jwt.sign(payload, playKey);
console.log(token);
```

防盗链设置
防盗链综述

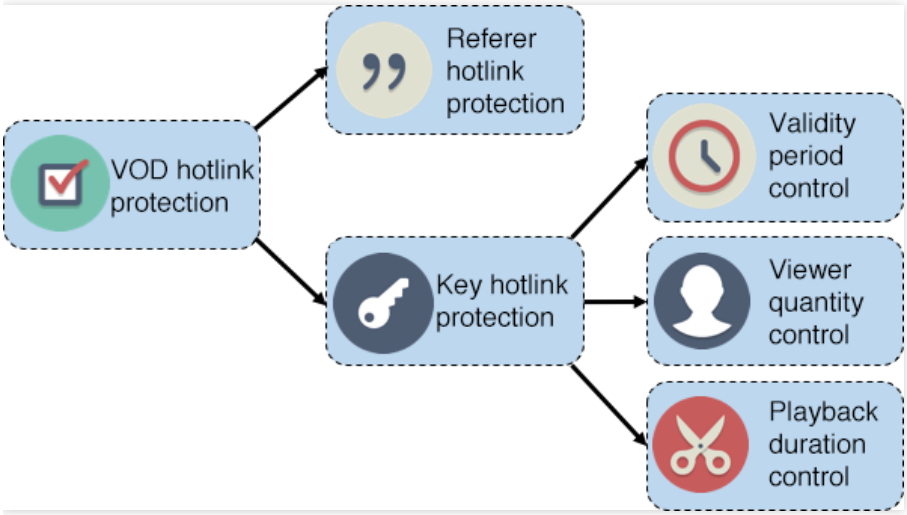
最近更新时间：2021-03-17 10:37:28

简介

为支持视频播放的权限控制，云点播推出了防盗链的解决方案。开通防盗链后，腾讯云 CDN 节点将对播放请求中的关键信息进行检查，并对检查通过的请求返回视频数据。本方案对播放器并无限制要求，即无论是云点播的播放器 SDK，还是一般性播放器均可使用。

类型和能力

云点播防盗链支持 Referer 防盗链和 Key 防盗链。



Referer 防盗链

基于 HTTP 协议支持的 Referer 机制，通过播放请求 Header 中携带的 Referer 字段识别请求的来源。开发者可以设置一批域名为黑名单或白名单，CDN 节点将按照名单中的域名做鉴权，从而允许或拒绝播放请求。

Key 防盗链

允许开发者将视频的播放控制参数以 QueryString 的形式拼接在视频 URL 中，CDN 节点将检查 URL 中的播放控制参数，并依据参数控制视频的播放。目前，Key 防盗链通过“过期时间参数”、“允许播放的 IP 数量参数”和“试看时间参数”，支持“防盗链有效时间控制”、“防盗链播放人数控制”和“视频播放时长控制”。

防盗链有效时间控制

在视频 URL 中指定过期时间。如果请求的视频 URL 已过期，则视频无法播放。通过这种方式，可以为视频 URL 设置有效时间，防范他人将视频 URL 转移到其他站点后长期使用。

防盗链播放人数控制

在视频 URL 中指定链接最多能供多少人播放。不在同一内网的播放终端，它们的公网 IP 一般是不同的。通过限制一个 URL 允许最多能被多少公网 IP 播放，就能够限制同一个 URL 可以播放的人数，从而可以防范他人将视频 URL 转移到其他站点后，无限制地分发给任意多的人数观看。

视频允许播放时长控制

在视频 URL 中指定试看时长（如仅允许播放视频的前5分钟）。通过这种方式，可以实现对未付费用户的试看功能。

说明：

- 关于 Referer 防盗链，更多详情请参见 Referer 防盗链。
- 关于 Key 防盗链，更多详情请参见 Key 防盗链。

Referer 防盗链

最近更新时间：2022-12-22 10:26:17

功能介绍

- 基于 HTTP 协议支持的 Referer 机制，通过 HTTP 头部中携带的 Referer 字段识别请求的来源。开发者可以通过配置 Referer 黑白名单，对视频请求来源进行识别和鉴权 。
- 支持黑名单和白名单两种模式。当视频播放请求到达 CDN 节点后，节点将依据用户配置的 Referer 黑白名单对请求来源鉴权。对于符合规则的请求，CDN 将返回视频数据，否则，将返回403响应码，拒绝播放请求。

> 说明：
> 开启 Referer 防盗链请参见 设置防盗链。

注意事项

- 该功能为可选项，默认不启用。
- 开启功能后，选择并填写黑名单或白名单，黑名单和白名单互斥，同一时间仅支持一种模式。
- 黑名单或白名单中的域名支持1条 - 10条，每一行一条记录。
- 域名前不要带协议名（ `http://` 和 `https://` ），域名为前缀匹配（如填写 `abc.com` ，则 `abc.com/123` 和 `abc.com.cn` 也会匹配），且支持通配符（如 `*.abc.com` ） 。

Key 防盗链

最近更新时间：2024-08-26 16:43:46

功能介绍

支持在视频 URL 中指定过期时间，他人获取后无法长期使用。
支持在视频 URL 中指定最大允许播放 IP 数，他人获取后不能无限制地分发给更多人观看。
支持在视频 URL 中指定试看时长，实现试看功能。
支持在视频 URL 中指定地区访问限制，支持黑名单和白名单两种模式。
支持在视频 URL 中指定 Referer 黑白名单。
开发者使用密钥 `KEY` 对视频 URL 签名，并在 URL 中带上签名结果。只要用户密钥不泄露，其他用户无法伪造视频 URL。
CDN 节点检查视频 URL 中的参数和签名，对视频播放请求进行控制。如果请求检查不通过，则返回403响应码。
支持的文件类型：MP4、TS、M3U8、FLV、AAC、MOV、WMV、AVI、MP3、RMVB、MKV、MPG、3GP、WEBM、M4V、ASF、F4V、WAV、MPEG、VOB、RM、WMA、DAT、M4A、MPD、M4S。
说明：
开启 Key 防盗链请参见 设置防盗链。
目前防盗链 key 的试看功能暂不支持音频格式文件。

防盗链 URL 生成方式

开发者在云点播中的视频均存在**视频原始 URL**。未开启防盗链时，使用视频原始 URL 即可播放视频。
开启 Key 防盗链后，视频原始 URL 不再能播放，此时需要构造视频的**防盗链 URL**。
防盗链 URL 的生成规则是在原始 URL 尾部，以 QueryString 的方式加入防盗链参数，形如：

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=[t]&exper=[exper]&rlimit=[
```

下面详细介绍防盗链 URL 中各个参数的含义和取值方法。

防盗链参数

参数名	必选	说明
KEY	是	开启 Key 防盗链时填写的密钥。必须由大小写字母（a - Z）或者数字（0 - 9）组成，长度在8 - 20个字符之间。建议在控制台中单击【生成KEY】生成，具体操作步骤请参见 设置防盗链。
Dir	是	视频原始 URL 的 PATH 中除去文件名的那部分路径。如果原始 URL 为 http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4，则播放路径为 /dir1/dir2/。

t	是	播放地址的过期时间戳，以 Unix 时间的十六进制小写形式表示。 过期后该 URL 将不再有效，返回403响应码。考虑到机器之间可能存在时间差，防盗链 URL 的实际过期时间一般比指定的过期时间长5分钟，即额外给出300秒的容差时间。 建议过期时间戳不要过短，确保视频有足够时间完整播放。
exper	否	试看时长，单位为秒，以十进制表示，不填或者填0表示不试看（即返回完整视频）。 试看时长不要超过视频原始时长，否则可能导致播放失败。
rlimit	否	最多允许多少个不同 IP 的终端播放，以十进制表示，最大值为9，不填表示不做限制。 当限制 URL 只能被1个人播放时，建议 rlimit 不要严格限制成1（例如可设置为3），因为移动端断网后重连 IP 可能改变。
us	否	链接标识，用于随机化一个防盗链 URL，增强链接的唯一性。 建议每次生成防盗链 URL 时，指定一个随机的 us 值。
whreg	否	允许访问的地区列表，支持1条 - 10条，用半角逗号分隔，取值为 ISO 3166-1三位字母代码。
bkreg	否	禁止访问的地区列表，支持1条 - 10条，用半角逗号分隔，取值为 ISO 3166-1三位字母代码。
whref	否	允许访问的域名列表，支持1条 - 10条，用半角逗号分隔。域名前不要带协议名（http://和 https://），域名为前缀匹配（如填写 abc.com，则 abc.com/123 和 abc.com.cn也会匹配），且支持通配符（如 *.abc.com）。
bkref	否	禁止访问的域名列表，支持1条 - 10条，用半角逗号分隔。域名前不要带协议名（http://和 https://），域名为前缀匹配（如填写 abc.com，则 abc.com/123 和 abc.com.cn也会匹配），且支持通配符（如 *.abc.com）。
sign	是	防盗链签名，以32个字符长的十六进制数表示，用于校验防盗链 URL 的合法性 签名校验失败将返回403响应码。下面将介绍 签名计算公式。

签名计算公式

```
sign = md5(KEY + Dir + t + exper + rlimit + us + whref + bkref + whreg + bkreg)
```

公式中的 `+` 代表字符串拼接，选填参数可以为空字符串。

防盗链 URL 生成示例

如果某个开发者在云点播中有一个视频，视频的原始播放 URL 是 `http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4` 。该开发者开通了 Key 防盗链，生成的密钥是 `24FEQmTzro4V5u3D5epW` ，生成的随机字符串为 `72d4cd1101` ，现有如下需求：

1. 为这个视频生成防盗链 URL，URL 的过期时间是2018年01月31日20:00（Unix 时间为1517400000）。
2. 生成一个试看 URL，试看时长为视频的前5分钟（视频原始时长大于5分钟）。
3. 限制 URL 可播放的 IP 数，允许最多3个不同 IP 的终端可以播放该 URL。

下面分别对“视频播放地址有效时间控制”、“视频播放地址允许最多播放 IP 数”和“视频允许播放时长控制”的场景，介绍如何生成防盗链 URL。

示例1：播放地址有效时间控制

步骤1：确定防盗链参数

参数名	取值	说明
KEY	24FEQmTzro4V5u3D5epW	开发者开通 Key 防盗链时选择的密钥
Dir	/dir1/dir2/	原始播放 URL 的 PATH 中除去 myVideo.mp4 的剩余部分
t	5a71afc0	过期时间戳1517400000的十六进制表示结果
us	72d4cd1101	生成的随机字符串

步骤2：计算签名

```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc072d4cd1101") = "3d8488faeb37d52d
```

步骤3：生成防盗链 URL

将防盗链参数拼接到视频原始 URL 的 QueryString 中，得到视频防盗链 URL：

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&us=72d4cd1101&sig
```

示例2：播放地址最多可播放 IP 数

步骤1：确定防盗链参数

参数名	取值	说明
`KEY`	`24FEQmTzro4V5u3D5epW`	开发者开通 Key 防盗链时选择的密钥
`Dir`	`/dir1/dir2/`	原始播放 URL 的 PATH 中除去 `myVideo.mp4` 的剩余部分
`t`	`5a71afc0`	过期时间戳1517400000的十六进制表示结果
`rlimit`	`3`	限制最多允许3个不同的 IP 播放 URL
`us`	`72d4cd1101`	生成的随机字符串

步骤2：计算签名

```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc0372d4cd1101") = "c5214f0d5961b13
```

步骤3：生成防盗链 URL

将防盗链参数拼接到视频原始 URL 的 QueryString 中，得到视频防盗链 URL：

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&rlimit=3&us=72d4c
```

示例3：允许播放时长控制

步骤1：确定防盗链参数

参数名	取值	说明
KEY	24FEQmTzro4V5u3D5epW	开发者开通 Key 防盗链时选择的密钥
Dir	/dir1/dir2/	原始播放 URL 的 PATH 中除去 myVideo.mp4 的剩余部分
t	5a71afc0	过期时间戳1517400000的十六进制表示结果
exper	300	试看前5分钟，即300秒
us	72d4cd1101	生成的随机字符串

步骤2：计算签名

```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc030072d4cd1101") = "547d98c4b91e8
```

步骤3：生成防盗链 URL

将防盗链参数拼接到视频原始 URL 的 QueryString 中，得到视频防盗链 URL：

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&exper=300&us=72d4
```

Key 防盗链生成和校验工具

云点播为开发者提供了 Key 防盗链 URL 的生成工具和校验工具，开发者可以使用该工具快速准确地生成和校验符合要求的防盗链 URL。

Key 防盗链生成工具

Key 防盗链校验工具

注意事项

该功能为可选项，默认不启用。
启用该功能后，视频原始 URL 将不再能直接播放，需要按规则生成合法的防盗链 URL。
密钥 `KEY` 必须由大小写字母（a - Z）或者数字（0 - 9）组成，长度在8 - 20个字符之间。
若防盗链 URL 过期，或者签名不能通过，将无法播放视频，并返回403响应码。
防盗链 URL 中 QueryString 的各参数必须按照 `t` 、 `exper` 、 `rlimit` 、 `us` 、 `sign` 的顺序出现，如果顺序不正确将无法播放视频。
如果使用试看功能，需确保试看时长不大于视频时长，否则将导致视频无法播放。
试看对视频的格式有较严格的要求（仅支持 H.264，视频元信息在视频文件的头部等），不符合格式要求的原始视频使用试看功能将产生异常。建议使用云点播转码功能进行转码，对转码后视频设置试看（转码后的格式均符合试看格式要求）。

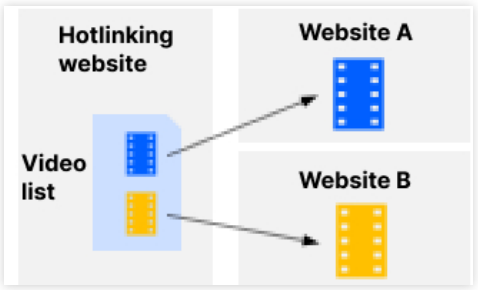
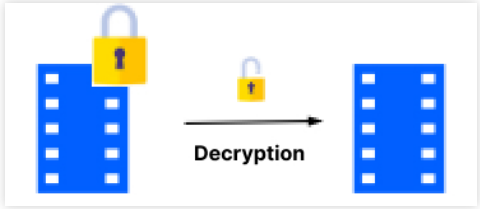
媒体加密与版权保护
版权保护综述

最近更新时间：2024-10-16 14:17:20

在线教育平台和 OTT 视频门户，都包含大量的优质内容。用户购买影片或课程，或者购买会员套餐，获权观看完整内容。
然而，随着盗版侵权行为的日益猖獗，版权安全成为音视频行业点播领域的最大挑战。

侵权的主要形式

盗版侵权的形式，主要归纳为三种形式：**盗链**、**破解**和**盗录**。

盗版侵权形式	盗链	破解
说明	播放 URL 被挂到其他站点。  	加密后的视频被破解传播 。 

版权保护能力

腾讯云点播在版权保护领域积累多年的丰富经验，提供了一整套的版权保护能力。

类别	能力项	云点播对应功能
内容保护	媒体内容加密保护	HLS 私有加密、商业级 DRM
	防抓包和破解工具破解	HLS 私有加密、商业级 DRM
	防浏览器插件破解	HLS 私有加密、商业级 DRM
	防浏览器插件截录屏 当识别到截屏、录屏行为时，播放器黑屏，录制/截屏内容均为黑屏内容	商业级 DRM
	防系统软件截录屏 当识别到截屏、录屏行为时，播放器黑屏，录制/截屏内容均为黑屏内容	商业级 DRM
播放保护	限制播放的国家和地区	Key 防盗链

	限制播放的站点	Referer 防盗链
	限制播放链接的有效时长	Key 防盗链
	限制播放链接的最大播放 IP 数	Key 防盗链
	限制内容的试看内容时长	Key 防盗链
盗录追踪	追踪盗录者身份	浮动水印、溯源水印

媒体加密安全级别与兼容度

云点播的媒体加密，有 HLS 私有加密和商业级 DRM 两套方式：

HLS 私有加密

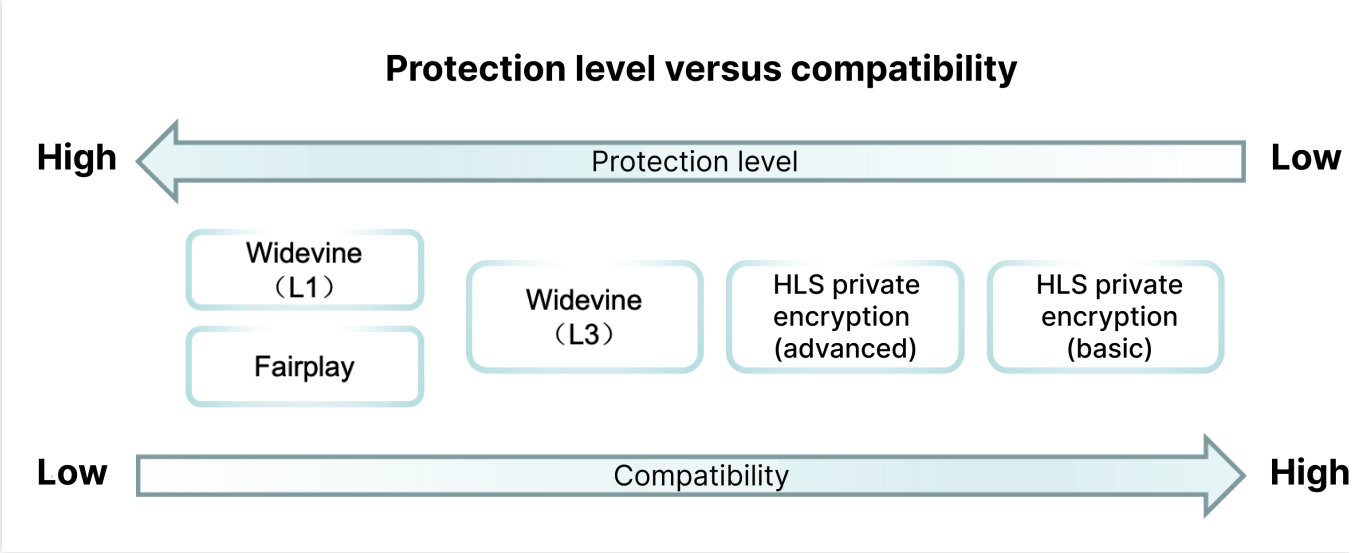
是腾讯云点播自研的媒体加密，默认工作在「标准」模式，当遇到部分播放端不支持标准模式时，播放器会自动尝试以「基础」模式播放。

商业级 DRM

云点播目前支持 FairPlay（苹果系平台） 和 Widevine（谷歌系平台）。其中，Widevine 支持 L1 和 L3 两种级别，L1 级别的解密需要专门的硬件 TEE 环境，对播放端的要求更高。如果设备本身不支持 L1，则会自动尝试以 L3 级别播放。

内容保护方式	安全级别	兼容度
HLS 私有加密（标准）	**高** 对内容做加密保护。 防抓包和破解工具能力强。 防浏览器插件破解、截录屏能力强。 防系统和软件截录屏能力较弱。	**高** 支持移动端 App 播放。 PC 端支持大部分浏览器。 Android 端支持大部分浏览器，iOS 浏览器暂不支持。
HLS 私有加密（基础）	**中** 对内容加密保护。 防破解和录屏工具能力较弱。	**极高** 基本可兼容已知的各端播放。
FairPlay	**极高** 硬件级解密。 防各类工具和插件破解能力极强。 防系统、软件、插件截录屏能力极强。	**较高** 支持 iOS 端 App 播放。 iOS 端和 PC 端，支持 Safari 浏览器播放。
Widevine（L1）	**极高** 硬件级解密。	**较低** 不支持浏览器播放。

	防各类工具和插件破解能力极强。 防系统、软件、插件截录屏能力极强。	支持部分 Android 端 App 播放。
Wiedvine（L3）	**高** 软件级解密。 防各类工具和插件破解强。 对录屏工具的防范能力较强。	**中** 支持 Chrome 浏览器和部分 Chromium 内核浏览器播放。 支持部分 Android 端的 App播放。



从上面的图表可以看出来，内容保护方式（HLS 私有加密、商业级 DRM）的安全级别和兼容度呈逆相关的关系，即通常安全级别越高，对播放端的兼容度越低。因此，如何平衡更高的安全级别和更高的兼容度，成为版权保护必须思考的问题。

下面的最佳实践部分，将为您实施版权保护提供指引。

最佳实践

最佳实践部分，将从内容保护、盗录追踪、播放保护三个方面，介绍如何为您组建最佳的版权方案。

内容保护

前面我们已经了解到，内容保护的安全级别和播放端兼容度呈现逆相关。如果要同时兼顾更高的安全级别和兼容度，建议根据分辨率划分档次，采用不同的内容保护方式：

HLS 私有加密，仅允许播放 720P 以下的规格。

Widevine 和 FairPlay 保护的内容，允许播放全部分辨率的规格。

云点播的播放器 SDK 播放加密的媒体内容时，会先尝试播放 Widevine 和 FairPlay 保护的输出；如果终端不支持商业级 DRM，则自动降级，播放 HLS 私有加密保护的输出。
采用上面的方式，您的高价值内容（720P 分辨率及以上）受到商业级 DRM 硬件级加解密的保护，做到极强的防破解、防截录屏能力；而 HLS 私有加密做到了更广的兼容度，即使内容遭到破解或盗录，也只会泄露价值较低的内容（720P以下）。

盗录追踪

因为商业级 DRM 的兼容度问题，一般需要配合使用 HLS 私有加密，而播放私有加密仍存在盗录的风险。因此，启用浮动水印和溯源水印，可以对盗录者的身份进行追踪的关键手段。
浮动水印是客户端在播放时，在视频画面上覆盖的一层水印，水印的内容通常是观看者的 ID。浮动水印的使用成本极低，同时也能一定程度上震慑盗录者。
溯源水印是一种数字水印，被编码到画面和声音中。音视频被盗录后，可从云端提取到盗录者的 ID。相比浮动水印，溯源水印更美观（肉眼不可见），更安全（水印被编码，无法擦除和遮挡）。
因此，建议您同时使用浮动水印和溯源水印，实现对盗录行为的有效追踪。

播放保护

通过防盗链，可以实现对您媒体链接的播放保护，建议您：
开启 Referer 白名单，将您的站点域名加入白名单中，防止白名单以外的站点访问视频。
在 KEY 防盗链中，将播放链接的有效时间指定为视频时长加30分钟。
在 KEY 防盗链中，将播放链接允许播放的最大 IP 数指定为3。
在 KEY 防盗链中，将播放链接设置为仅允许站点所在国家或地区播放。

最佳实践操作举例

下面将针对上面提到的最佳实践，举例说明如何操作和体验。

如何申请 FairPlay 证书

播放 FairPlay 加密的内容，需要先根据 这里 的指引申请 FairPlay 证书，并根据 这里 的指引提交 FairPlay 证书。

如何加密视频并打溯源水印

将您的视频上传到云点播，并转出 HLS 私有加密、FairPlay 和 Widevine 加密的自适应码流：
1. 登录 云点播控制台，进入应用后在导航栏中的**媒资管理 > 音视频管理**，单击**上传音视频**，上传您的视频。
2. 上传完成之后，选中您的视频，单击**任务流**按钮，从“任务流模板”中选择 MultiDRMPreset 任务流，并单击**确定**开始加密；
等待任务执行完成之后，您已经转出了 HLS 私有加密输出（仅包含 480P 和以下的分辨率），FairPlay 加密的输出和 Widevine 的加密输出。同时，所有的加密输出都打上了溯源水印。
注意：
MultiDRMPreset 任务流，会为 HLS 私有加密的输出打上溯源水印，而溯源水印要求您的视频时长必须大于6分钟。如果视频不足6分钟，任务将执行失败。

如何启用防盗链

启用和设置 Referer 防盗链：

1. 登录 云点播控制台，进入应用后在导航栏中的**分发播放设置 > 域名管理**，对您使用的域名单击**设置。**
2. 进入**访问控制**菜单栏，开启 “Referer 防盗链”，“防盗链类型”选择“**白名单**”，文本框中输入允许播放视频的站点域名列表，单击**确定。**
3. 打开 “Key 防盗链”，输入或随机生成防盗链 Key，单击**确定**。

此时，已为域名开启了 Referer 防盗链和 Key 防盗链。

如何组装播放器签名

使用云点播的播放器播放视频时，必须使用播放器签名，签名的传参规则是：

1. appId 和 fileId 分别填您的 appId 和视频对应的 fileId；
2. contentInfo 中的 audioVideoType 填 ProtectedAdaptive；
3. drmAdaptiveInfo 中的 privateEncryptionDefinition 填14，widevineDefinition 填21，fairPlayDefinition 填12；
4. urlAccessInfo 中的 t 填播放链接过期时间（当前时间 + 视频时长 + 30分钟）；rlimit 填3；us 填随机生成的字符串（每次生成签名都随机生成）；uv 填观看者唯一的 ID。

播放加密的视频

1. 根据 Web 端、iOS 端、Android 端 的播放指引，为播放端接入播放器 SDK；
2. 参考 Web 端、Android 端、iOS 端 的播放指引，在视频播放时，设置客户端上的浮动水印；
3. 播放器向您的业务服务器获取播放器签名，开始播放。

按照以上的方式进行操作，您的播放器将开始播放加密的视频。对于支持 FairPlay 和 Widevine 的播放端，将优先播放商业级 DRM 保护的输出；对于不支持 FariPlay 和 Widevine 的播放端，将播放私有加密的输出。

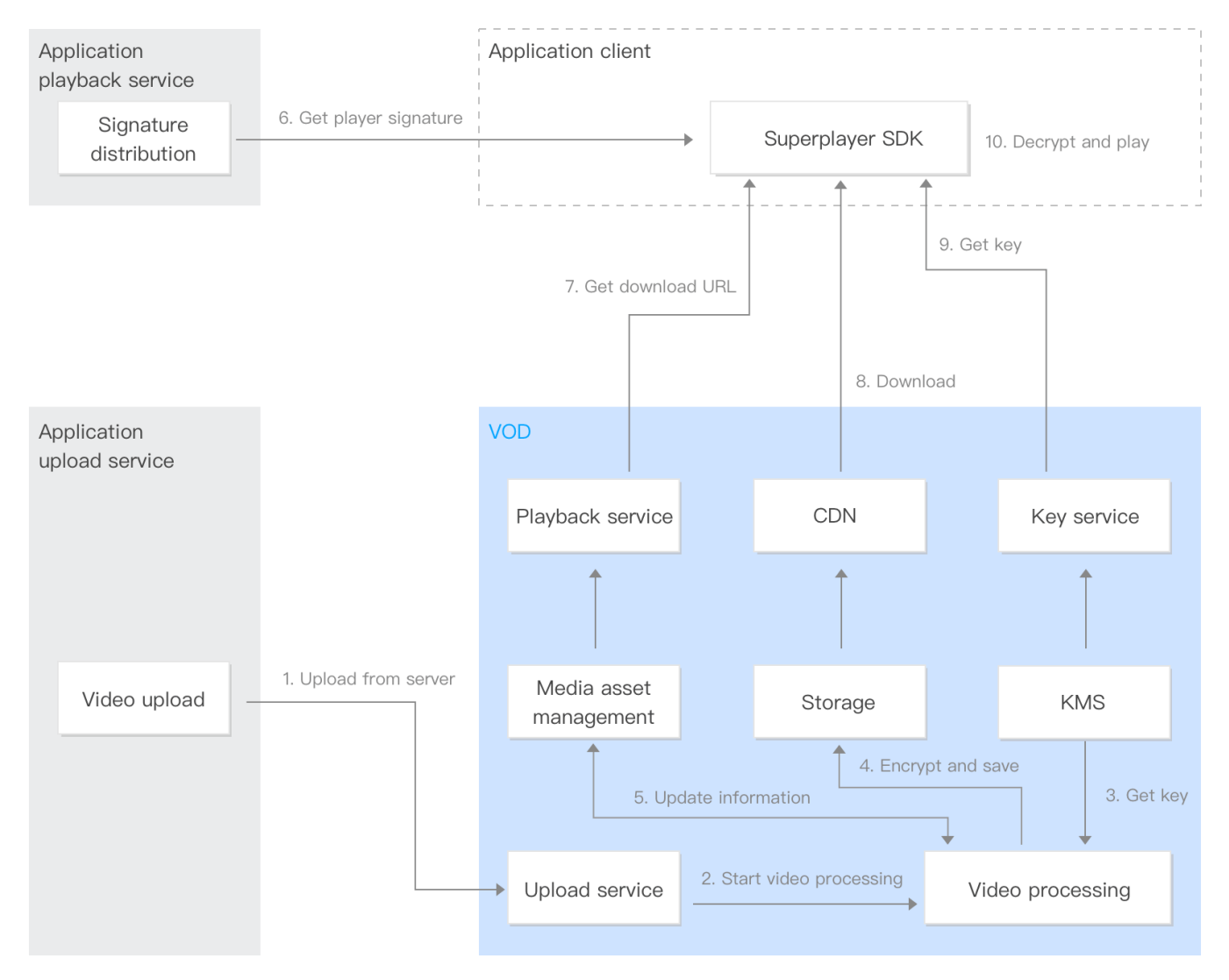
HLS 私有加密

最近更新时间：2022-08-22 15:31:50

HLS 私有加密是云点播独创的视频内容加密方案，使用私有协议保护内容密钥，可有效防范多种浏览器插件和灰产工具的破解。

原理架构

HLS 私有加密和解密播放的整体架构流程如下：



1. **服务端上传**：业务后台通过控制台、服务端 API 等方式，将视频上传到云点播。
2. **触发视频处理**：上传视频的同时，指定带加密的转自适应码流。上传后，视频开始加密过程。
3. **获取密钥**：转自适应码流并加密，点播从 KMS 模块获取加密视频时使用的密钥。

4. **加密并写入存储**：视频转自适应码流并加密后，输出的视频内容被写入到点播的存储中。
5. **更新媒资**：加密后的视频信息，被写入到媒资管理模块。
6. **获取播放器签名**：业务终端集成点播超级播放器，播放器向业务服务器请求播放器签名。
7. **请求下载地址**：超级播放器会从点播的播放服务获取视频的下载地址。
8. **下载内容**：超级播放器通过下载地址，从点播 CDN 下载被加密的内容。
9. **获取密钥**：超级播放器携带播放器签名请求内容密钥，此时内容密钥本身被云点播的私有协议保护，防止密钥被盗取。
0. **解密播放**：超级播放器使用私有协议得到内容密钥，解密播放。

接入参考

为了帮助您快速接入点播的加密能力，我们为您提供了视频加密 接入指引，以示例的方式为您讲解接入步骤。

常见问题

1. 如何对上传的视频进行 HLS 私有加密？
   云点播的 转自适应码流 功能，可以将视频转成多种分辨率并加密。具体实践方式，请您参见 接入指引。
2. 终端如何播放加密后的视频？
   播放点播加密的视频，您需要在终端集成超级播放器 SDK。同时，您还需要搭建一个签名派发服务。具体实践方式，请您参见 接入指引。

商业级 DRM
商业级 DRM 综述

最近更新时间：2022-08-31 17:54:01

当前网络视频行业不断高速增长，网络盗版变得越发猖獗，给内容版权商也带来了非常大的损失，因此，内容版权保护的重要性不言而喻。

商业级 DRM 是一类基于播放许可证（License）实现高安全级别的内容版权保护解决方案。终端播放视频时，必须先获取 License（License 中包含了解密密钥、密钥有效期、终端信息等），然后使用 License 中的密钥解密播放视频。

使用商业级 DRM，有如下优势：

- 密钥不可见：密钥本身被加密，仅内容解密模块（CDM）能读取密钥。
- License 终端绑定：License 仅对单个终端有效，其他终端无法使用。
- License 支持过期：支持指定 License 的有效期。
- 解码过程安全：支持硬件级（TEE）解密和解码。

目前主流的商业级 DRM 解决方案有 Widevine 和 FairPlay 两种：

商业级 DRM 解决方案	适用的自适应码流协议	适用播放环境
Widevine	HLS、DASH	Andriod 播放器，以及 Chrome、Firefox、Edge、Opera 浏览器等
FairPlay	HLS	iOS 播放器，以及 Safari 浏览器等

目前点播支持两种 DRM 许可证服务集成方案：

- 云点播 DRM 集成成案：由腾讯云点播提供 DRM 许可证服务；
- 第三方 DRM 集成方案：由华曦达（SMDC）提供 DRM 许可证服务。

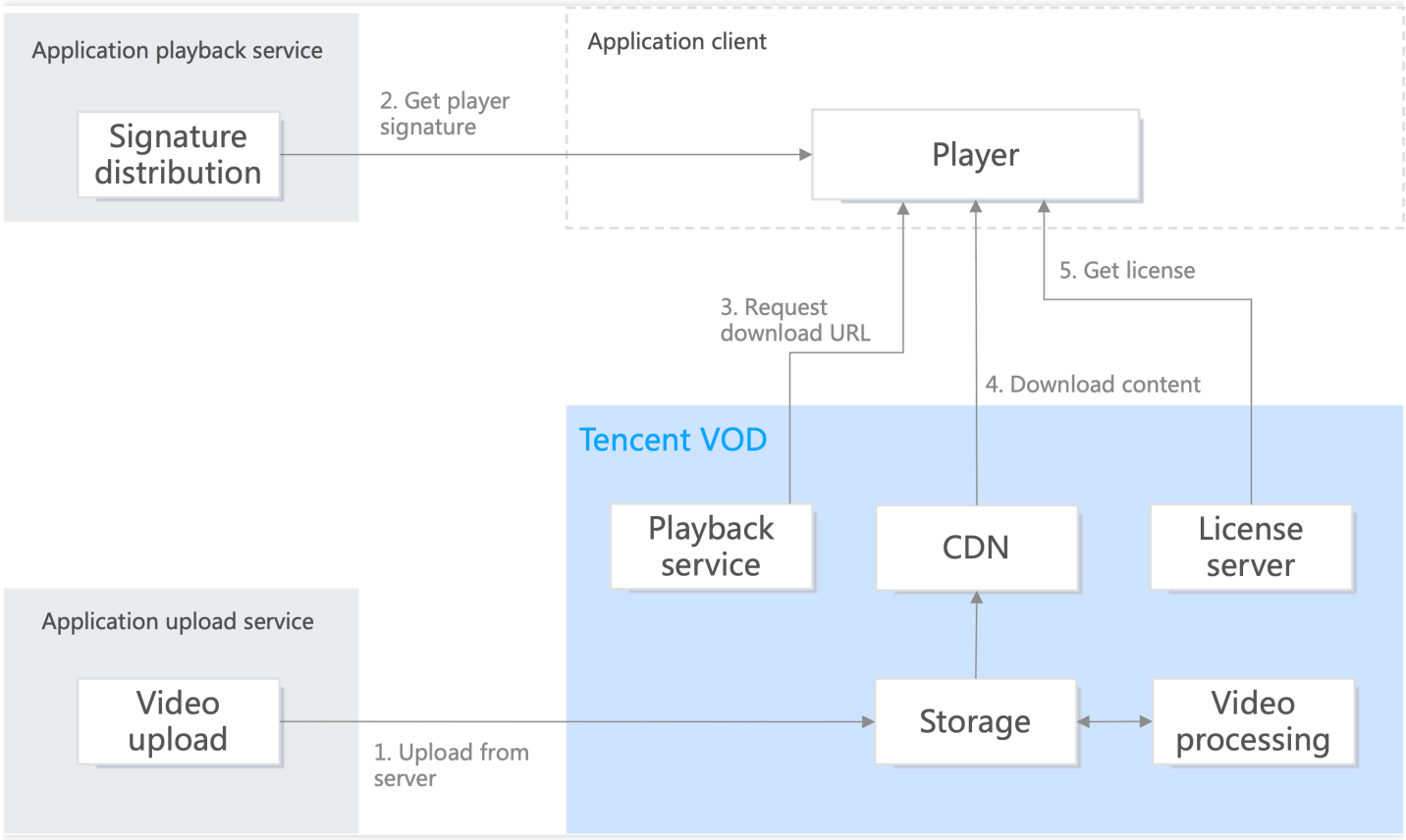
您可以根据需要，选择以上两种方案中的任意一种。

云点播 DRM 集成方案

商业级 DRM 能够为视频内容提供高级别的安全保障，但从零接入的门槛很高。因此，在此方案中，云点播提供了集成商业级 DRM 的一站式解决方案，包括 DRM 加密、证书管理、License 派发、解密播放等功能，帮助您轻松集成

DRM 能力。

加密和解密播放的整体架构流程图如下：

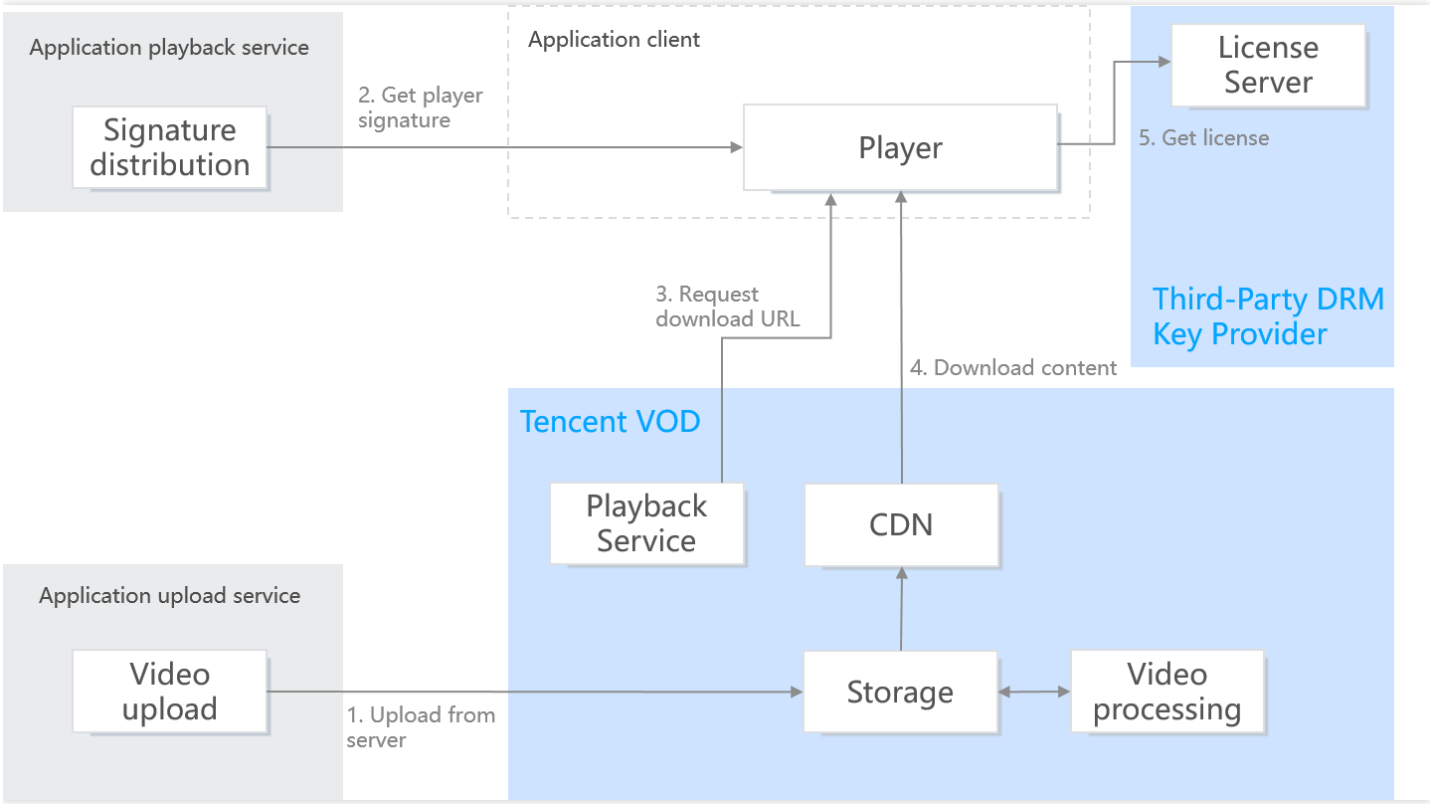


为了帮助您快速接入，我们为您提供了 教程，以示例的方式为您讲解接入步骤。

第三方 DRM 集成方案

在此方案中，云点播提供转码、加密、存储、CDN 等功能，第三方 DRM 提供商（华曦达 SDMC）提供证书管理、License 派发等功能。

加密和解密播放的整体架构流程图如下：



为了帮助您快速接入，我们为您提供了 教程，以示例的方式为您讲解接入步骤。

费用相关

使用商业级 DRM 加密，主要涉及以下费用：

- 转码费用：对视频进行 DRM 加密时，需要转码或转自适应码流，因此会产生转码费用。
- 存储费用：转码或转自适应码流的输出，会占用存储空间，因此会产生存储费用。
- DRM 许可证费用：终端播放 DRM 加密的视频时，需要获取 DRM 许可证，产生 DRM 许可证费用。如果使用第三方 DRM 集成方案，此项费用将由第三方 DRM 提供商收取。

以上费用的具体单价，请参见 购买指南。

如何申请 FairPlay 证书信息

最近更新时间：2023-01-05 11:50:57

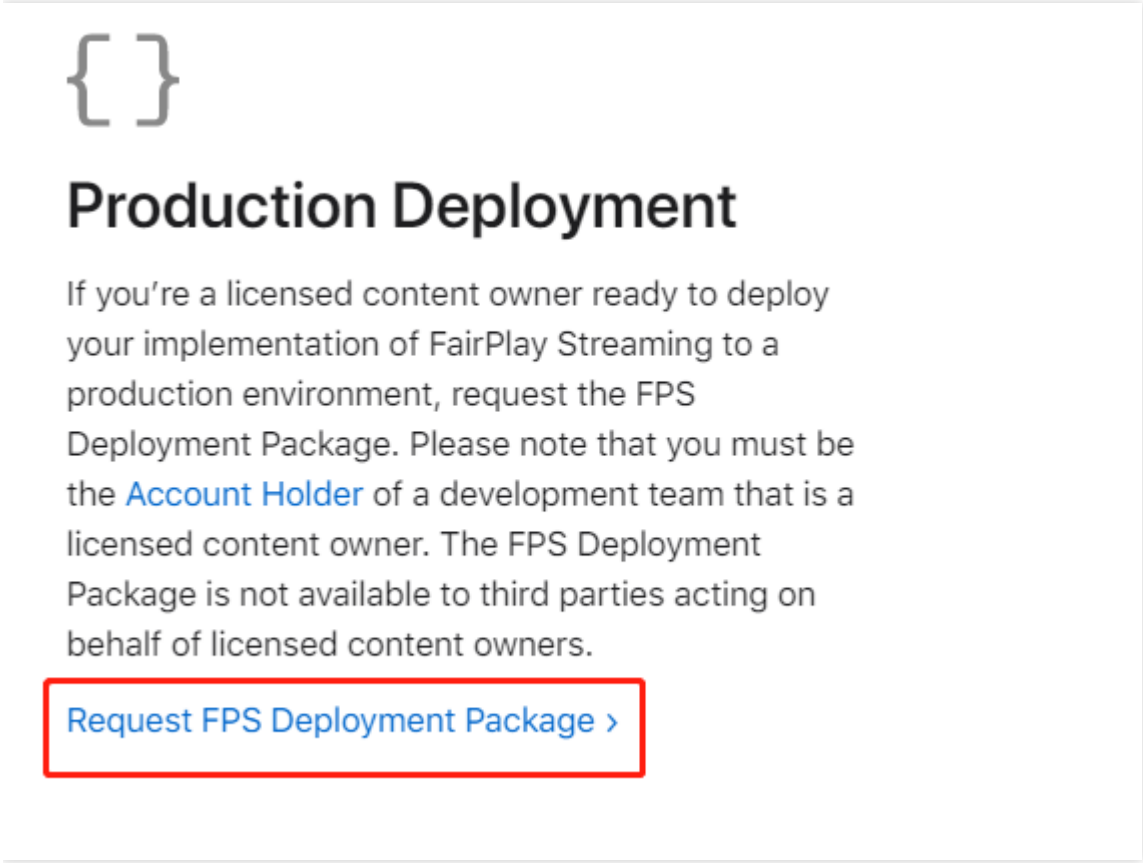
要使用苹果的 FairPlay DRM（FPS），您需要先向苹果申请获得 FPS 部署包。本文手把手教您如何获得 FPS 部署包，和以下关键信息：

- FPS 证书文件（.cer）
- 私钥文件（.pem）
- 私钥密码
- ASK（Application Secret Key）

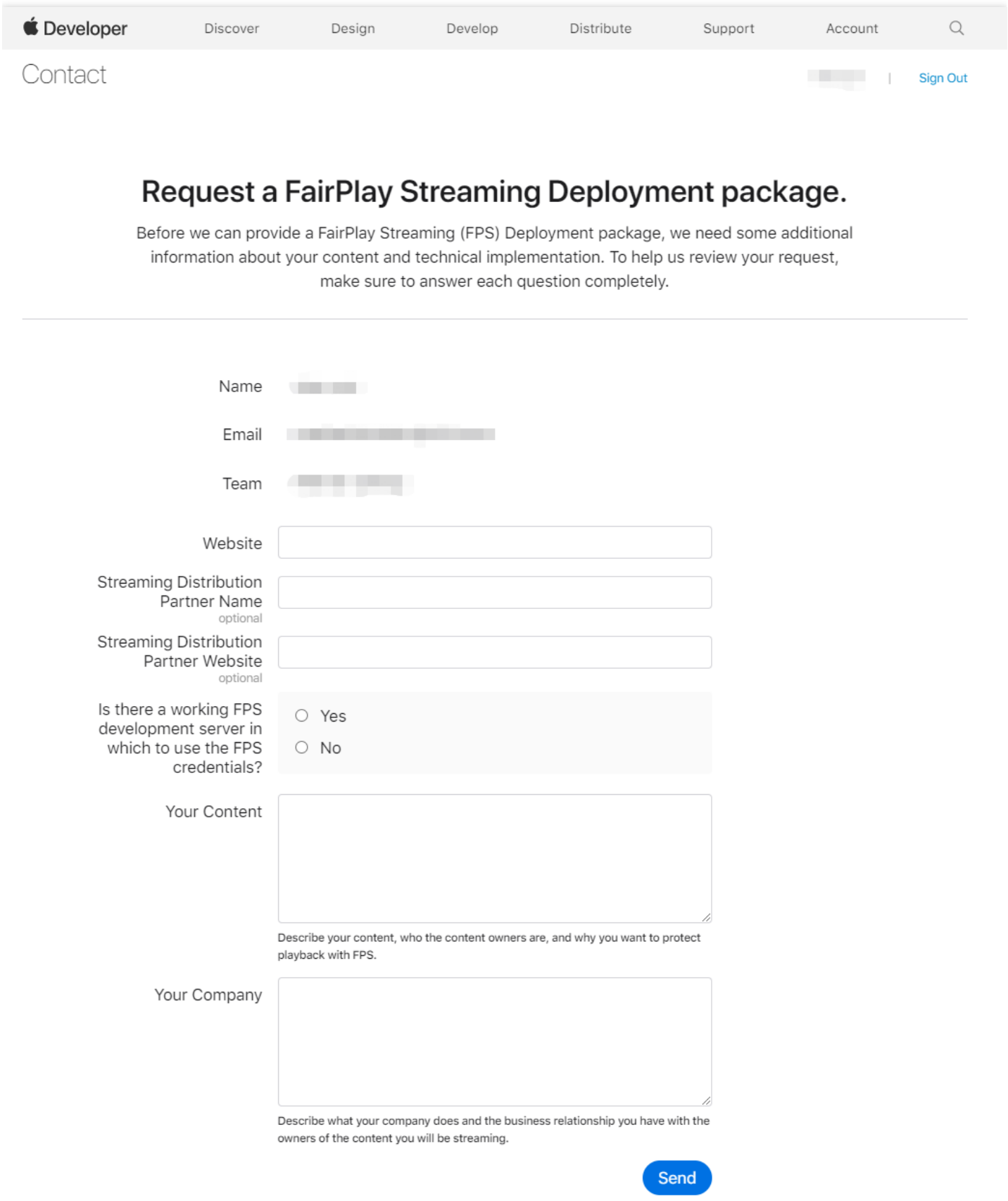
第一步：获取 FairPlay Streaming Deployment Package

1. 访问 苹果 FairPlay 页面，单击页面底部链接 `Request FPS Deployment Package` 后，你将看到一个表单页面。

> 注意：
> 您需要拥有一个苹果开发者账号，成功登录后才能看到表单。



2. 填写页面申请表单，提交后等待苹果公司审批。



3. 当苹果公司通过申请后，您将得到一个 `FPS_Deployment_Package.zip` 压缩包。

说明：
在申请过程中，您将会被询问是否已完成密钥服务器模块（KSM）的实现和测试，对此可以回答：

```
> I am using a 3rd party DRM company and the company has already built and tested KSM
```

第二步：创建私钥和证书签名请求（CSR，Certificate Signing Request）

解压 `FPS_Deployment_Package.zip` ，根据解压后的说明文档（.pdf），创建受密码保护的私钥以及证书签名请求（CSR）。

注意：
需在执行下述过程的 PC 或服务器环境上安装 OpenSSL。

1. 创建私钥文件（ `privatekey.pem` ），执行以下命令：

```
openssl genrsa -aes256 -out privatekey.pem 2048
```

在创建过程中，需要指定私钥密码，务必将私钥密码记录下来，后续步骤需要使用到。另外，建议私钥密码不要超过32个字符。

```
% openssl genrsa -aes256 -out privatekey.pem 2048
Generating RSA private key, 2048 bit long modulus
.........+++
.+++
e is 65537 (0x10001)
Enter pass phrase for privatekey.pem:
Verifying - Enter pass phrase for privatekey.pem:
```

2. 创建证书签名请求（ `certreq.csr` ），执行以下命令：

```
openssl req -new -sha1 -key privatekey.pem -out certreq.csr -subj "/CN=SubjectName/OU=OrganizationalUnit/O=Organization/C=US"
```

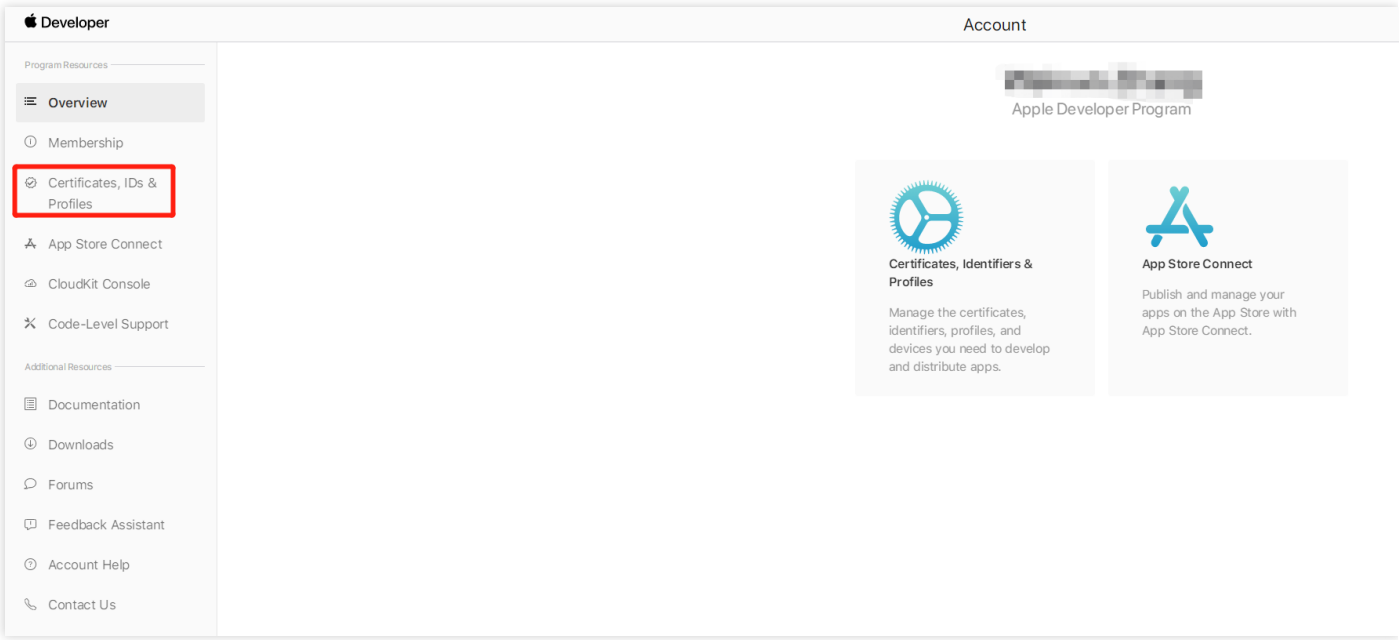
在创建过程中，需要输入在创建私钥文件时指定的私钥密码。

```
% openssl req -new -sha1 -key privatekey.pem -out certreq.csr -subj "/CN=SubjectName/OU=OrganizationalUnit/O=Organization/C=US"
Enter pass phrase for privatekey.pem:
```

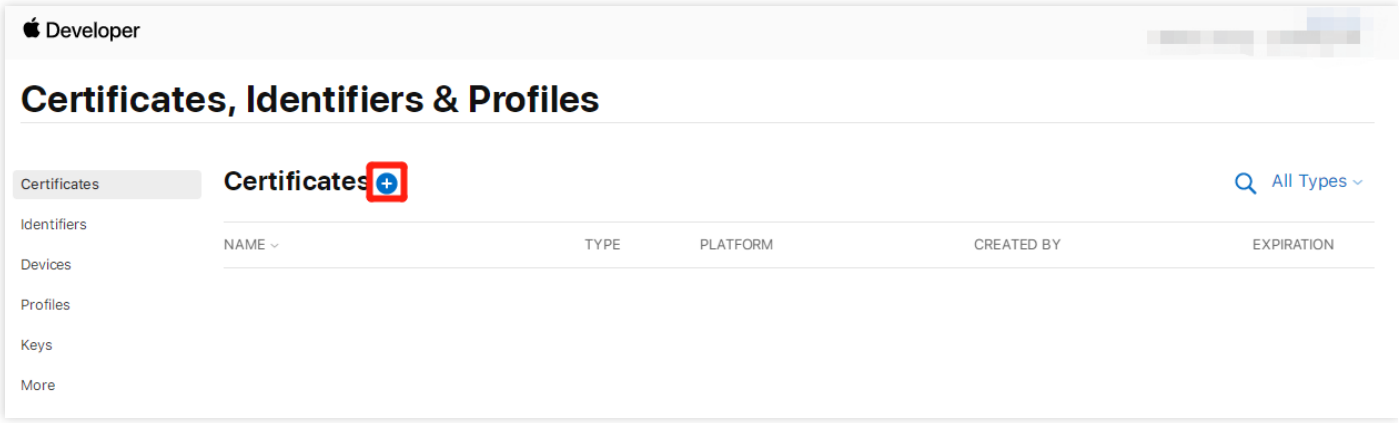
第三步：生成 FPS 证书（FairPlay Streaming Certificate）

访问 苹果开发者页面，获取 FPS 证书和 ASK。

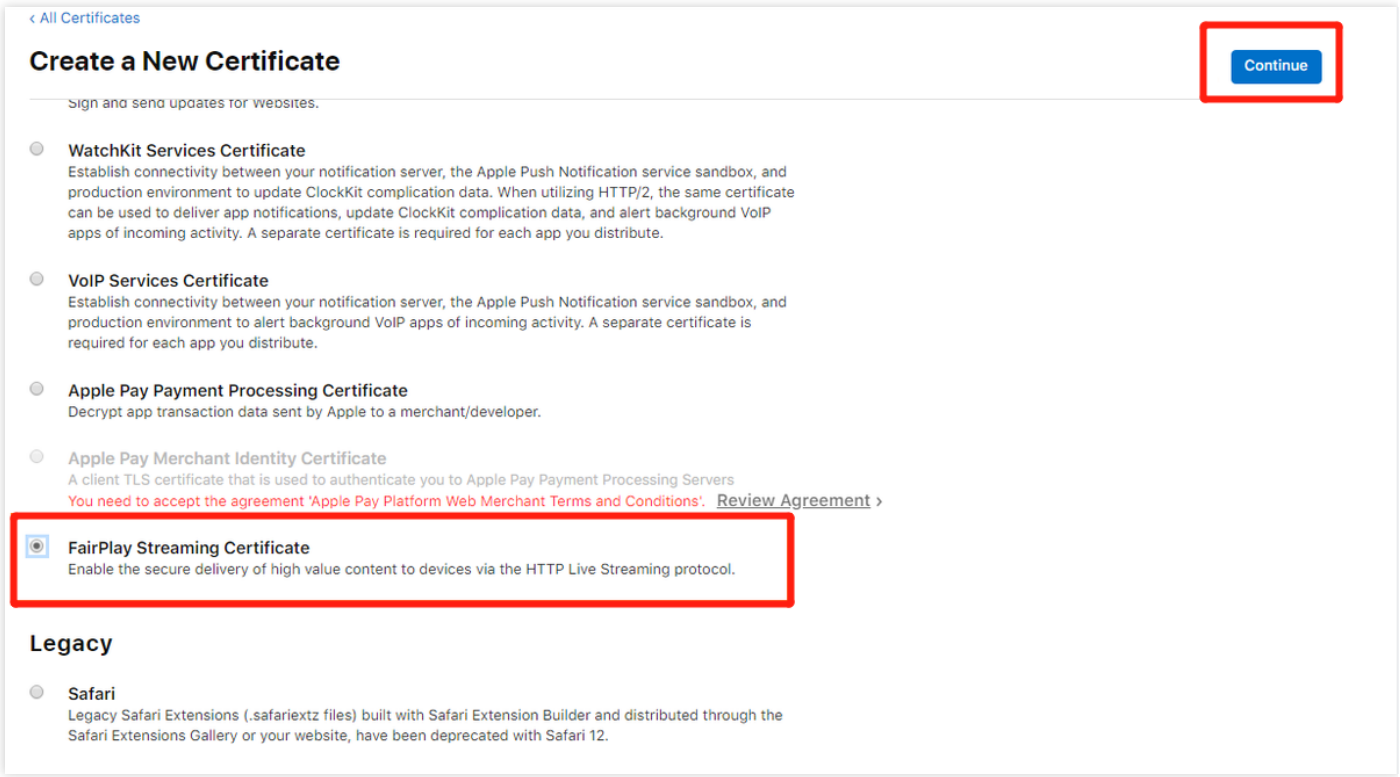
1. 访问到 苹果开发者页面，单击左侧导航栏 `Certificates, Identifiers & Profiles`



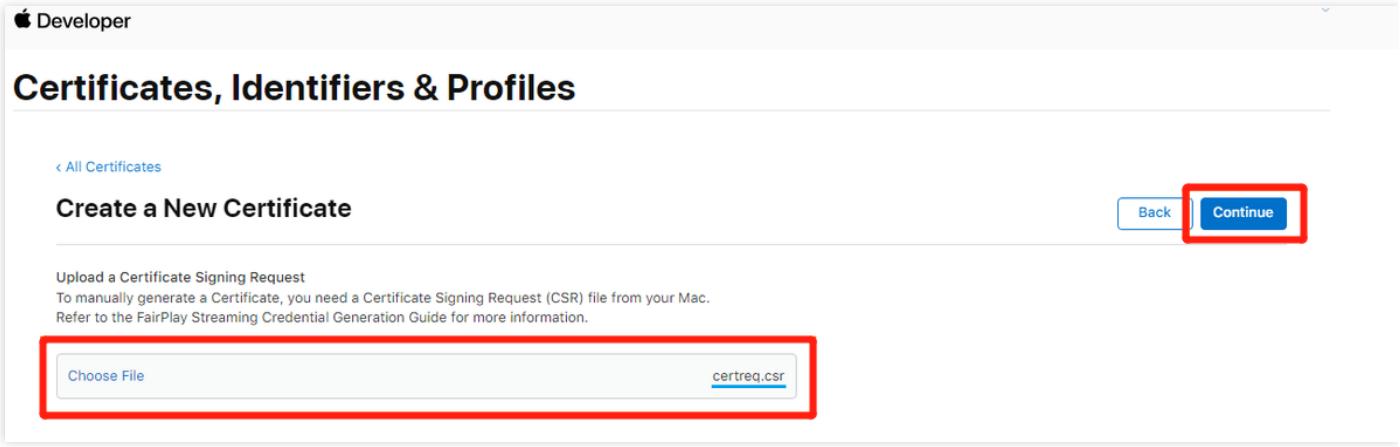
2. 单击页面中的 `+` 按钮。



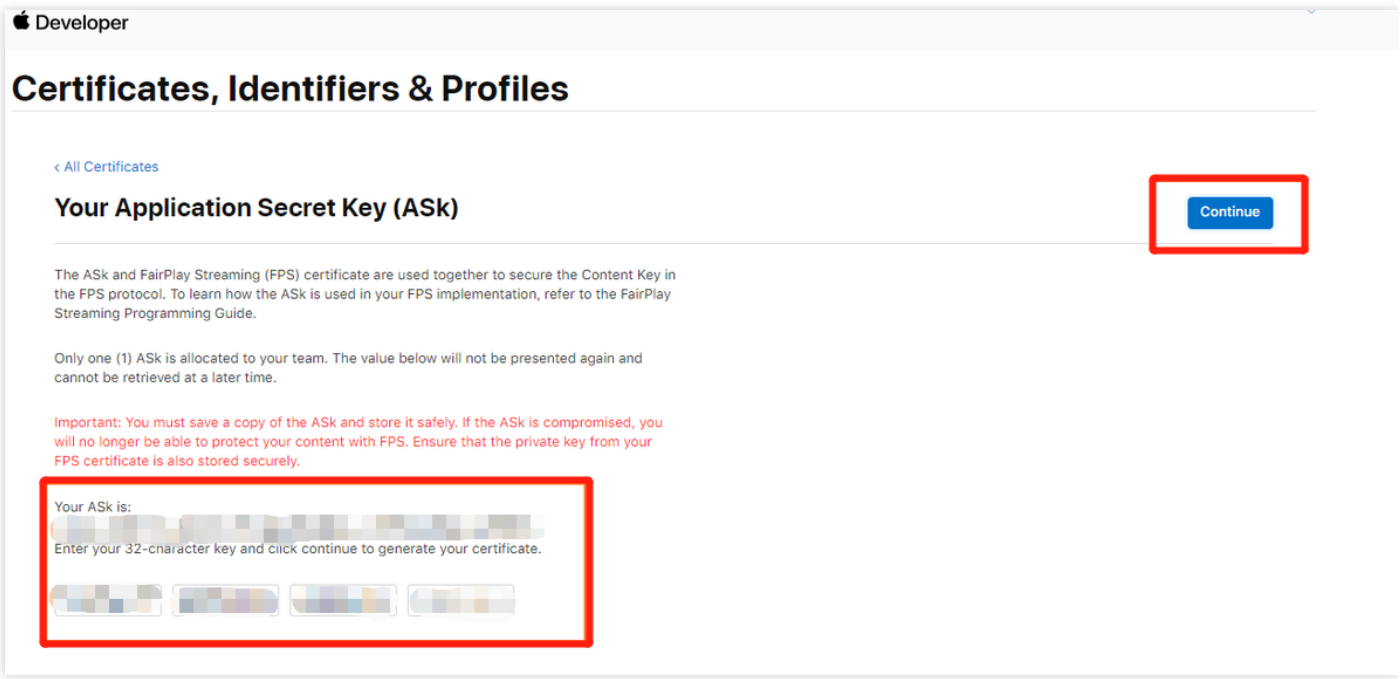
3. 选择页面中的 `FairPlay Streaming Certificate` 选项，并单击 `Continue` 按钮。



4. 单击页面中的 `Choose File` 按钮，选择在上一步中创建的 `certreq` 文件，并单击 `Continue` 按钮。

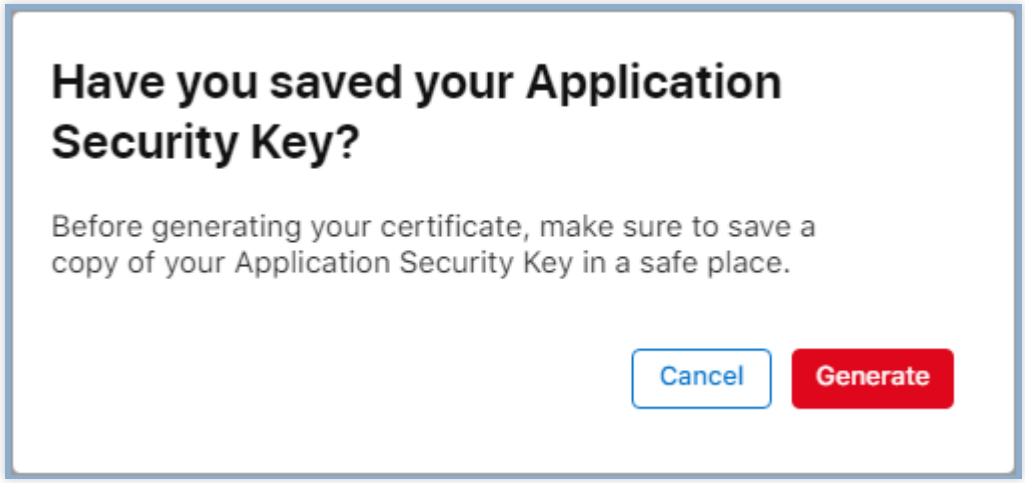


5. 将页面中的 `Application Secret Key (ASK)` 拷贝并备份，接着将 `ASK` 在下方输入栏中重新输入，并单击 `Continue` 按钮。

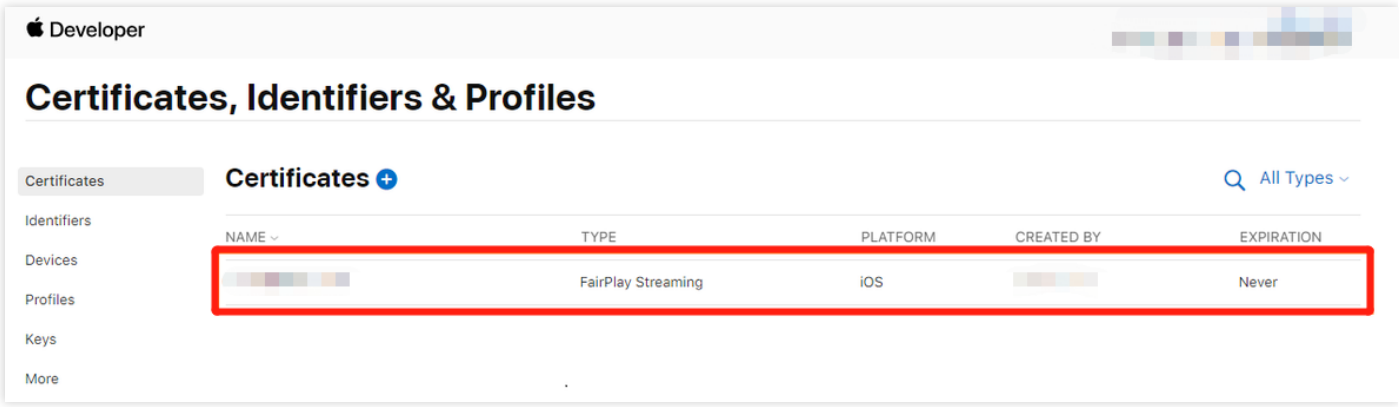


6. 上一步结束后，会出现一个弹框，让您再次确认是否已将 `ASK` 备份，确认已备份后，单击 `Generate` 按钮。

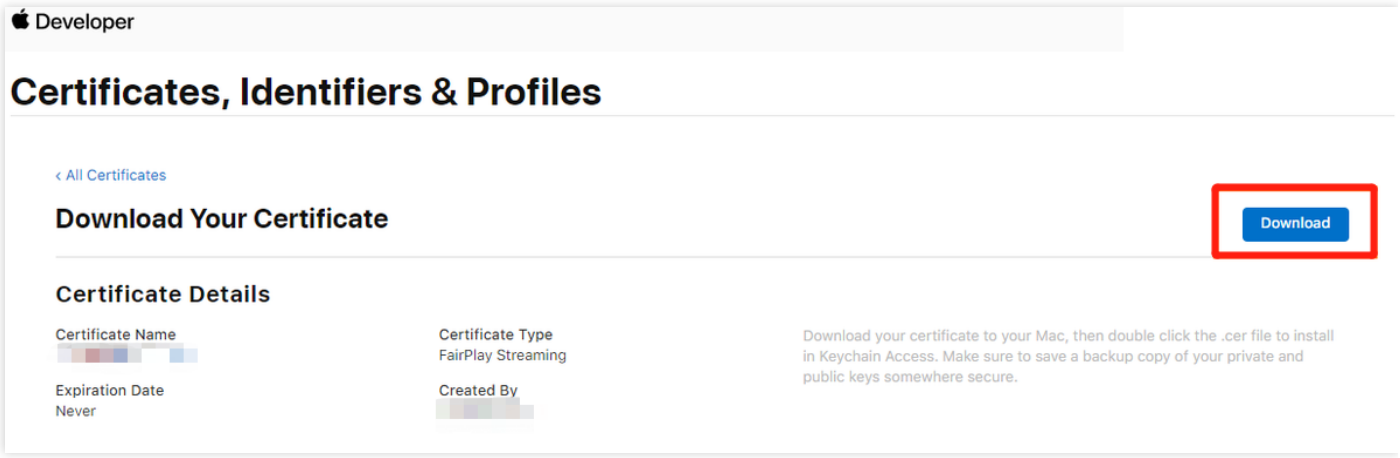
> 注意：
> 请务必确认已将 ASK 备份，此步骤完成后将无法再次查询 ASK。



7. 当以上步骤完成后，证书列表页面中将出现刚才所创建的 FPS 证书，并且证书类型为 `FairPlay Streaming` 。



8. 单击 `Download` 按钮下载 FPS 证书（ `fairplay.cer` ）



总结

至此，您已经完成了 `FairPlay` 证书信息的申请。

云点播 DRM 集成方案
如何在点播控制台提交 FairPlay 证书信息

最近更新时间：2022-08-31 17:54:01

本文介绍在腾讯云点播控制台，提交以下 FairPlay 证书信息：

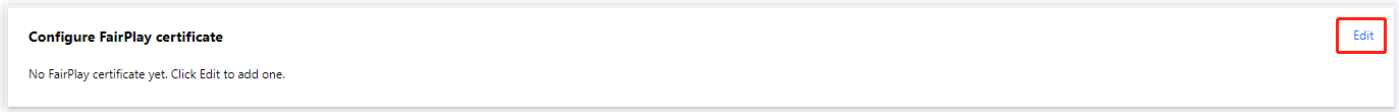
- FPS 证书文件（.cer）
- 私钥文件（.pem）
- 私钥密码
- ASK（Application Secret Key）

如果您还没有申请以上 FairPlay 证书信息，请参考 如何申请 FairPlay 证书信息。

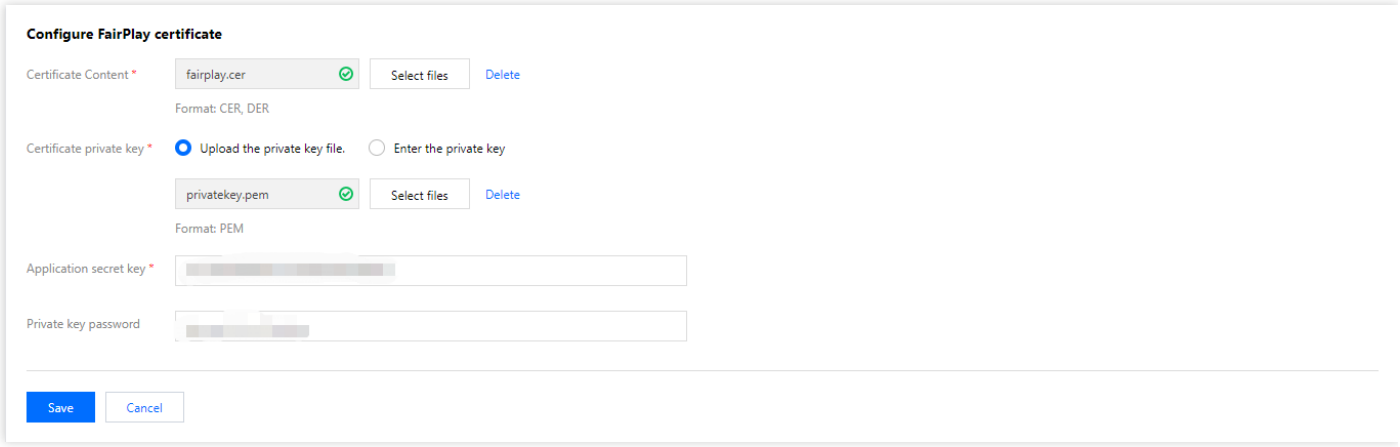
操作步骤

1. 登录腾讯云点播控制台。

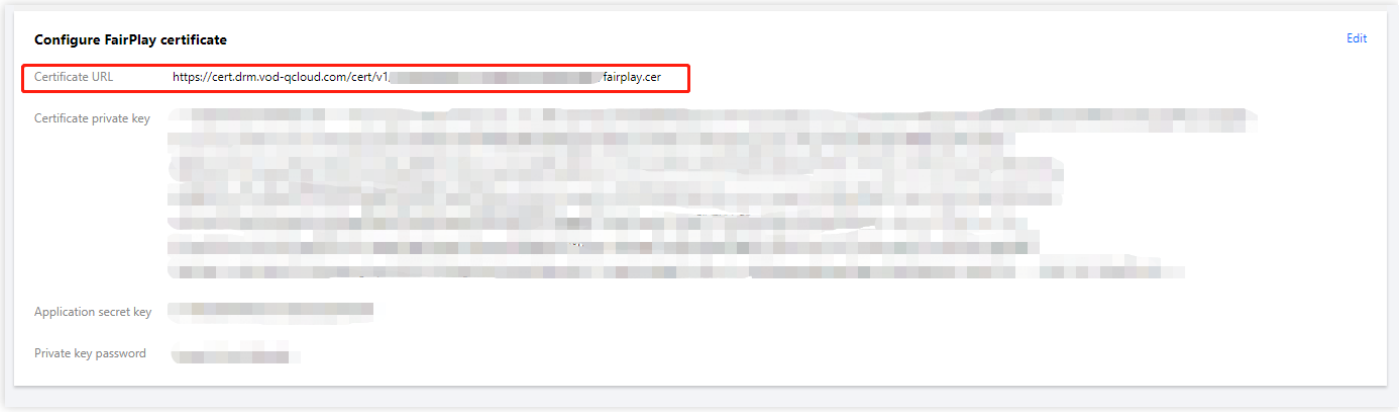
2. 点击展开左侧导航栏 `媒体处理设置` ，点击 `商业级 DRM 加密设置` ，点击右侧 `编辑` 。



3. 设置 FPS 证书信息，包括证书文件（ `fairplay.cer` ）、私钥文件（ `privatekey.pem` ）、私钥密码、ASK，并点击 `保存` 。



4. 保存之后，可以看到 `FairPlay` 的证书信息。



总结

至此，您已经在腾讯云点播控制台完成了 `FairPlay` 证书信息的配置。

播放 DRM 加密视频

最近更新时间：2022-10-27 12:09:18

学习目标

学习本阶段教程，您将了解并掌握如何对视频进行 DRM 加密，并使用播放器播放加密后的视频 。

前置条件

在开始本教程之前，请您确保已满足以下前置条件。

开通云点播

您需要开通云点播，步骤如下：

1. 注册 腾讯云账号，并完成 实名认证。
2. 购买云点播服务，具体请参见 计费概述。
3. 选择 **云产品>视频服务>云点播**，进入云点播控制台。

至此，您已经完成了云点播的开通步骤。

申请 FairPlay 证书信息

请参考 如何申请 FairPlay 证书信息。

提交 FairPlay 证书信息

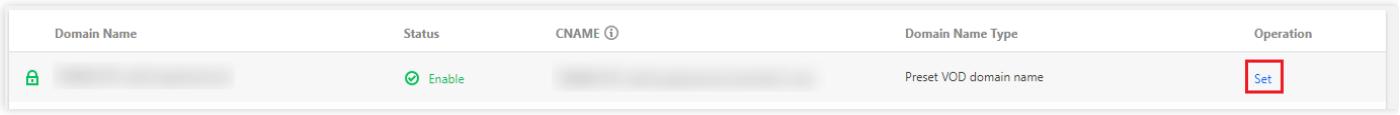
请参考 如何在点播控制台提交 FairPlay 证书信息。

步骤1：开启防盗链

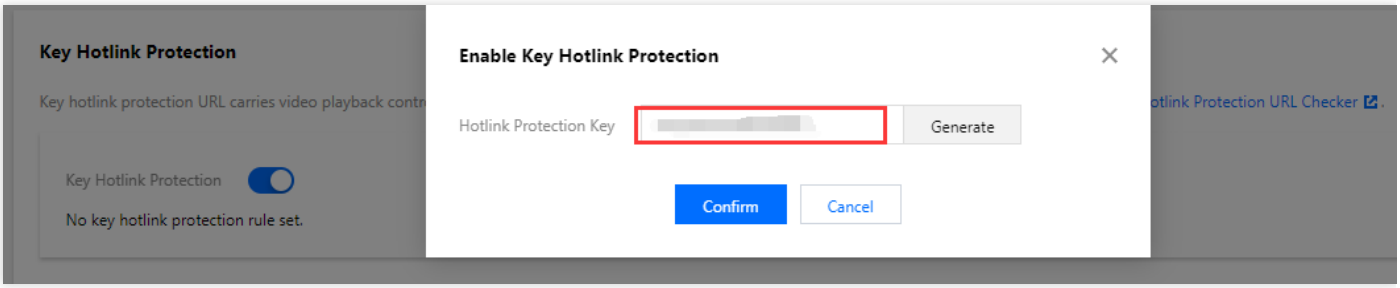
以您账号下的默认分发域名开启 Key 防盗链为例：

> 说明：
> 请避免直接对正在使用的现网域名开启防盗链，否则可能造成现网的视频无法播放。

1. 登录云点播控制台，选择【分发播放设置】>【域名管理】，单击“默认分发域名”的【设置】，单机【访问控制】，进入设置页面。

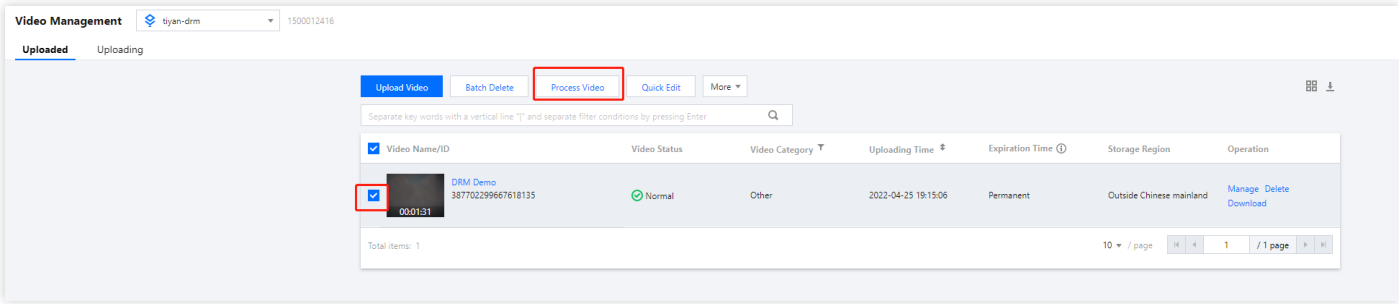


2. 打开【启用 Key 防盗链】，并单击【生成随机 Key】生成一个随机的 Key，本教程为 `vodtestkey`，将生成好的 Key 复制下来，然后单击【确定】保存生效。防盗链 Key 可用于后续步骤中生成播放器签名。



步骤2：对视频进行 DRM 加密

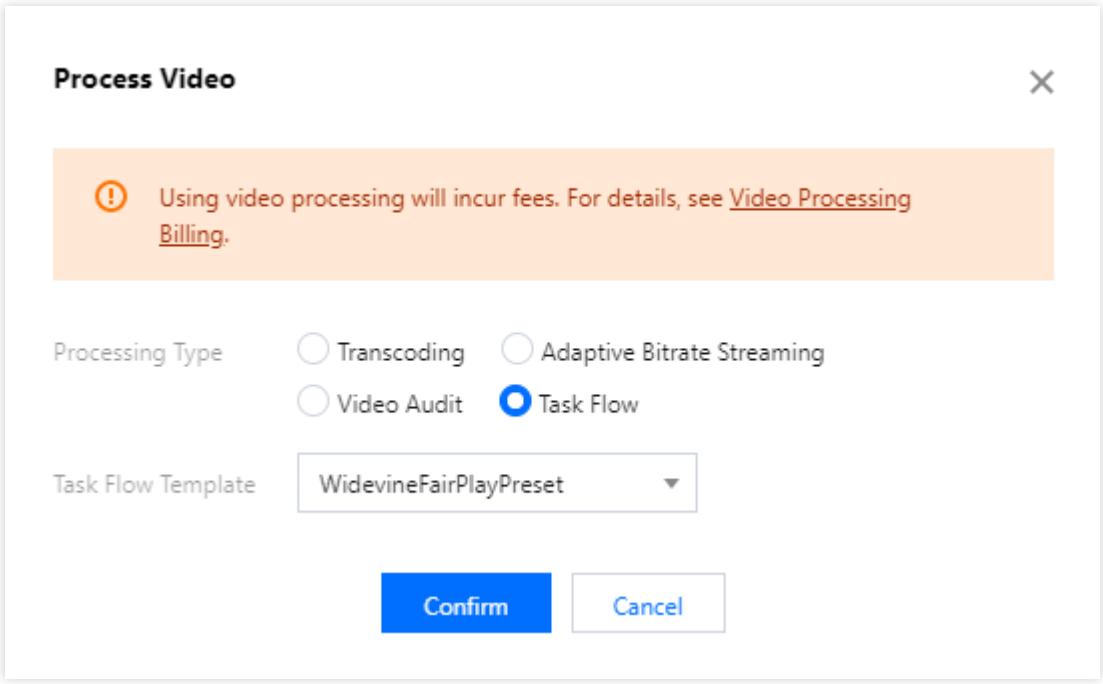
1. 登录云点播控制台，选择 **媒资管理**>**视频管理**，勾选要处理的视频（FileId 为387702304941991610），单击 **视频处理**。



2. 在视频处理界面：

- **处理类型** 选择 **任务流**。

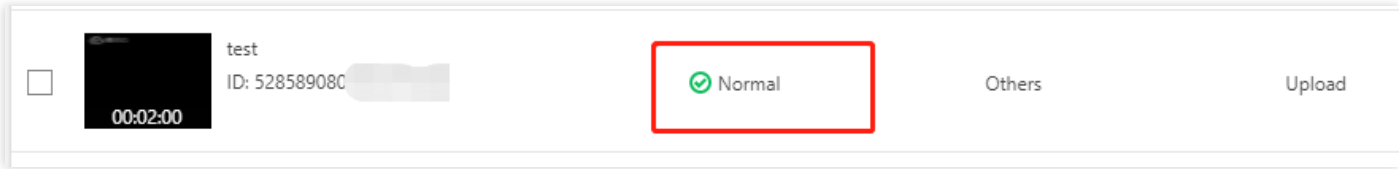
- **任务流模板** 选择 **WidevineFairPlayPreset**。



说明：

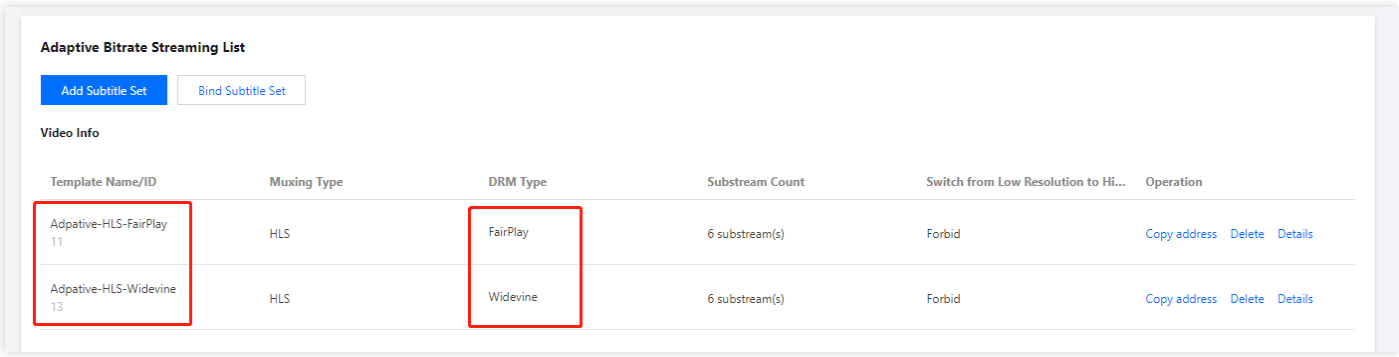
- WidevineFairPlayPreset 是预置任务流：分别使用11、13模板转自适应码流，10模板截图做封面，10模板截雪碧图。
- 11模板自适应码流是加密类型为 `FairPlay` 的多码率输出，13模板自适应码流是加密类型为 `Widevine` 的多码率输出。

3. 单击 **确定**，等待“视频状态”栏从“处理中”变为“正常”，表示视频已处理完毕：

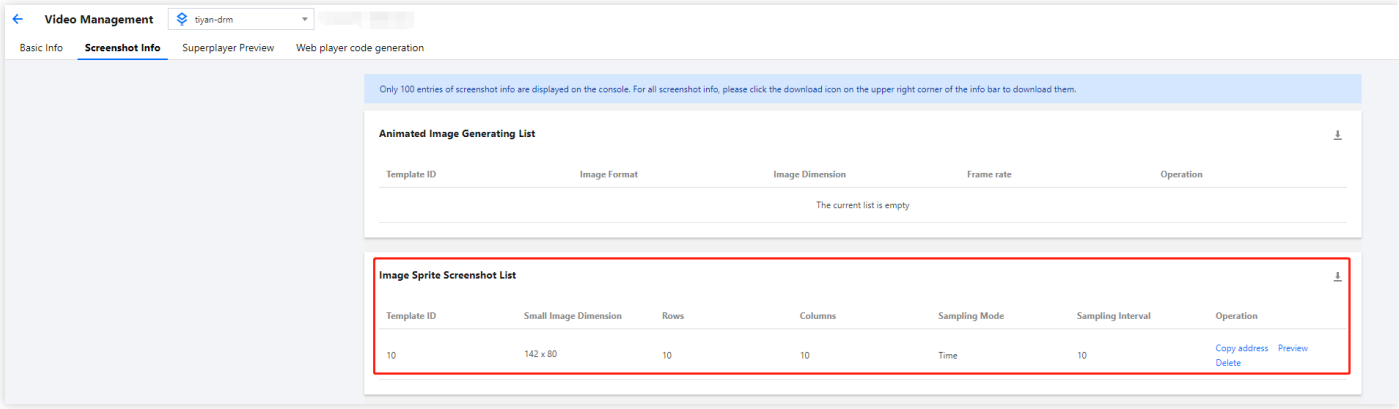


4. 单击视频“操作”栏下的 **管理**，进入管理页面：

- 选择“基本信息”页签，可以看到生成的封面，以及 DRM 加密的自适应码流输出（模板 ID 为11和13）。



- 选择“截图信息”页签，可以看到生成的雪碧图（模板 ID 为10）。



步骤3：生成播放器签名

播放器签名，用于后续查询播放信息，生成方式请参考 播放器签名文档 。 本教程的播放器签名的 PayLoad 如下：

```
{
"appId": 1500014561,
"fileId": "387702304941991610",
"currentTimeStamp": 1661163373,
"expireTimeStamp": 2648557919,
"pcfg":"advanceDrmPreset"
}
```

本教程的 Key 为 `vodtestkey` 时，生成的播放器签名（ `psign` ）如下：

```
eyJhbGciOiJIUzI1NiJ9.eyJhcHBJZCI6MTUwMDAxNDU2MSwiZmlsZUlkIjoiMzg3NzAyMzA0OTQxOTkxNj
EwIiwiY3VycmVudFRpbWVTdGFtcCI6MTY2MTE2MzM3MywiZXhwaXJlVGltZVN0YW1wIjoyNjQ4NTU3OTE5LC
JwY2ZnIjoiYWR2YW5jZURybVByZXNldCJ9.rEZLhjgsoLc2htIUI_HckxvhVmdBhQyf5d-2Kku1JeA
```

步骤4：使用播放器播放 DRM 加密视频。

Web 端

使用点播播放器播放

您只需在初始化播放器时传入必要的播放文件参数即可播放 DRM 加密视频。

step 1：在页面中引入文件

在适当的地方引入播放器样式文件与相关脚本文件：

```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/tcplayer.min.css" rel="stylesheet"/>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/TXLivePlayer-1.2.3.min.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/hls.min.1.1.5.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/flv.min.1.6.3.js"></script>

<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/dash.all.min.4.4.1.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/tcplayer.v4.5.4.min.js"></script>
```

step 2：放置播放器容器

在需要展示播放器的页面位置加入播放器容器，代码如下：

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-playsinline>
</video>
```

> 说明：
> 容器 ID 以及宽高都可以自定义。

step 3：初始化代码

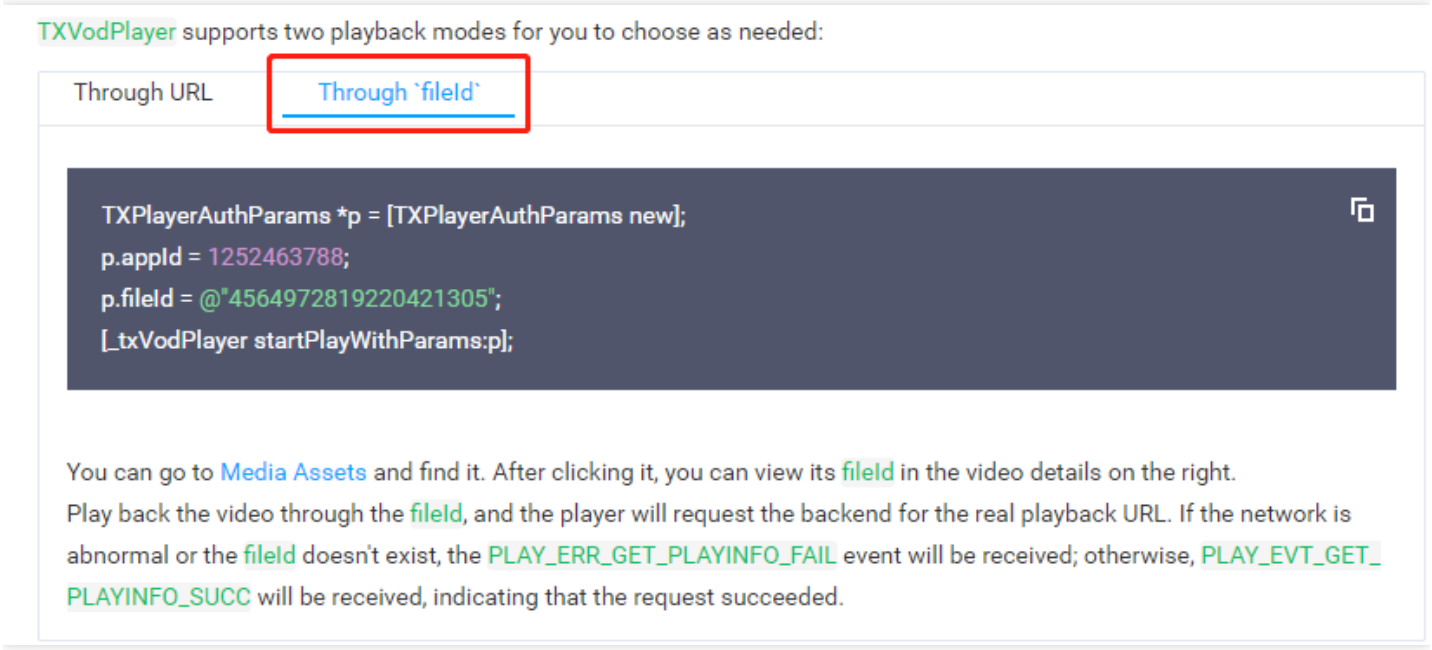
在页面初始化的代码中加入以下初始化脚本，传入必须的初始化参数（其中包含步骤3中生成的播放器签名 `psign` ），代码如下：

```
var player = TCPlayer('player-container-id', {
appID: '1500014561', // 请传入点播账号的appID （必须）
fileID: '387702304941991610', // 请传入需要播放的视频filID （必须）
psign: 'eyJhbGciOiJIUzI1NiJ9.eyJhcHBJZCI6MTUwMDAxNDU2MSwiZmlsZUlkIjoiMzg3NzAyMzA0OTQxOTkxNjEwIiwiY3VycmVudFRpbWVTdGFtcCI6MTY2MTE2MzM3MywiZXhwaXJlVGltZVN0YW1wIjoyNjQ4NTU3OTE5LCJwY2ZnIjoiYWR2YW5jZURybVByZXN1dCJ9.rEZLhjgsoLc2htIUI_HckxvhVmdBhQyf5d-2Kku1JeA',
// 其他参数请在开发文档中查看 https://intl.cloud.tencent.com/document/product/266/39105
});
```

iOS 端

请参考 接入指引 （通过 FileId 方式）播放 DRM 加密视频。其中，这一过程中需要使用到步骤3中生成的播放器签名 `psign` 。

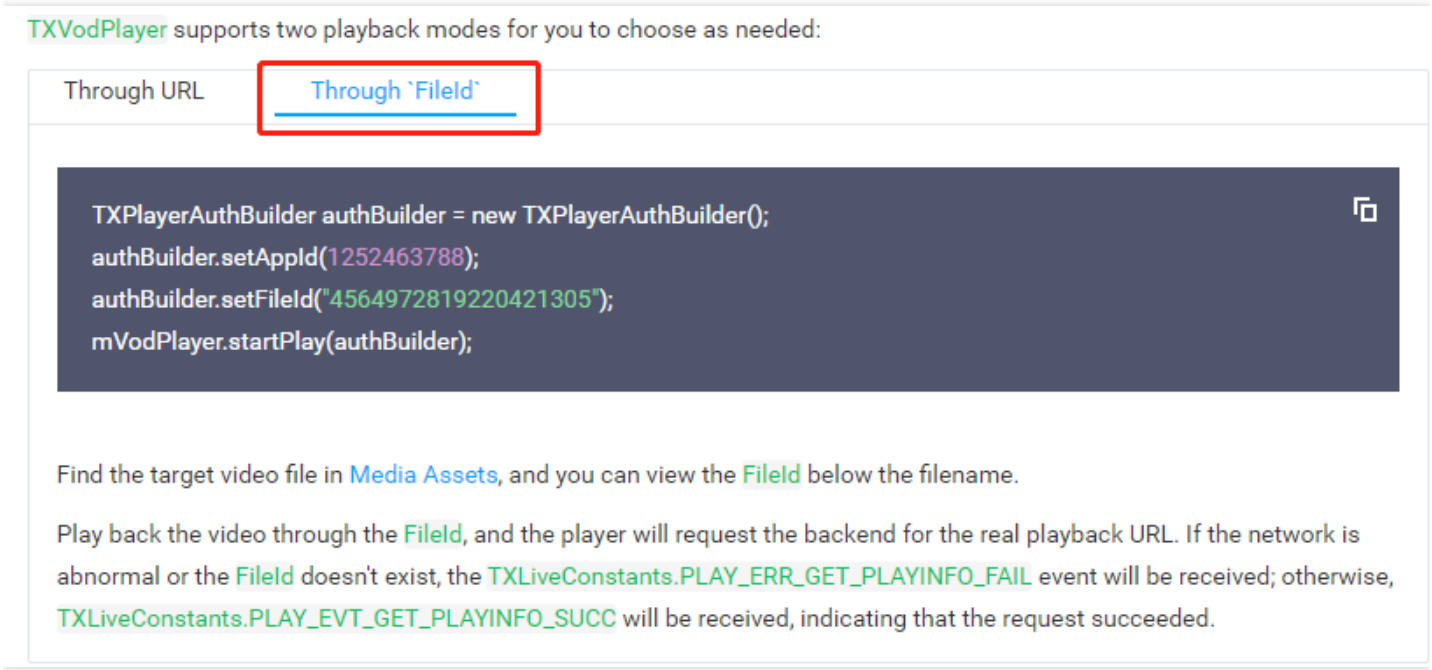
> 说明：
> 在接入前，请您提交工单 联系我们获取支持 DRM 功能的 SDK 。



Android 端

请参考 接入指引（通过 FileId 方式）播放 DRM 加密视频。其中，这一过程中需要使用到步骤3中生成的播放器签名 `psign` 。

说明：
在接入前，请您提交工单 联系我们获取支持 DRM 功能的 SDK 。



总结

学习本教程后，您已经掌握如何对视频进行 DRM 加密，并使用播放器播放加密后的视频。

说明：
在您对接 DRM 或者华曦达的过程中的任何问题，都可以提工单联系我们，我们全程负责帮您解决。

第三方 DRM 集成方案（华曦达）
如何在华曦达控制台提交 FairPlay 证书信息

最近更新时间：2022-08-31 17:54:01

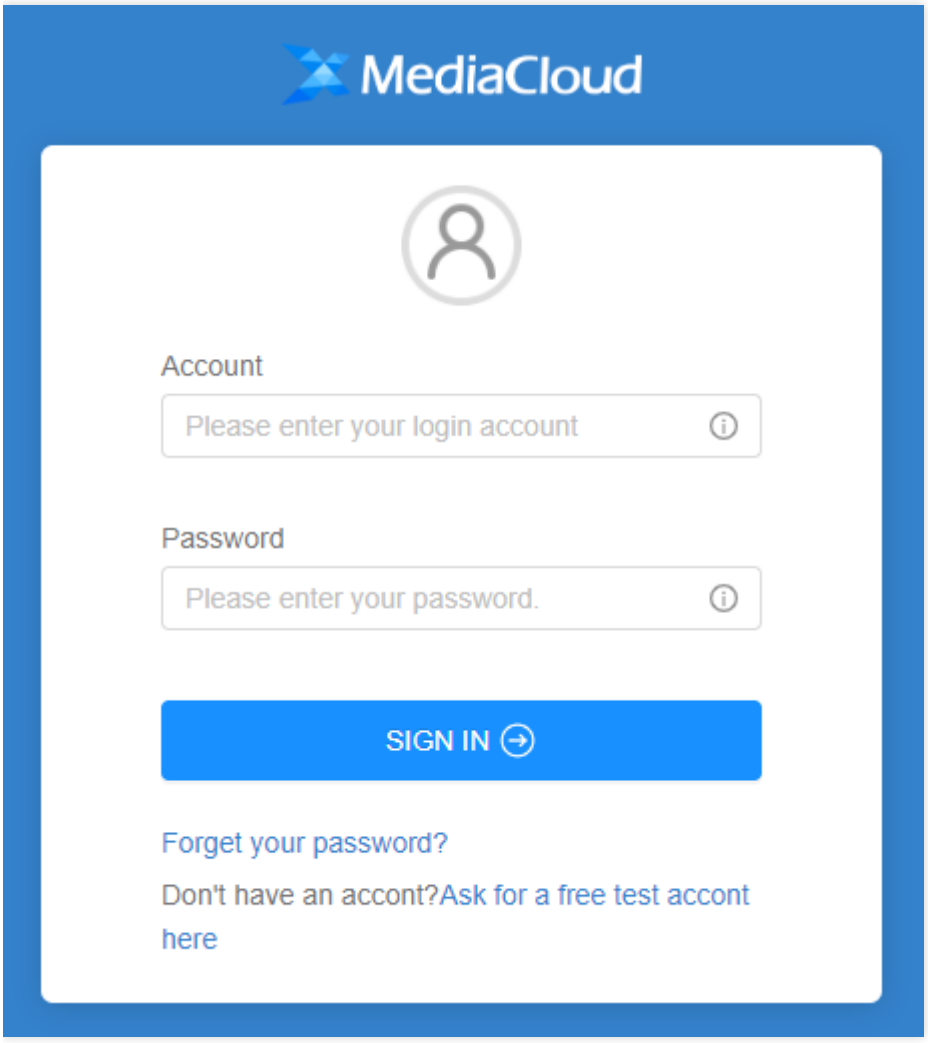
本文介绍在华曦达（SDMC）控制台，提交以下 FairPlay 证书信息：

- FPS 证书文件（.cer）
- 私钥文件（.pem）
- 私钥密码
- ASK（Application Secret Key）

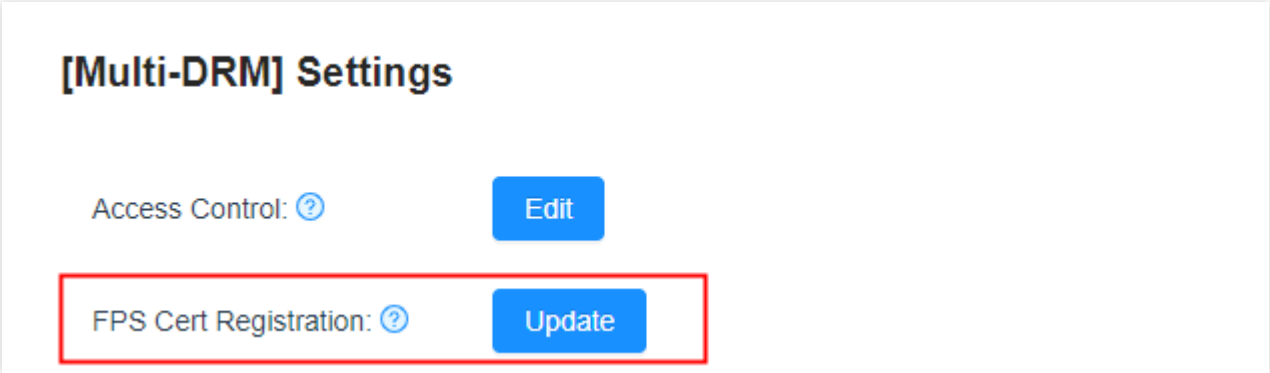
如果您还没有申请以上 FairPlay 证书信息，请参考 如何申请 FairPlay 证书信息。

操作步骤

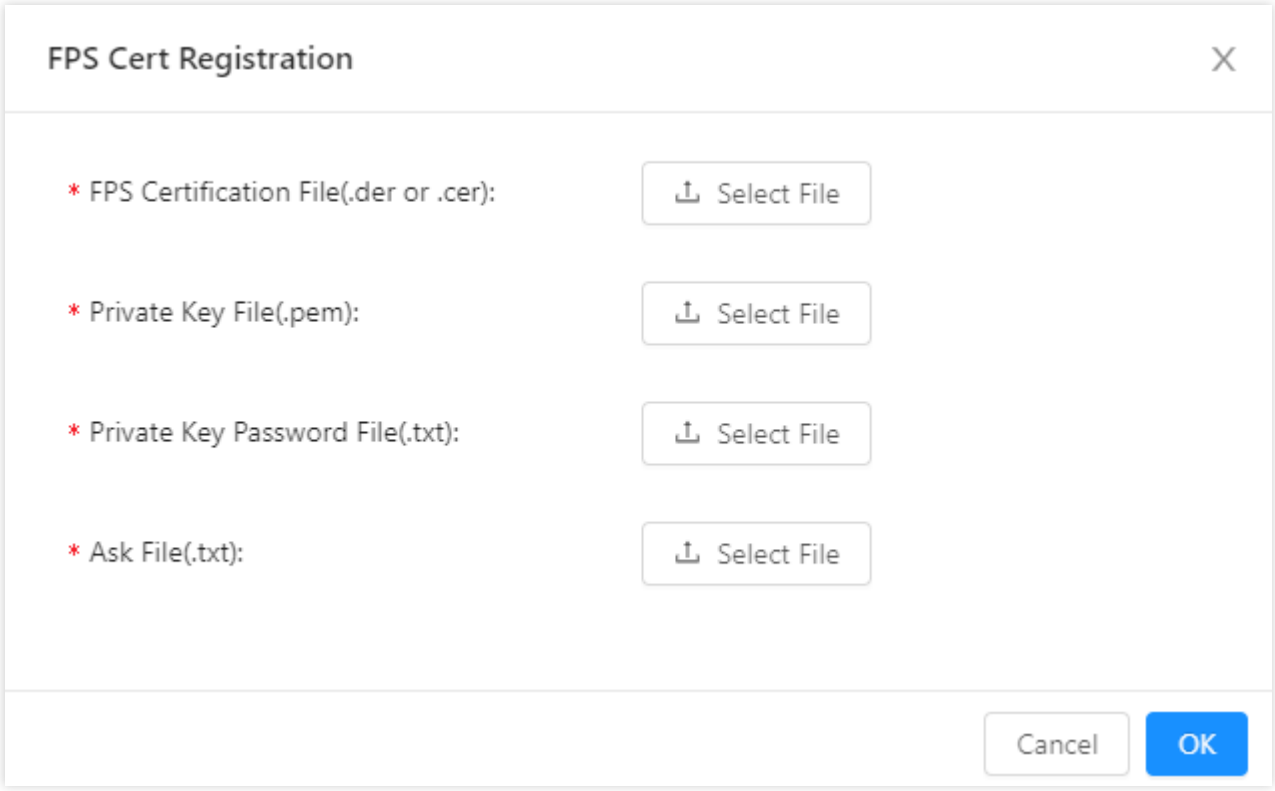
1. 登录 华曦达控制台。



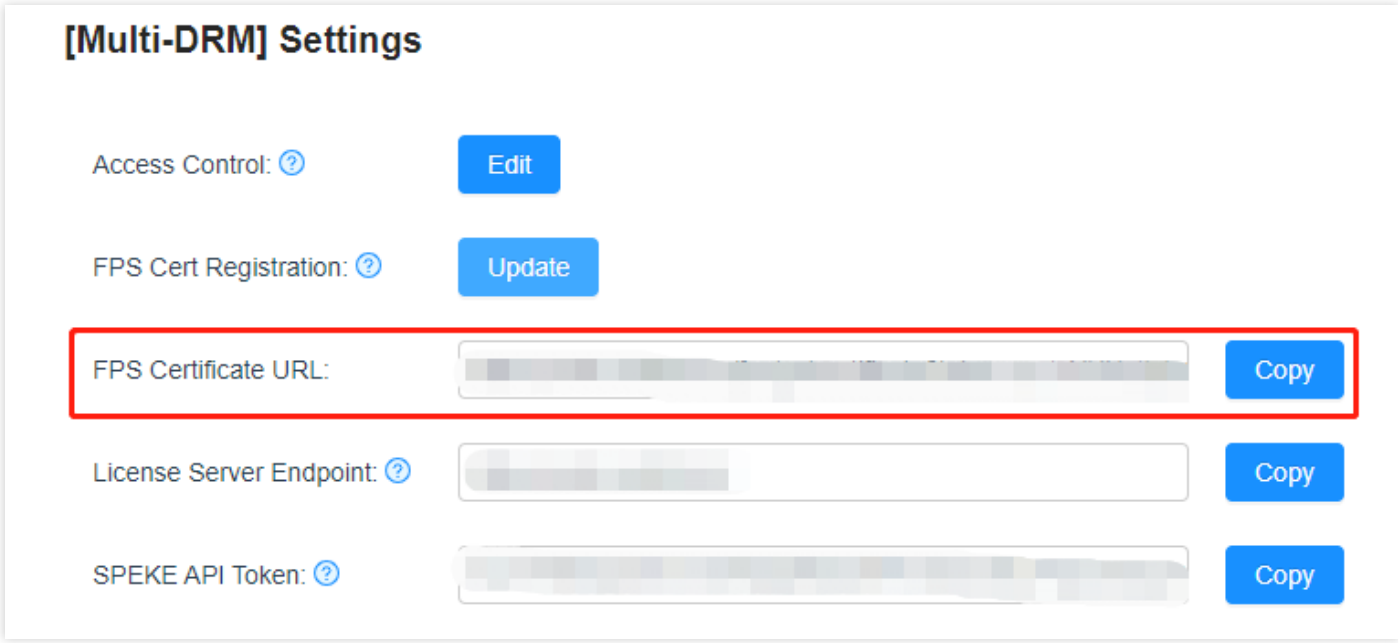
2. 点击展开左侧导航栏 `DRM 设置`， 进入到 FPS 证书注册，单击更新按钮进行证书上传 。



3. 上传 FPS 证书文件、私钥文件、私钥密码文件和 ASK 文件，然后单击 **OK** 进行上传。



4. 上传完成后，您将可以查看到 FPS 证书地址信息。



总结

至此，您已经在华曦达控制台完成了 `FairPlay` 证书信息的配置。

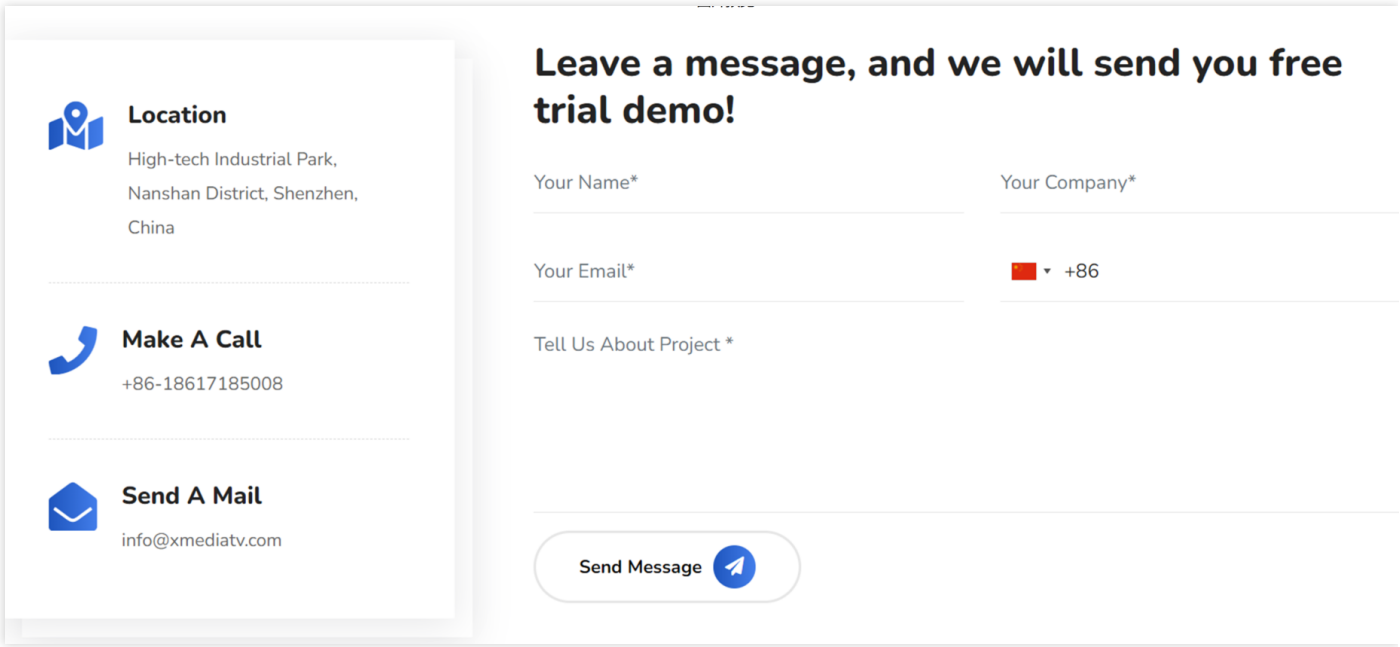
设置华曦达用户密钥信息

最近更新时间：2022-08-31 17:54:01

如果使用第三方 DRM 密钥提供商（华曦达 SDMC）集成 DRM 功能，那么需要从华曦达获取相关的用户密钥信息（UID、SecretID 、SecretKey 以及 FairPlay 证书地址）。因此，本文主要说明如何在华曦达控制台获取用户密钥信息。

操作步骤

1. 如果您没有华曦达账号，则需先完成注册，可访问 华曦达（SDMC）官网 。



2. 填写完右侧表单后，点击 **Send message**，一般几个小时后，您将收到由华曦达发送的系统邮件反馈，后续将会有华曦达商务人员联系您并确认相关信息。

Thanks for your interesting in our solution. This is XMediaTV Business Team from SDMC, taking charge of OTT platform & Cloud business. Glad to work togehter with you.

Regarding to your project, we have the corresponding solution to meet your requirements and we also have delivered many global similiar projects successfully. May i know your whatsapp or Skype to discuss directly?

who we are:

SDMC is invested by Tencent, which is one of the biggest internet company in China. SDMC and Tencent joint together launched Cloud Online Video SaaS platform TXMedia to provide high quality video streaming service to customers.

Grow your video business in:

We glad to invite you to video and audio streaming service community, here we freely sharing and answering any streaming related questions, helping you empower your video business by content share, white-label platform, video revenue model, cutting-edge infos....come and talk to our experts in:
Whatsapp group → https://chat.whatsapp.com/
Facebook group → https://www.facebook.com/
Linkedin group → https://www.linkedin.com/

Know our streaming solution first!
SDMC XMediaTV Introduction V4.7 20210626.pdf

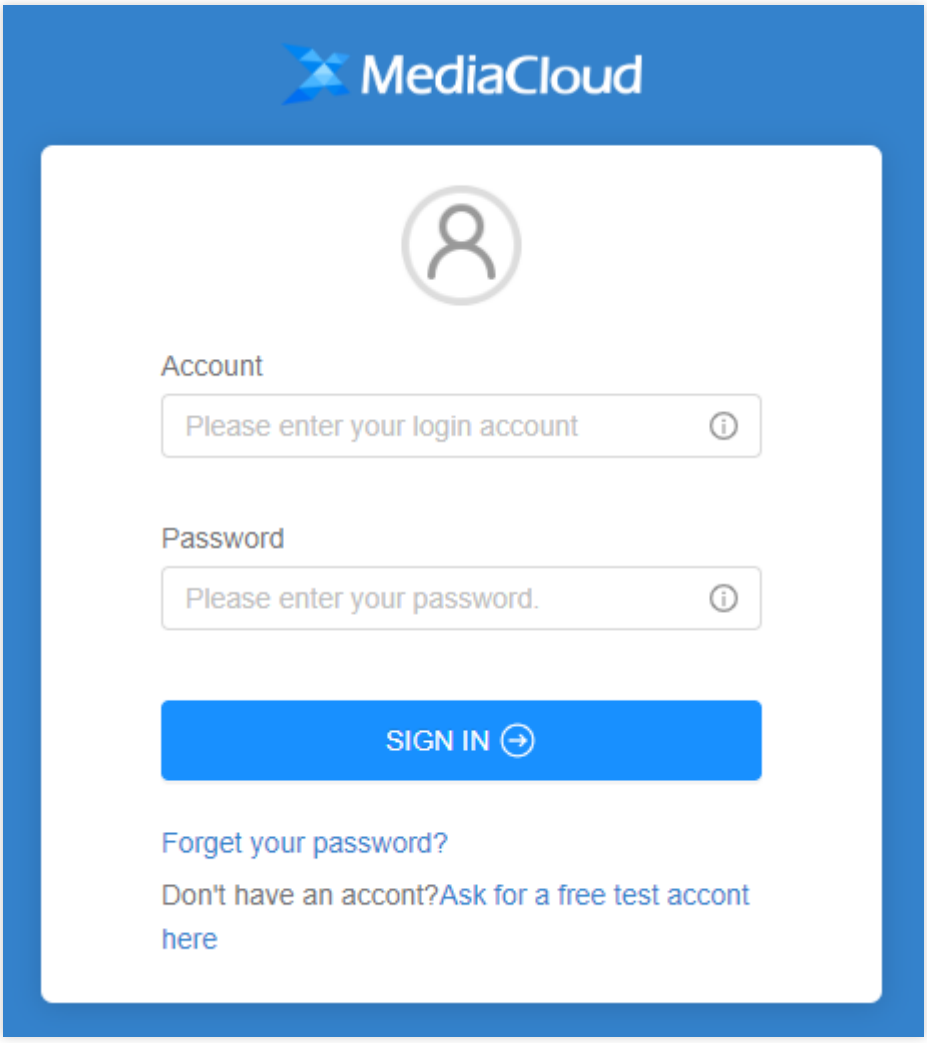
Please grab some time with me whenever you're ready - Schedule a Meeting

Best regards.

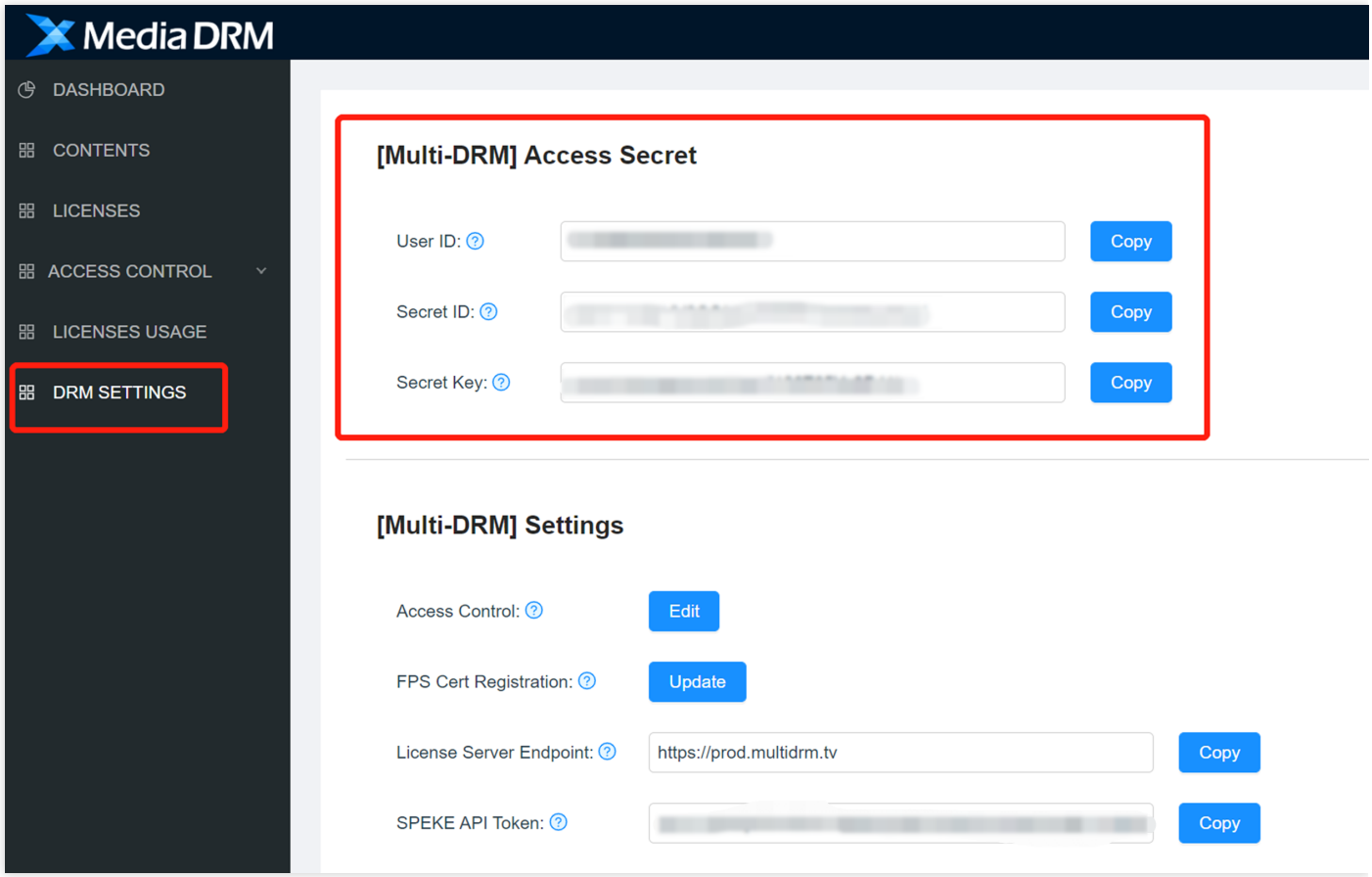
Xmedia Business Team

3. 华曦达审核通过后会给您发送 DRM 服务开通邮件，邮件中包含华曦达 DRM 控制台地址及登录初始密码。

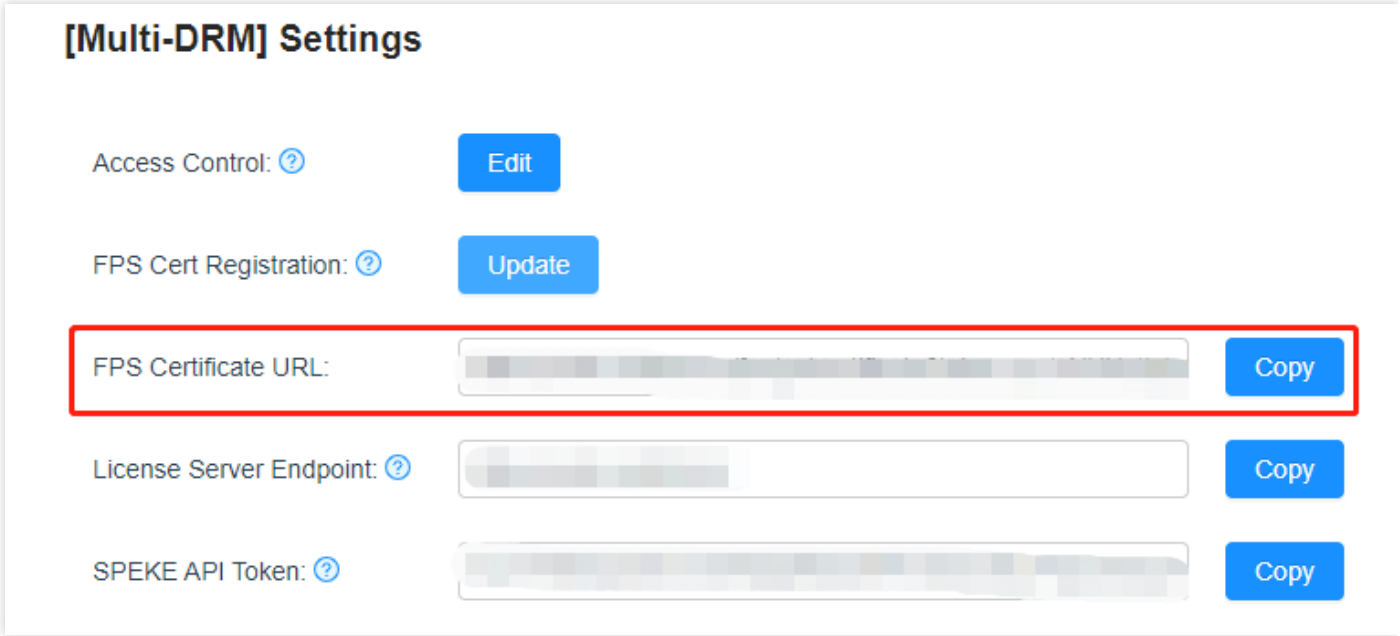
4. 登录 华曦达 DRM 控制台，输入账号及密码即可成功登录。



5. 进入控制台点击左侧导航栏 `DRM SETTING` ，查询用户密钥信息 UID、SecretID 及 SecretKey。



6. 查询得到 FairPlay 证书地址。

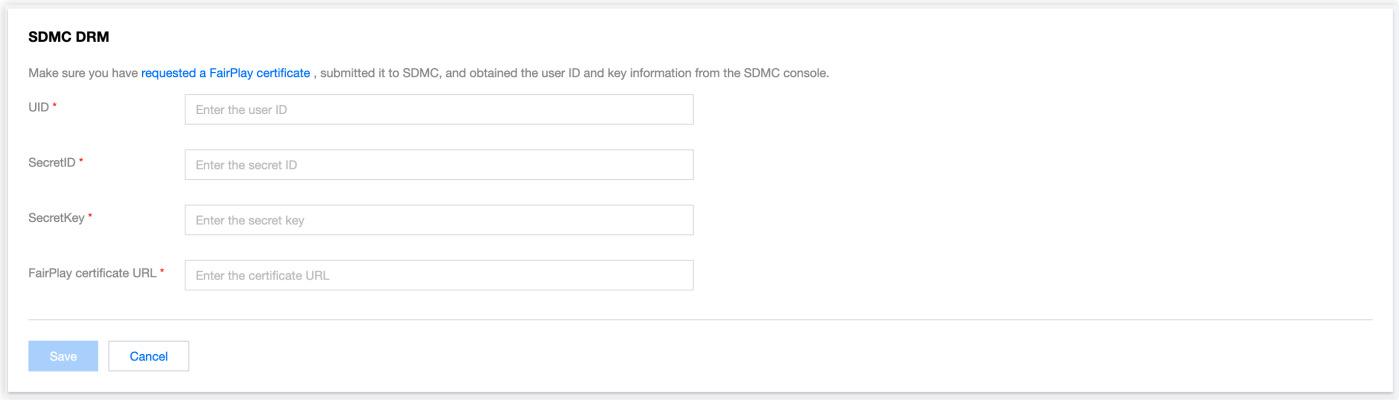


7. 登录腾讯云点播控制台。

8. 提交华曦达的用户密钥信息（包括Uid、SecretID、SecretKey 以及 FairPlay 证书地址）。

点击展开左侧导航栏 视频处理设置 ，点击 商业级 DRM 配置 ，点击右侧 华曦达用户密钥信息配置 中的 编辑 ：

填写用户密钥信息并保存：



总结

至此，您已在腾讯云点播控制台完成了华曦达用户密钥信息的配置。

> 说明：
> 在您对接 DRM 或者华曦达的过程中的任何问题，都可以提工单 联系我们，我们全程负责帮您解决。

播放 DRM 加密视频

最近更新时间：2022-09-13 10:47:49

学习目标

学习本阶段教程，您将了解并掌握如何对视频进行 DRM 加密，并使用播放器播放加密后的视频 。

前置条件

在开始本教程之前，请您确保已满足以下前置条件。

开通云点播

您需要开通云点播，步骤如下：

1. 注册 腾讯云账号，并完成 实名认证。
2. 购买云点播服务，具体请参见 计费概述。
3. 选择 **云产品>视频服务>云点播**，进入云点播控制台。

至此，您已经完成了云点播的开通步骤。

申请 FairPlay 证书信息

请参考 如何申请 FairPlay 证书信息。

提交 FairPlay 证书信息

请参考 如何在华曦达控制台提交 FairPlay 证书信息。

设置华曦达用户密钥信息

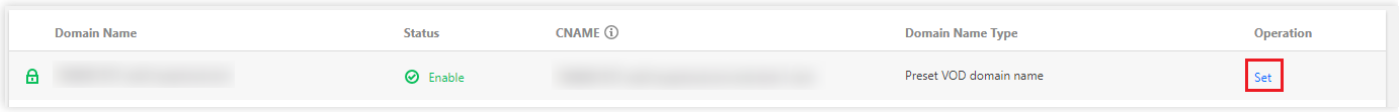
请参考 设置华曦达用户密钥信息。

步骤1：开启防盗链

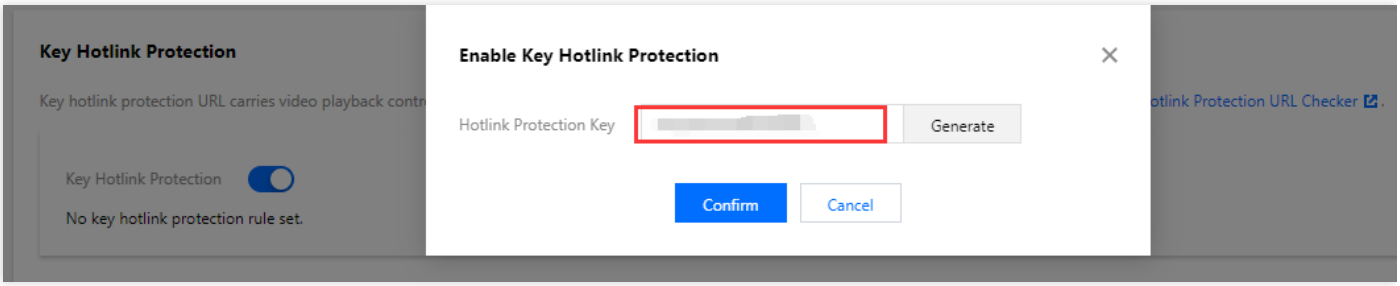
以您账号下的默认分发域名开启 Key 防盗链为例：

> 说明：
> 请避免直接对正在使用的现网域名开启防盗链，否则可能造成现网的视频无法播放。

1. 登录云点播控制台，选择【分发播放设置】>【域名管理】，单击“默认分发域名”的【设置】，单机【访问控制】，进入设置页面。

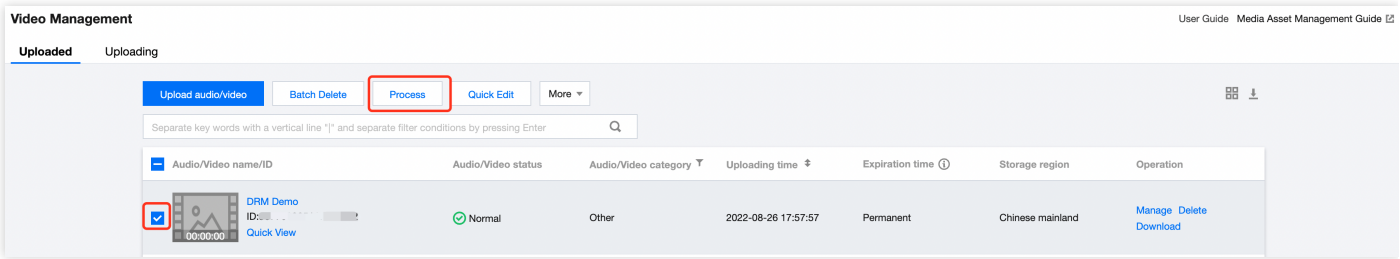


2. 打开【启用 Key 防盗链】，并单击【生成随机 Key】生成一个随机的 Key，本教程为 `vodtestkey`，将生成好的 Key 复制下来，然后单击【确定】保存生效。防盗链 Key 可用于后续步骤中生成播放器签名。



步骤2：对视频进行 DRM 加密

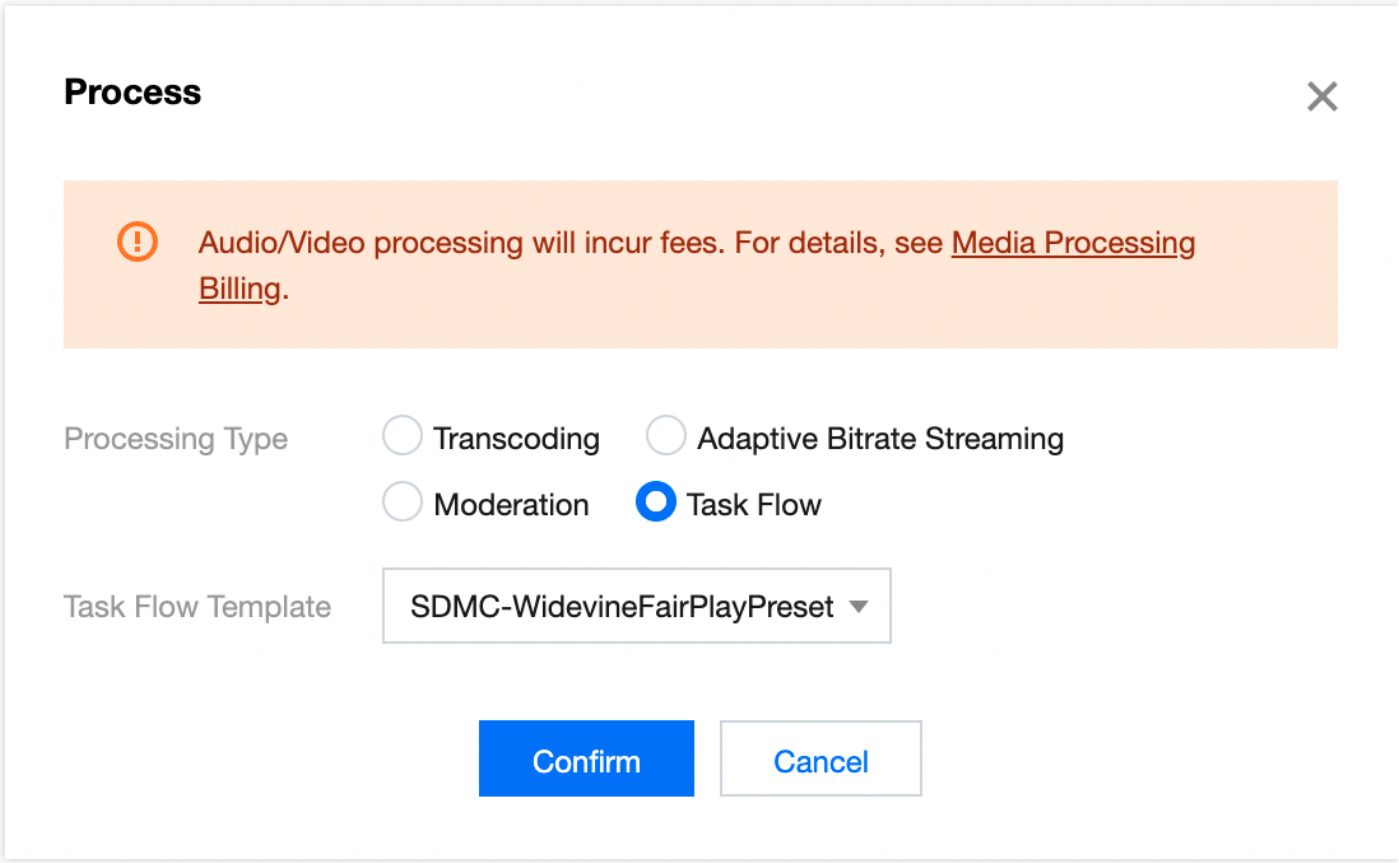
1. 登录云点播控制台，选择 **媒资管理**>**音视频管理**，勾选要处理的视频（FileId 为387702304941991610），单击 **音视频处理**。



2. 在视频处理界面：

- **处理类型** 选择 **任务流**。

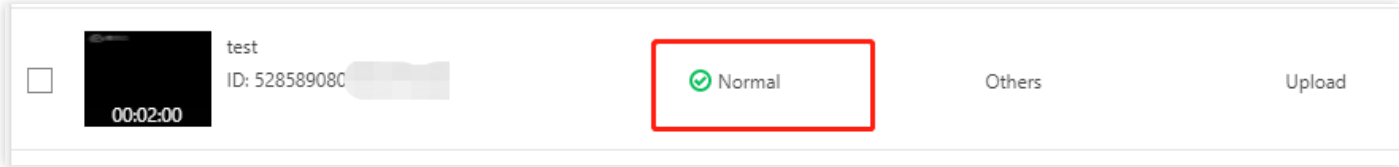
- **任务流模板** 选择 **SDMC-WidevineFairPlayPreset**。



说明：

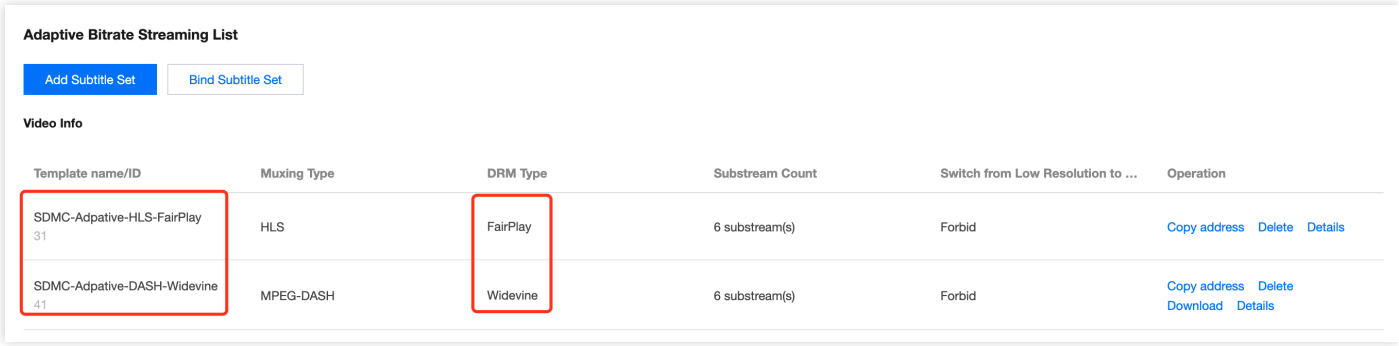
- SDMC-WidevineFairPlayPreset 是预置任务流：分别使用31、41模板转自适应码流，10模板截图做封面，10模板截雪碧图。
- 31模板自适应码流是加密类型为 `FairPlay` 的多码率输出，41模板自适应码流是加密类型为 `Widevine` 的多码率输出。

3. 单击 **确定**，等待“视频状态”栏从“处理中”变为“正常”，表示视频已处理完毕：

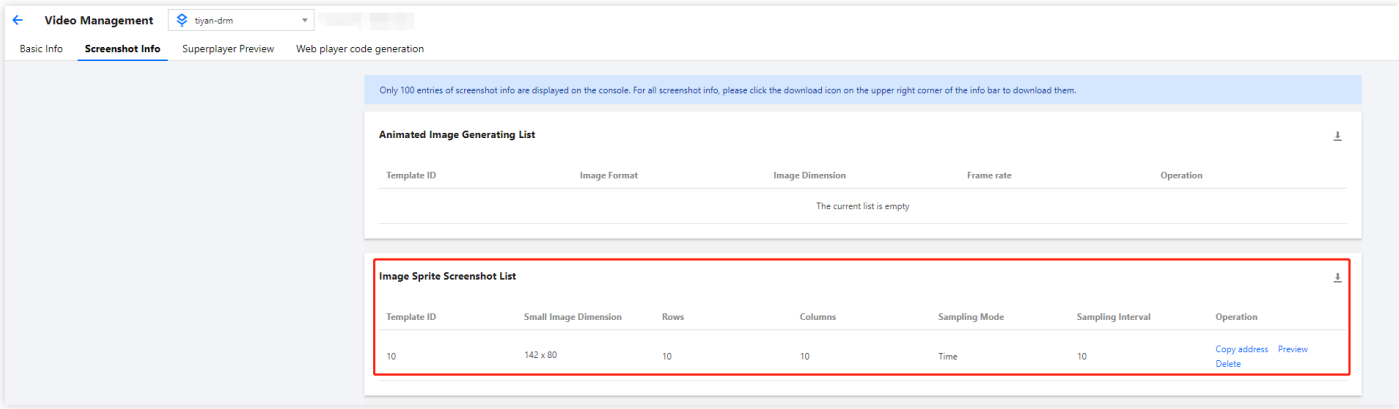


4. 单击视频“操作”栏下的 **管理**，进入管理页面：

- 选择“基本信息”页签，可以看到生成的封面，以及 DRM 加密的自适应码流输出（模板 ID 为31和41）。



- 选择“截图信息”页签，可以看到生成的雪碧图（模板 ID 为10）。



步骤3：生成播放器签名

播放器签名，用于后续查询播放信息，生成方式请参考 播放器签名文档 。 本教程的播放器签名的 PayLoad 如下：

```
{
"appId": 1500014561,
"fileId": "387702304941991610",
"currentTimeStamp": 1661163373,
"expireTimeStamp": 2648557919,
"pcfg": "SDMC-advanceDrmPreset"
}
```

本教程的 Key 为 `vodtestkey` 时，生成的播放器签名（ `psign` ）如下：

```
eyJhbGciOiJIUzI1NiJ9.eyJhcHBJZCI6MTUwMDAxNDU2MSwiZmlsZUlkIjoiMzg3NzAyMzA0OTQxOTkxNj
EwIiwiY3VycmVudFRpbWVTdGFtcCI6MTY2MTE2MzM3MywiZXhwaXJlVGltZVN0YW1wIjoyNjQ4NTU3OTE5LC
```

```
JwY2ZnIjoiU0RNQy1hZHZhbmNlRHJtUHJlc2V0In0.BYdxHHEMH0isrta4ERmksGbfu4cLiwl7f1cu04XV890
```

步骤4：使用播放器播放 DRM 加密视频。

Web 端

使用点播播放器播放

您只需在初始化播放器时传入必要的播放文件参数即可播放 DRM 加密视频。

step 1：在页面中引入文件

在适当的地方引入播放器样式文件与相关脚本文件：

```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/tcplayer.min.css" rel="stylesheet"/>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/TXLivePlayer-1.2.3.min.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/hls.min.1.1.5.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/flv.min.1.6.3.js"></script>

<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/libs/dash.all.min.4.4.1.js"></script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.4/tcplayer.v4.5.4.min.js"></script>
```

step 2：放置播放器容器

在需要展示播放器的页面位置加入播放器容器，代码如下：

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-playsinline>
</video>
```

> 说明：
> 容器 ID 以及宽高都可以自定义。

step 3：初始化代码

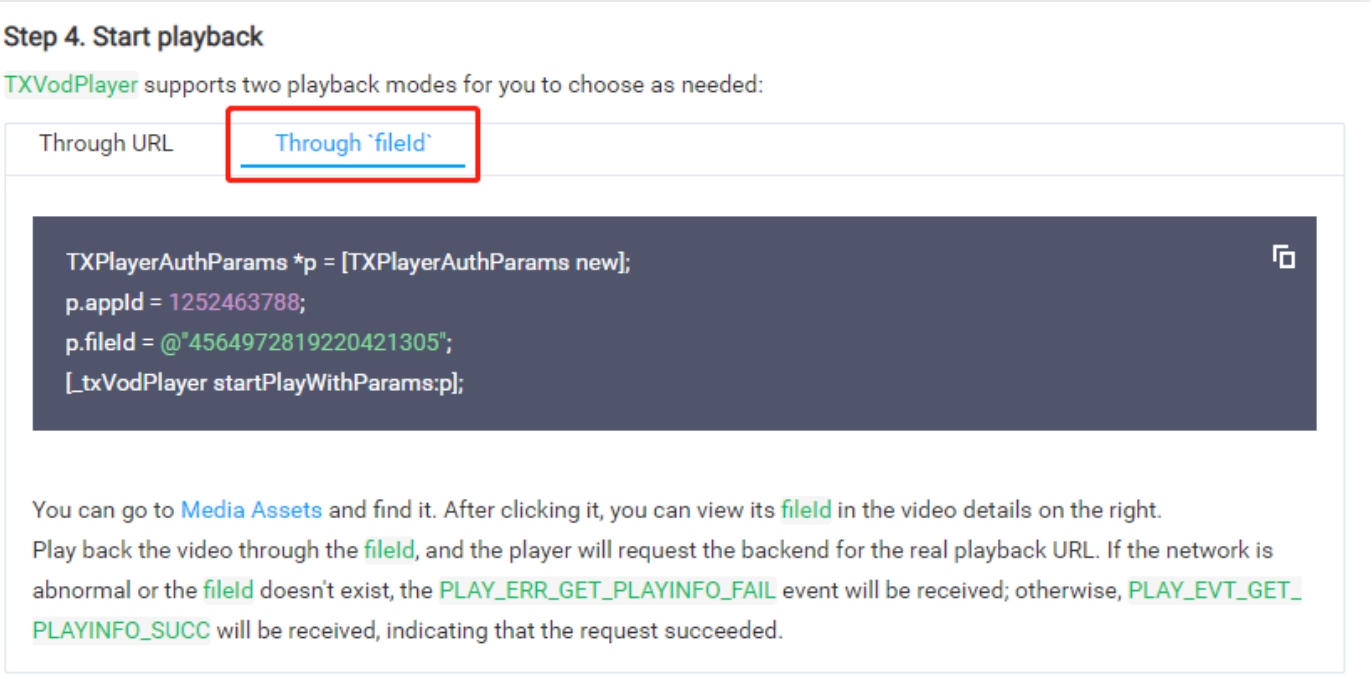
在页面初始化的代码中加入以下初始化脚本，传入必须的初始化参数（其中包含步骤3中生成的播放器签名 `psign` ），代码如下：

```
var player = TCPlayer('player-container-id', {
appID: '1500014561', // 请传入点播账号的appID （必须）
fileID: '387702304941991610', // 请传入需要播放的视频filID （必须）
psign: 'eyJhbGciOiJIUzI1NiJ9.eyJhcHBJZCI6MTUwMDAxNDU2MSwiZmlsZUlkIjoiMzg3NzAyMzA0OTQxOTkxNjEwIiwiY3VycmVudFRpbWVTdGFtcCI6MTY2MTE2MzM3MywiZXhwaXJlVGltZVN0YW1wIjoyNjQ4NTU3OTE5LCJwY2ZnIjoiU0RNQy1hZHZhbmNlRHJtUHJlc2V0In0.BYdxHHEMH0isrta4ERmksGbfu4cLiwl7f1cu04XV890',
// 其他参数请在开发文档中查看 https://intl.cloud.tencent.com/document/product/266/39105
});
```

iOS 端

请参考 接入指引（通过 FileId 方式）播放 DRM 加密视频。其中，这一过程中需要使用到步骤3中生成的播放器签名 `psign` 。

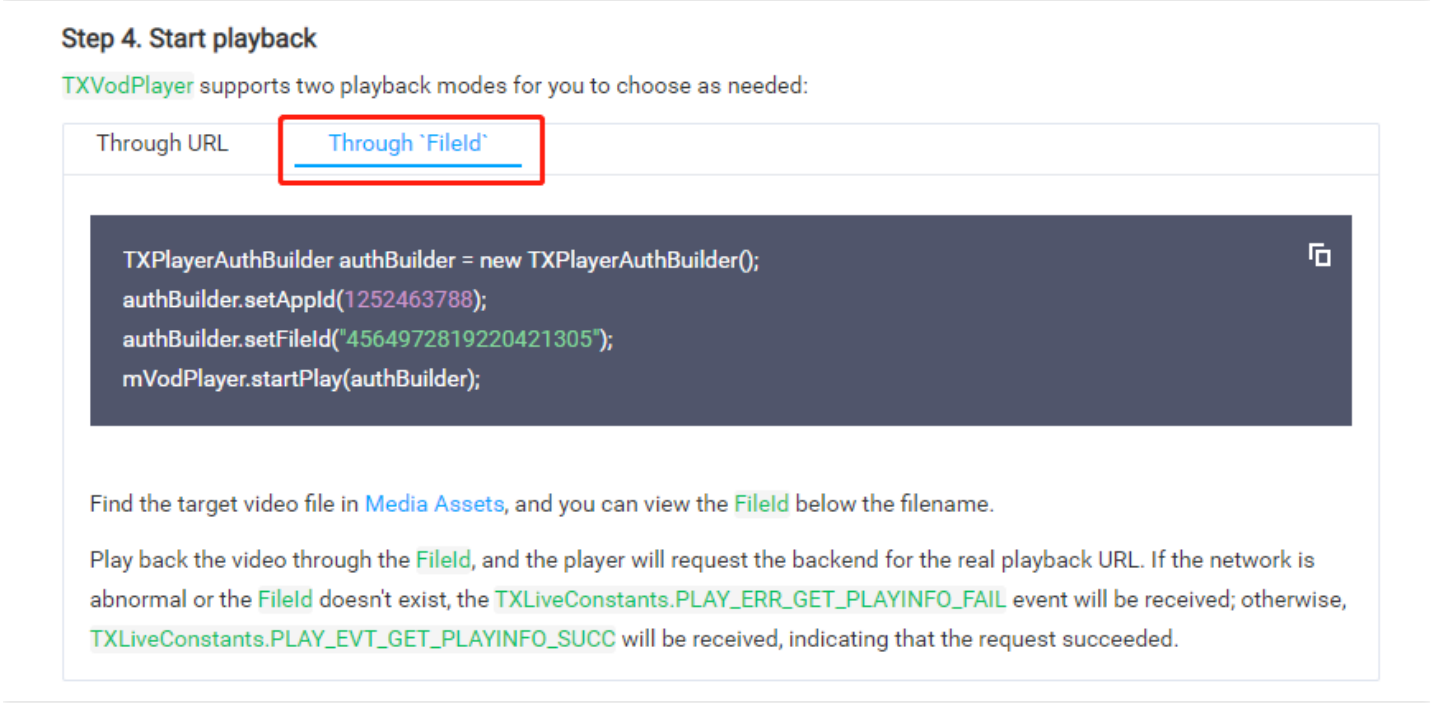
> 说明：
> 在接入前，请您提交工单联系我们获取支持 DRM 功能的 SDK 。



Android 端

请参考 接入指引（通过 FileId 方式）播放 DRM 加密视频。其中，这一过程中需要使用到步骤3中生成的播放器签名 `psign` 。

> 说明：
> 在接入前，请您提交工单联系我们获取支持 DRM 功能的 SDK 。



总结

学习本教程后，您已经掌握如何对视频进行 DRM 加密，并使用播放器播放加密后的视频。

> 说明：
> 在您对接 DRM 或者华曦达的过程中的任何问题，都可以提工单联系我们，我们全程负责帮您解决。

访问管理
访问管理综述

最近更新时间：2021-04-23 15:20:45

> ⚠ **注意：**
>
> 本文档主要介绍**云点播**访问管理功能的相关内容，其他产品访问管理相关内容请参见 支持 CAM 的产品。

云点播已接入腾讯云 访问管理 （Cloud Access Management，CAM），开发者可以根据自身需要为子账号分配合适的云点播访问权限。在开通云点播服务后，云点播访问管理功能即可直接使用。
阅读本文前，开发者需要对腾讯云访问管理和云点播子应用体系有所了解。本文涉及的概念主要有：

- 访问管理相关：用户类型、API 密钥、策略 、策略语法
- 云点播相关：子应用

应用场景

云点播访问管理有以下几种典型应用场景：

- **云产品维度权限隔离**
  某企业内有多个部门在使用腾讯云，其中 A 部门专门负责对接云点播。A 部门的人员需要有访问云点播的权限，但不能有访问其它腾讯云产品的权限。这时可以创建一个子用户，只授予该子用户云点播相关权限，然后将该子用户提供给 A 部门使用。
- **云点播子应用维度权限隔离**
  某企业内有多个业务在使用云点播，相互之间需要进行隔离。隔离包括资源隔离和权限隔离两个方面，前者由云点播子应用体系提供，后者则由云点播访问管理来实现。该企业可以为每个业务创建一个子用户，授予对应的云点播子应用权限，使得每个业务只能访问和自己相关的子应用。
- **云点播操作维度权限隔离**
  某企业的一个业务在使用云点播，该业务的产品运营人员需要访问云点播控制台，获取统计数据信息（如流量地域分布、视频播放次数等），同时不允许其进行敏感操作（如删除文件、关闭域名等），以免误操作影响业务。这时可以先创建自定义策略，该策略拥有云点播控制台登录、统计数据 API 的访问权限，然后创建一个子用户，与上述策略绑定，将该子用户提供给产品运营人员。

资源粒度和操作粒度

访问管理的核心功能可以表达为：**允许或禁止某账号对某些资源进行某些操作**。对于云点播来说，资源的粒度是子应用，操作的粒度是服务端 API。

能力限制

- 云点播访问管理的资源粒度为子应用，不支持对更细粒度的资源（如媒体文件、域名等）做授权。

支持资源级授权的 API

云点播访问管理支持 资源级授权，除了有特殊限制的 API，其余 API 均支持资源级授权。具体情况请参考下文。

不支持资源级授权的 API 列表

接口名称	接口功能	说明
DescribeSubAppIds	查询子应用列表	所有子用户都有权限调用该接口，不需要授权。请求该接口时也不需要指定具体的子应用。
ModifySubAppIdStatus	修改子应用状态	该接口可停用指定的子应用，属于高危操作，因此仅具备云点播完整权限（即 预设策略 `QcloudVODFullAccess` ）的子用户才允许访问。如果一个子用户拥有某个子应用的写权限，但没有 `QcloudVODFullAccess` 权限，也无权调用该接口。

支持资源级授权的 API 列表

除了上述不支持资源级授权的 API 列表，其它所有展示在 API 概览 中的接口都支持资源级授权。策略语法中对这些接口的资源描述形式均相同，具体为： `qcs::vod::uin/$uin:subAppId/$subAppId` 。

预设策略

最近更新时间：2022-05-31 10:48:39

> 注意：
> 本文档主要介绍**云点播**访问管理功能的相关内容，其他产品访问管理相关内容请参见 支持 CAM 的产品。

访问管理实质上是将子账号与策略进行绑定，或者说将策略授予子账号。开发者可以在控制台上直接使用预设策略来实现一些简单的授权操作，复杂的授权操作请参见 自定义策略。

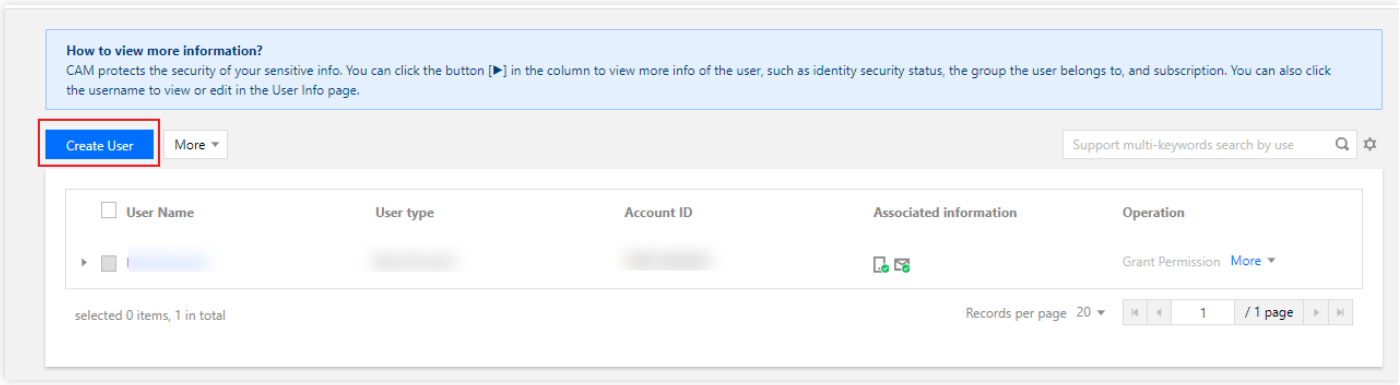
云点播目前提供了以下预设策略：

策略名称	策略描述
QcloudVODFullAccess	云点播全读写访问权限
QcloudVODReadonlyAccess	云点播只读访问权限

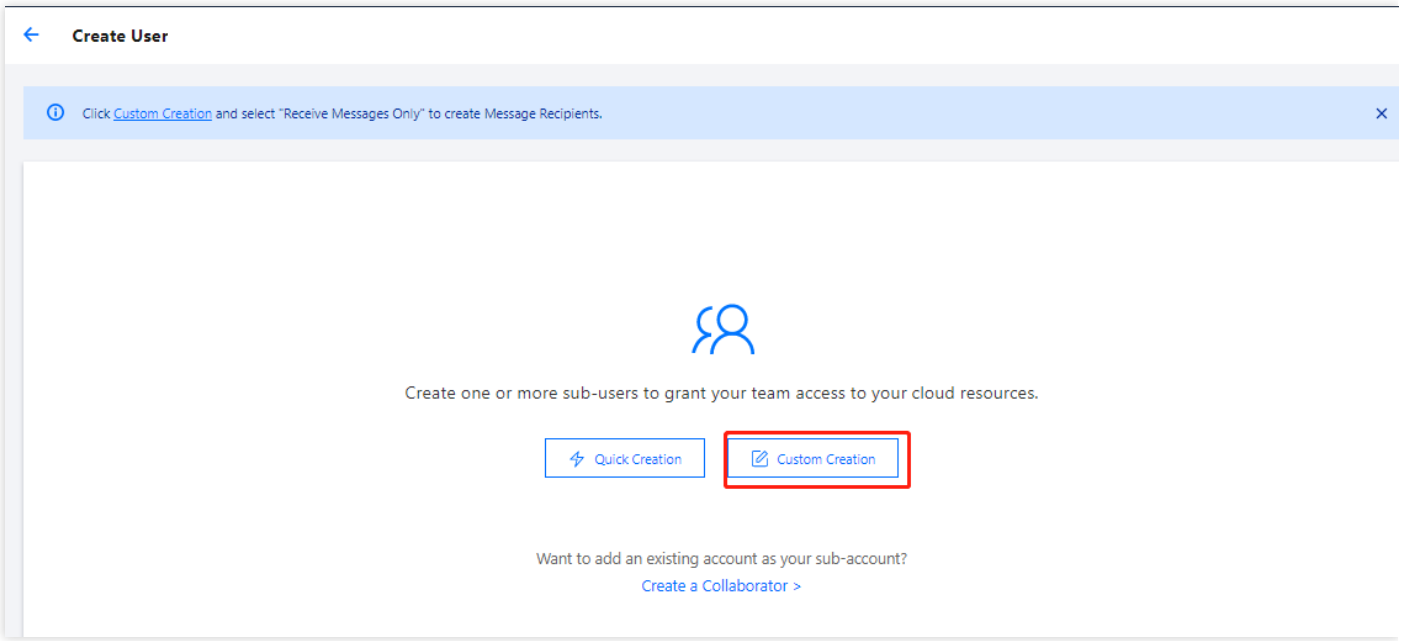
预设策略使用示例

创建拥有云点播完整权限的子用户

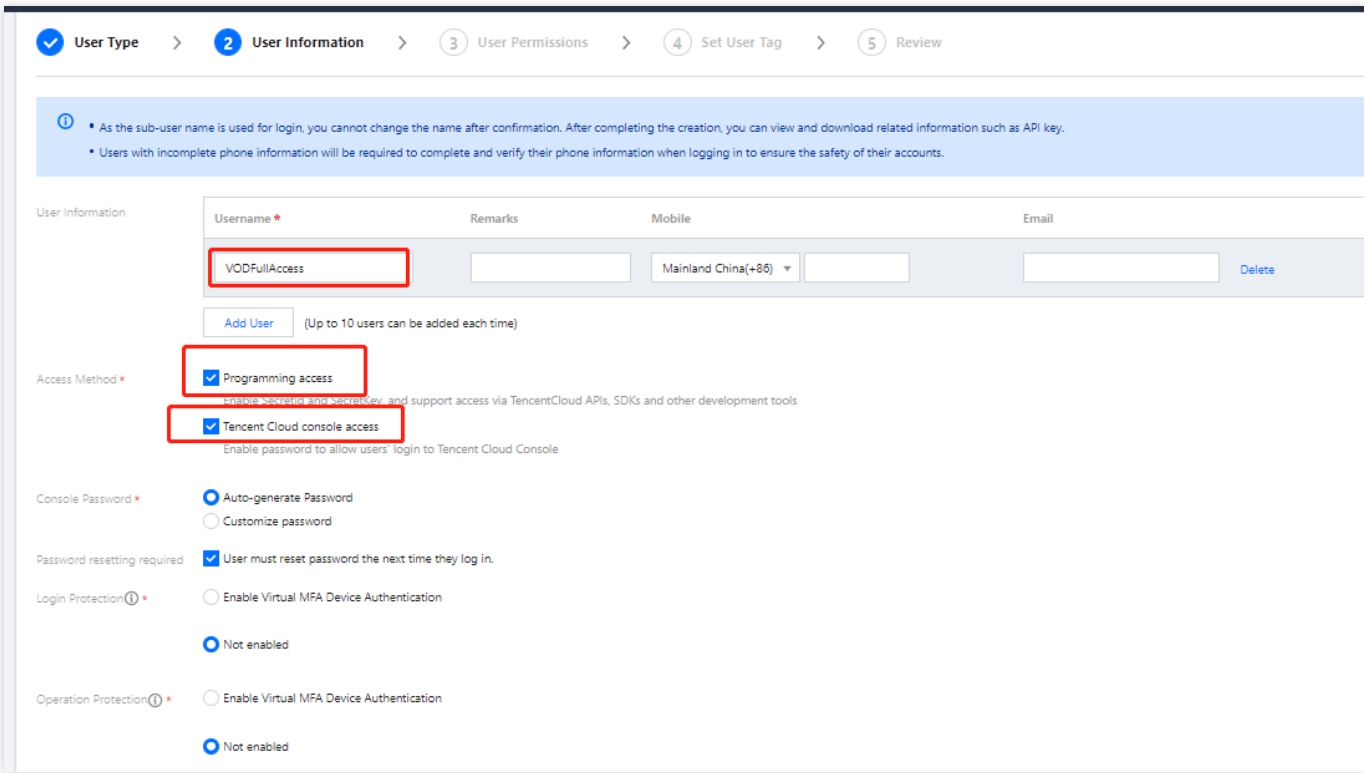
1. 以 主账号 的身份访问 CAM 控制台的【用户列表】，单击【新建用户】。



2. 在“新建用户”页面单击子用户类型下的【自定义创建】，进入“新建子用户”页面。

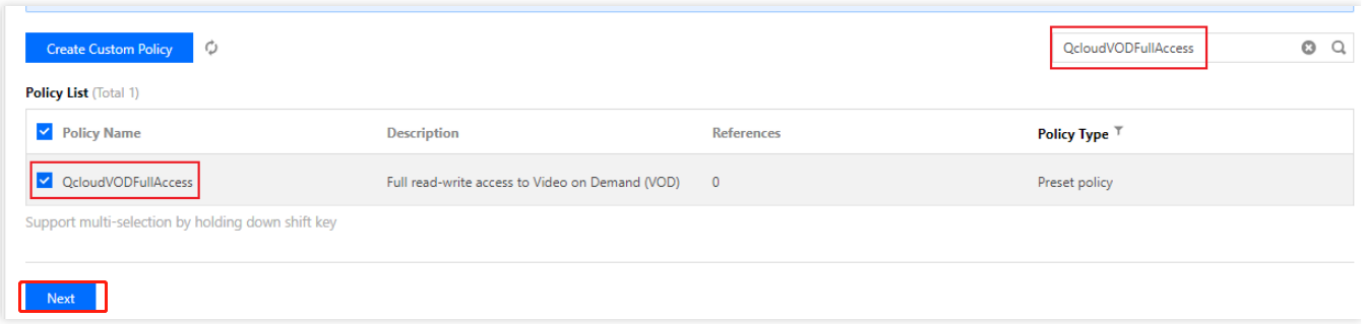


3. 单击【下一步】，填写用户信息。
   - 填写用户名，勾选【编程访问】和【腾讯云控制台访问】，其余选项按需配置。
   - 单击【下一步】，按照页面的提示完成身份验证。

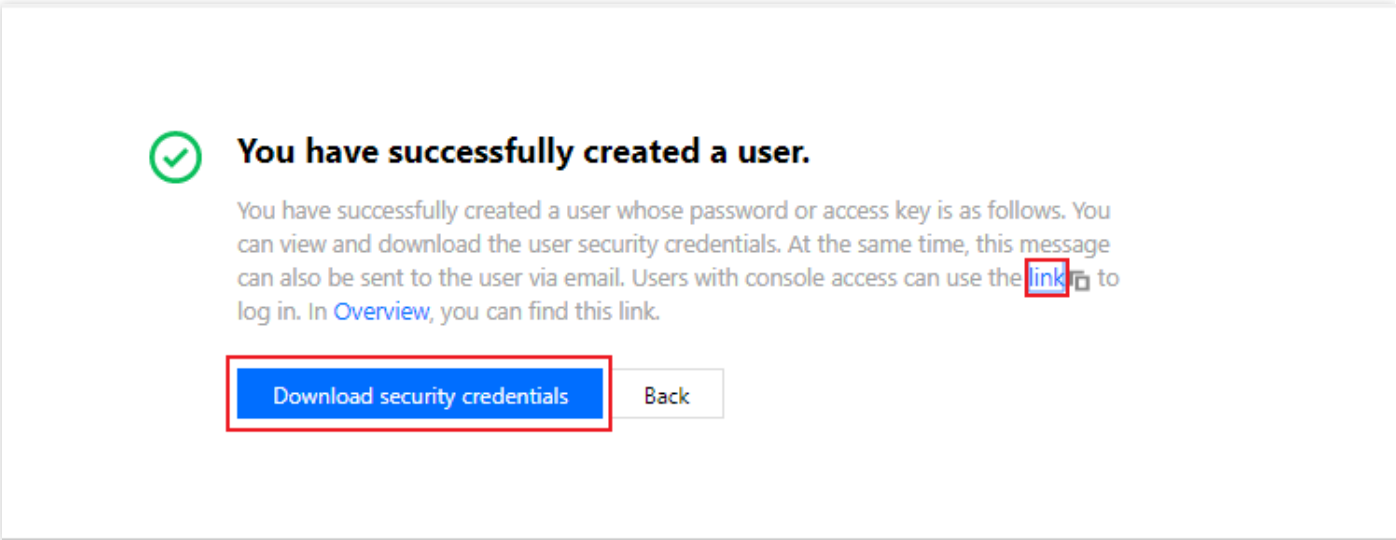


4. 设置用户权限。
   - 搜索并勾选预设策略 `QcloudVODFullAccess` 。

- 单击【下一步】。



5. 在“审阅信息和权限”分栏下单击【完成】，完成子用户的创建，在成功页面下载并保管好该子用户的登录链接和安全凭证，其中包含的信息如下表：



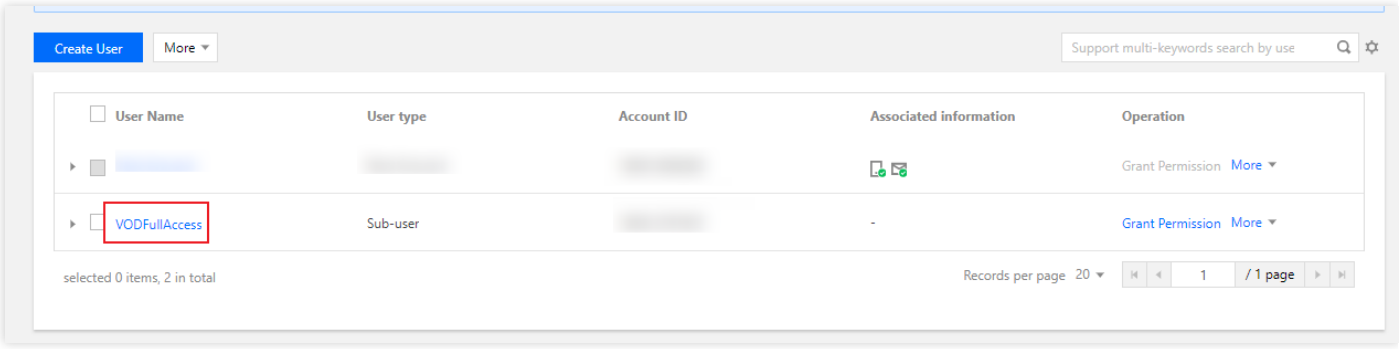
信息	来源	作用	是否必须保存
登录链接	在页面中复制	方便登录控制台，省略填写主账号的步骤	否
用户名	安全凭证 CSV 文件	登录控制台时填写	是
密码	安全凭证 CSV 文件	登录控制台时填写	是
SecretId	安全凭证 CSV 文件	调用服务端 API 时使用，详见 访问密钥	是
SecretKey	安全凭证 CSV 文件	调用服务端 API 时使用，详见 访问密钥	是

将上述登录链接和安全凭证提供给云点播使用方，后者即可使用该子用户对云点播做所有操作（包括访问云点播控制台、请求云点播服务端 API 等）。

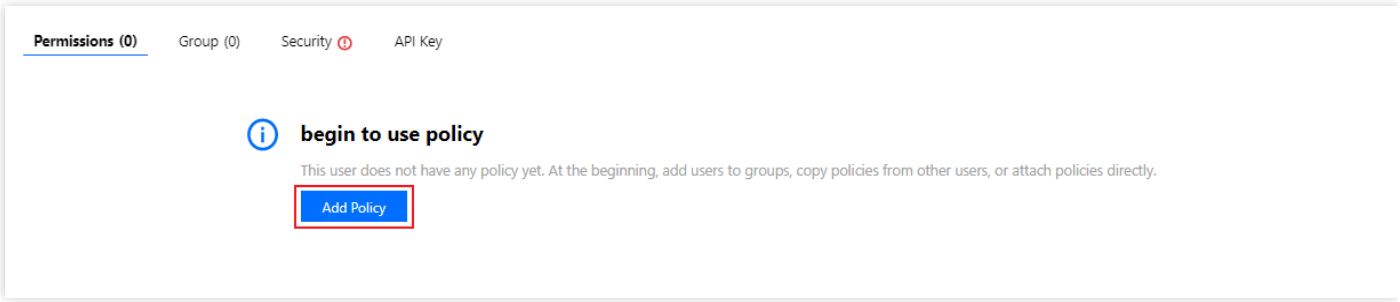
说明：
创建子用户的通用流程请参见 CAM 的 创建子用户 文档。

将云点播完整权限授予已存在的子用户

1. 以 主账号 的身份访问 CAM 控制台的【用户列表】，单击想要进行授权的子账号。



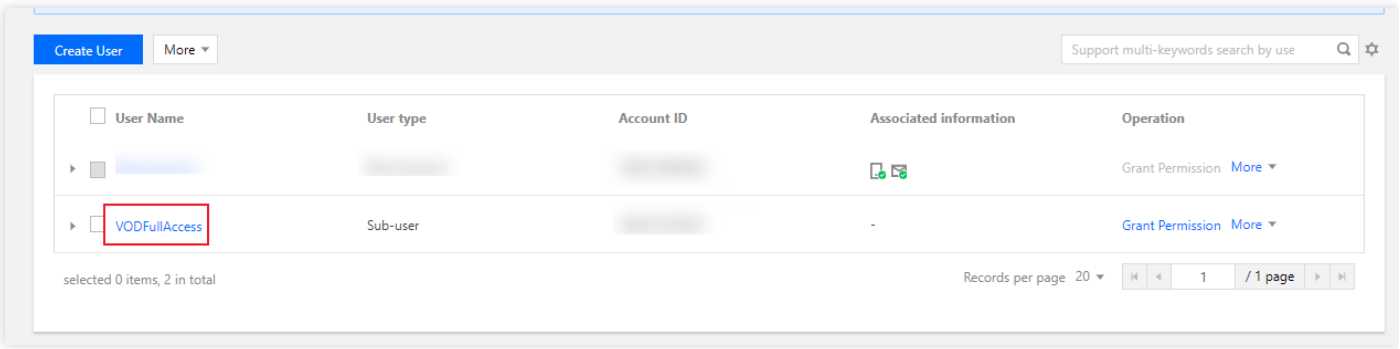
2. 单击“用户详情”页面权限栏的【添加策略】，如下图所示（实际操作中，根据子账号已有权限的不通过，页面的信息可能有所差异。如果子账号的权限非空，则单击【关联策略】）。



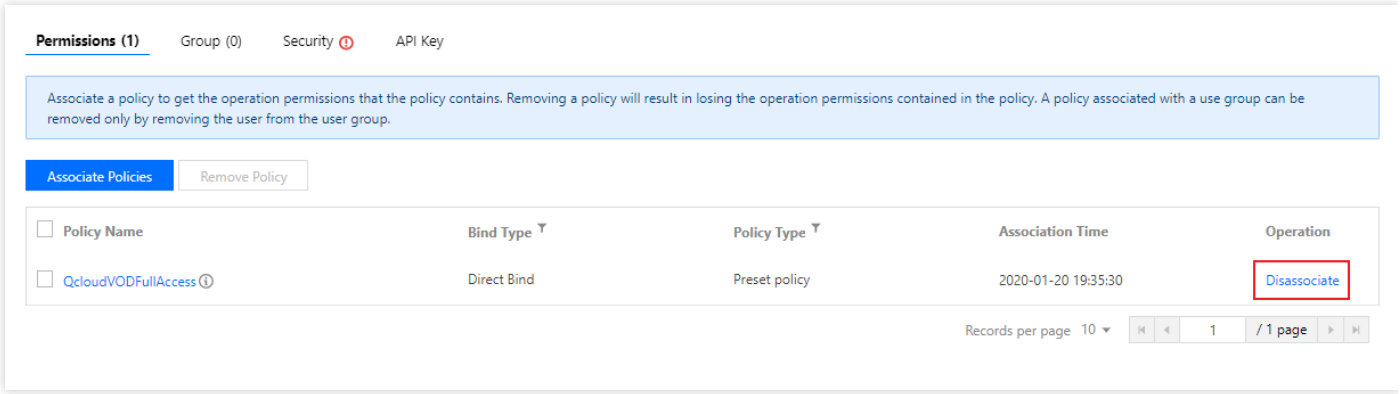
3. 选择【从策略列表中选取策略关联】，搜索并勾选预设策略 `QcloudVODFullAccess` 。后续按页面提示完成授权流程即可。

解除子用户的云点播完整权限

1. 以 主账号 的身份访问 CAM 控制台的【用户列表】，单击想要解除授权的子账号。



2. 在“用户详情”页面权限栏找到预设策略 `QcloudVODFullAccess`，单击右侧的【解除】。按页面提示完成解除授权流程即可。



自定义策略

最近更新时间：2022-05-31 11:04:02

注意：
本文档主要介绍**云点播**访问管理功能的相关内容，其他产品访问管理相关内容请参见 支持 CAM 的产品。

在访问管理中使用 预设策略 来实现授权虽然方便，但权限控制粒度粗，不能细化到子应用和 API 粒度。如果开发者要求精细的权限控制能力，则需要创建自定义策略。

自定义策略创建方法

自定义策略有多种创建方法，下方表格展示各种方法的对比，具体操作流程请参考下文。

创建入口	创建方法	效力（Effect）	资源（Resource）	操作（Action）	灵活性	难度
控制台	策略生成器	手动选择	语法描述	手动选择	中	中
控制台	策略语法	语法描述	语法描述	语法描述	高	高
服务端 API	CreatePolicy	语法描述	语法描述	语法描述	高	高

说明：

- 云点播不支持按照产品功能创建自定义策略。
- **手动选择**是指用户在控制台所展示的候选项列表中选择对象。**语法描述**是指通过策略语法来描述对象。

策略语法资源描述

如上文所述，云点播权限管理的资源粒度是子应用。子应用的策略语法描述方式遵循 CAM 标准。在下文的示例中，开发者的主账号 ID 是12345678，APPID 是1250000001（主应用 ID 等于 APPID），开发者创建了两个云点播子应用，ID 分别是1400000001和1400000002。

- **云点播所有资源的策略语法描述**

```
"resource": [
"qcs::vod::uin/12345678:subAppId/*"
]
```

- **主应用的策略语法描述**

```
"resource": [
"qcs::vod::uin/12345678:subAppId/1250000001"
]
```

- **单个子应用的策略语法描述**

```
"resource": [
"qcs::vod::uin/12345678:subAppId/1400000001"
]
```

- **主应用和单个子应用的策略语法描述**

```
"resource": [
"qcs::vod::uin/12345678:subAppId/1250000001",
"qcs::vod::uin/12345678:subAppId/1400000001"
]
```

策略语法操作描述

如上文所述，云点播权限管理的操作粒度是服务端 API。在下文的示例中，以 `DescribeMediaInfos` 、 `DescribeAllClass` 等服务端 API 为例。

- **云点播所有服务端 API 的策略语法描述**

```
"action": [
"name/vod:*"
]
```

- **单个服务端 API 操作的策略语法描述**

```
"action": [
"name/vod:DescribeMediaInfos"
]
```

- **多个服务端 API 操作的策略语法描述**

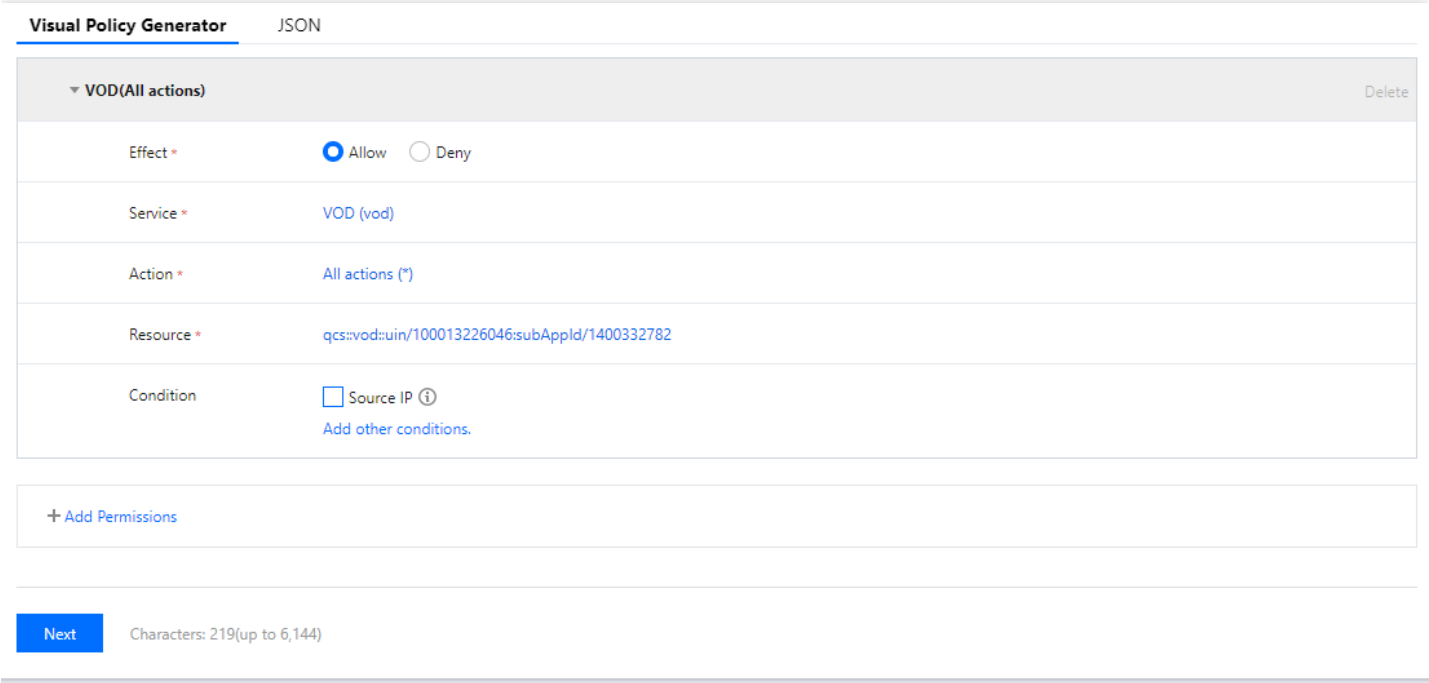
```
"action": [
"name/vod:DescribeMediaInfos",
"name/vod:DescribeAllClass"
]
```

自定义策略使用示例

使用策略生成器

在下文示例中，我们将创建一个自定义策略。该策略允许对1400000001这个云点播子应用进行任何操作，除了 `ProcessMedia` 这个服务端 API。

1. 以 主账号 的身份访问 CAM 控制台的【策略】，单击【新建自定义策略】。
2. 选择【按策略生成器创建】，进入策略创建页面。
3. 选择服务和操作。
   - 【效果(Effect)】配置项选择【允许】。
   - 【服务(Service)】配置项选择【云点播】。
   - 【操作(Action)】配置项勾选所有项。
   - 【资源(Resource)】配置项按照 资源语法描述 说明填写 `qcs::vod::uin/12345678:subAppId/1400000001` 。
   - 【条件(Condition)】配置项无需配置。



4. 单击【下一步】，按需修改策略名称（也可以不修改）。
5. 单击【完成】完成自定义策略的创建。后续将该策略授予子用户的方法同 将云点播完整权限授予已存在的子用户。

使用策略语法

在下文示例中，我们将创建一个自定义策略。该策略允许对1400000001和1400000002这两个云点播子应用进行任何操作，但不允许对1400000001进行 `ProcessMedia` 操作。

1. 以 主账号 的身份访问 CAM 控制台的【策略】，单击【新建自定义策略】。
2. 选择【按策略语法创建】，进入策略创建页面。
3. 在【选择模板类型】框下选择【空白模版】。

> 说明：
> 所谓策略模版，指新策略是现有策略（预置策略或自定义策略）的一个拷贝，然后在此基础上做调整。在实际使用中，开发者可以根据情况选择合适的策略模版，降低编写策略内容的难度和工作量。

4. 单击【下一步】，按需修改策略名称（也可以不修改）。
5. 在【编辑策略内容】编辑框中填写策略内容。本示例的策略内容为：

```
{
"version": "2.0",
"statement": [
{
```

```
"effect": "allow",
"action": [
"name/vod:*"
],
"resource": [
"qcs::vod::uin/12345678:subAppId/1400000001",
"qcs::vod::uin/12345678:subAppId/1400000002"
]
},
{
"effect": "deny",
"action": [
"name/vod:ProcessMedia"
],
"resource": [
"qcs::vod::uin/12345678:subAppId/1400000001"
]
}
]
}
```

说明：
策略内容遵循 CAM 策略语法 规范，其中资源和操作两个元素的语法分别如上文 策略语法资源描述 和 策略语法操作描述 所述。

6. 单击【创建策略】完成自定义策略的创建。后续将该策略授予子用户的方法同 示例：将云点播完整权限授予已存在的子用户。

使用服务端 API

对于大多数开发者来说，在控制台完成权限管理操作已经能满足业务需求。但如果需要将权限管理能力自动化和系统化，则可以基于服务端 API 来实现。
策略相关的服务端 API 属于 CAM，具体请参见 CAM 官网文档。此处仅列出几个主要接口：

- 创建策略
- 删除策略
- 绑定策略到用户
- 解除绑定到用户的策略

下载媒体文件

最近更新时间：2023-03-07 11:20:50

云点播支持将存储在云端的媒体文件下载到本地磁盘或其他存储上。

媒体文件

使用云点播服务时可能产生多种可下载的媒体文件，包括源文件、转码文件、视频截图、封面图片等。云点播提供了各类媒体文件的下载。媒体文件的分类如下：

- 音视频
  - 音视频源文件：上传到云点播的音视频源文件。
  - 媒体处理任务生成的音视频文件：普通转码文件、自适应码流等。
- 图片
  - 图片源文件：上传到云点播的图片源文件。
  - 媒体处理任务生成的图片文件：截图、雪碧图、动图等。

通过控制台获取下载地址

- 登录 云点播控制台，进入**媒资管理**，选择**音视频管理**或**图片管理**，点击对应的文件右侧的**管理**，即可获取源文件和媒体处理输出文件的下载地址，具体请参见 查看音视频 和 管理图片。
- 登录 云点播控制台，进入**媒资管理>音视频管理**，在 **更多批量操作** 中，可以批量导出音视频源文件和媒体处理输出文件的下载地址，具体请参见 导出音视频。

通过 API 获取下载地址

云点播也提供了 API 来获取相应的文件下载地址，请参见：

- 获取媒体详细信息
- 搜索媒体信息

自定义下载文件名

通常情况下，通过浏览器访问一个媒体文件 URL 时，浏览器会尝试直接打开这个文件而不是下载。例如，在浏览器中访问视频文件 URL 时，浏览器会直接开始播放这个视频。如果希望通过 URL 下载文件时，可以在 QueryString 中增加参数 `download_name`，即可让浏览器下载这个文件，同时自定义下载文件的名称，如：

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?download_name=[download_name]
```

注意：

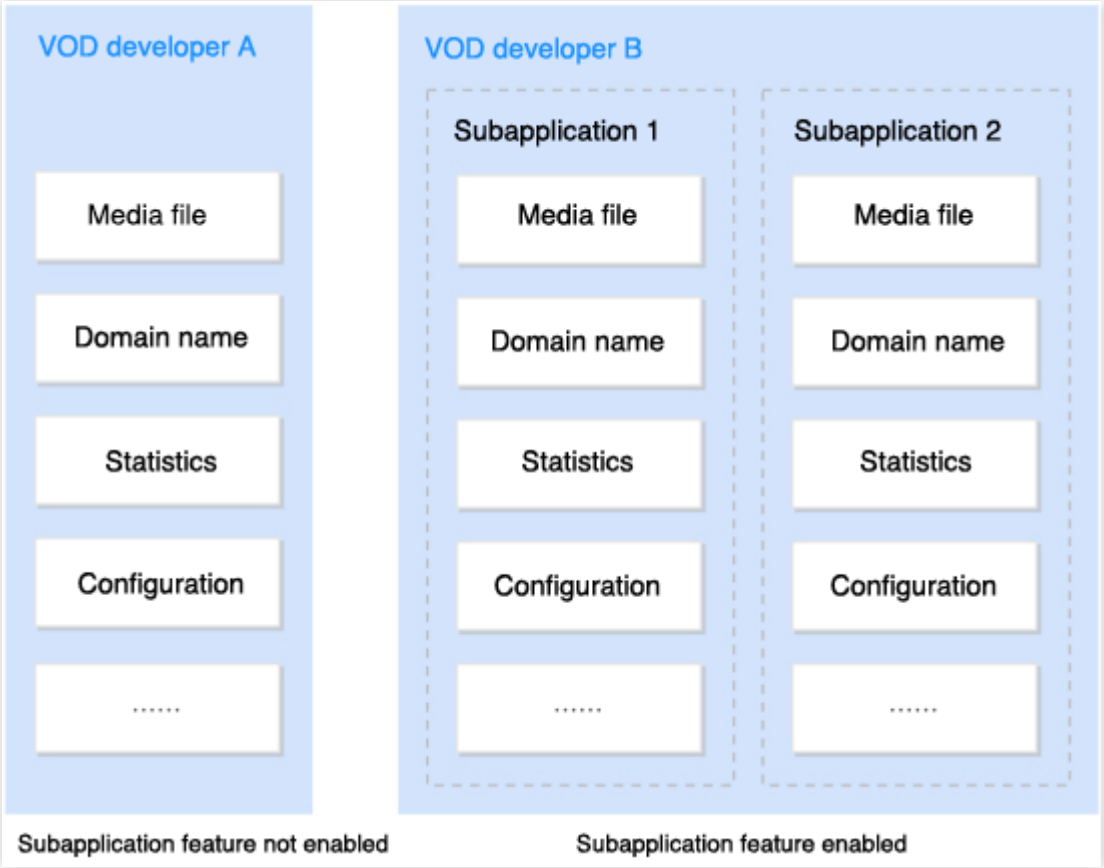
- 配置了防盗链时，下载时会受到 Referer 名单，URL 过期时间等限制，请参见 防盗链综述。
- 使用了视频加密功能后，下载的转码文件为加密文件，需要解密后才能播放，请参见 播放加密视频。
- 下载 HLS 文件时，需要分别下载索引文件和各个分片文件。可以将视频执行转码操作，转为 MP4 格式。

应用体系

最近更新时间：2023-09-07 17:45:55

概述

为了使开发者能够在云点播中实现资源隔离，云点播提供了**应用特性**。应用是云点播的一个内部概念，是一种资源划分的方式，一个应用的外在表现类似于一个独立的云点播账号。在创建应用后，点播资源的归属形式如下图所示：



说明：

本文所说的**资源**包括云点播中的媒体文件及其属性、由媒体文件衍生的其它文件、各类配置、CDN 域名、使用点播服务产生的统计信息等。

应用场景

云点播应用的典型应用场景如下：

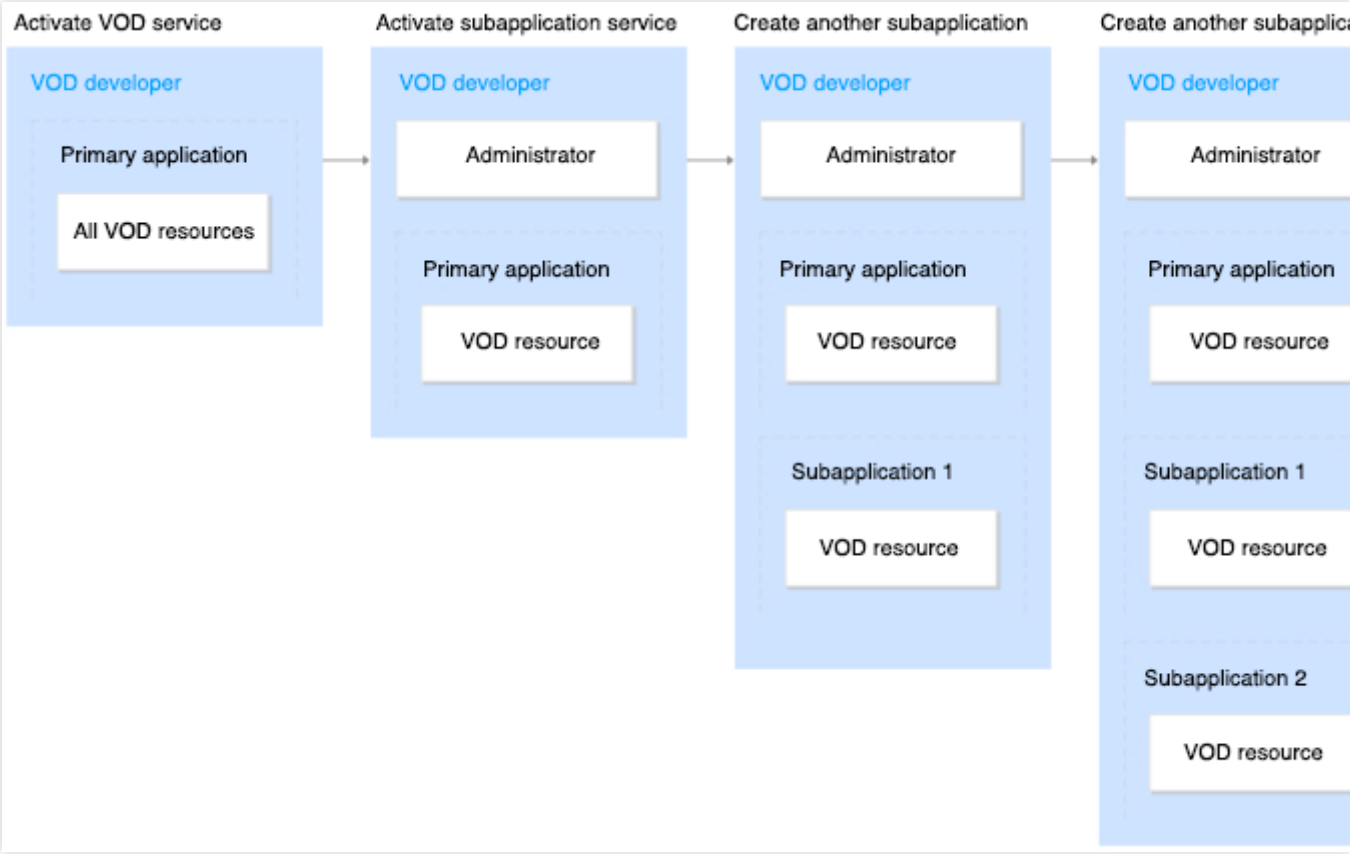
多部门/多业务隔离：某企业基于腾讯云开发自有产品，其中 A 部门需要使用云点播来开发一款短视频 App，B 部门则使用云点播来开发一个影视网站。这两个点播业务需要相互隔离，但出于财务管理的考虑，该企业无法为 A 和 B 部门分别开通一个独立的腾讯云账号。这时就可以使用云点播的应用特性，为 A 和 B 部门各分配一个应用。

权限控制：在上面的多部门/多业务隔离场景中，开发者可能会有更进一步的权限控制要求，例如要求每个部门仅能访问和自己业务关联的应用，而无权限访问其它应用。这时，账号管理者可以分别为 A 和 B 部门各分配一个子用户，并授予相应的云点播应用访问权限。操作详情请参见 访问管理。

区分正式环境和测试环境：开发者想对某些点播特性进行测试，又担心会影响线上运营（例如修改 事件通知 方式，或者开启 防盗链 等）。开发者可以为正式环境和测试环境各开通一个应用，新特性先在测试环境进行验证，确认无误后再变更线上环境。

身份定义和标识

应用体系中有两类身份：管理员和应用，我们结合下图来说明它们的定义。



1. 开发者开通云点播服务后，直接生成1个默认应用，此时所有的点播资源都归属于默认应用。默认应用的标识符为开发者的腾讯云 APPID，可在控制台的 账号信息中查看。

2. 开发者开通云点播应用功能后，会另外生成一个**管理员**身份。管理员并不拥有任何点播资源，所有资源仍然属于默认应用。

3. 开发者使用管理员身份创建一个**应用**，新建的应用拥有独立的点播资源，它和默认应用的地位平等且相互隔离，可以将默认应用理解为一个特殊的应用。创建应用时，云点播会为应用分配一个全平台唯一的标识符，称为应用

ID，查看方式请参见 控制台使用说明 - 应用管理。

4. 开发者使用管理员身份再次创建一个**应用**，这个新建的应用也拥有独立的点播资源，它与默认应用、其它应用的地位平等且相互隔离，依此类推。

说明：

如无特殊说明，下文不再区分默认应用和应用，统一用**应用**来表述。

能力

云点播应用体系提供以下能力：

创建及设置应用：开发者开通云点播应用功能后，能够以管理员身份在控制台上创建应用，并为每个应用设置名称和描述。

停用应用：除默认应用外，其它应用可以停用。停用操作不会清理应用下的点播资源，只会禁用其域名，其余功能（上传、转码等）不受影响。

资源隔离：应用之间的点播资源相互隔离。

通过控制台或者服务端 API 操作任一个应用的点播资源。

为每一个应用生成单独的数据统计信息，包括存储、带宽/流量、转码时长、视频智能识别时长、播放数据等。

为所有应用生成汇总的数据统计信息。

限制

云点播应用体系具有以下限制：

不支持修改默认应用的名称和描述。

不支持删除应用。

每个云点播账号最多可以创建50个应用。

不支持为应用设置单独的计费逻辑（如设置计费方式、单独生成账单、购买专用资源包等）。一个云点播账号下的所有应用都归属于同一个云点播账号，所有应用的点播用量（包括但不限于存储、流量、转码时长、视频智能识别时长等云点播计费项）都会进行合并计算，统一收费。

控制台使用说明

开通应用

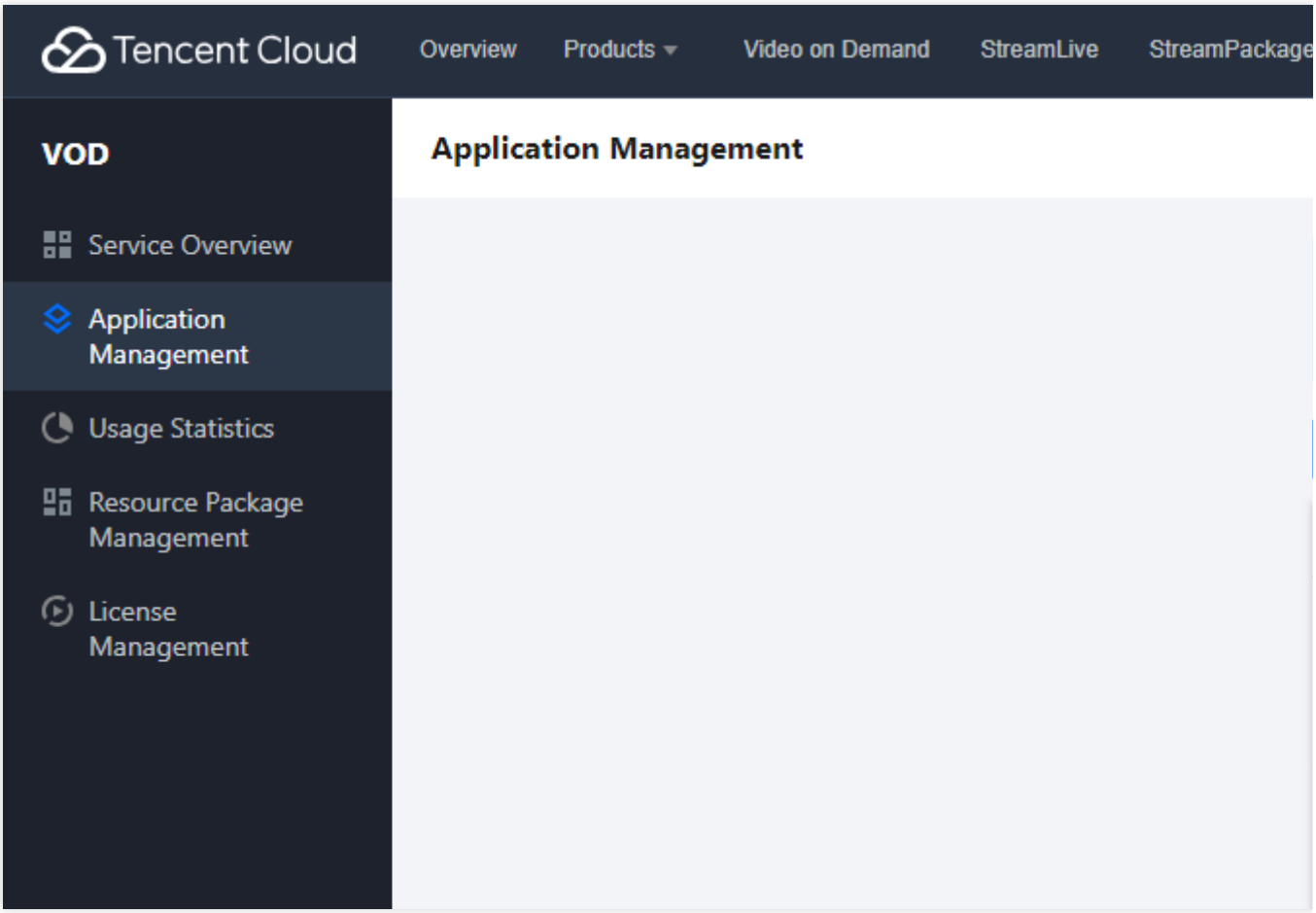
1. 登录 云点播控制台。
2. 单击左侧导航栏的**开通应用**，进入开通页面。
3. 单击该页面的**立即使用**，即可开通云点播应用功能。

说明：

如果已开通应用功能，则左侧导航栏的**开通应用**将不可见。

选择身份

开通应用功能后，进入 云点播控制台 **应用管理**列表，开发者可以在这里选择身份。如果开发者刚开通应用功能，则列表内只有一个选项：“默认应用”；如果开发者新创建应用后，列表会新增对应的身份选项。



管理员

在管理员身份下，左侧导航栏包括**服务概览、应用管理、用量统计、资源包管理** 和 **Licnese 管理**。
服务概览：该页面展示开发者的云点播计费方式、所有应用汇总后的关键业务数据以及各应用的关键业务数据。
应用管理：该页面可以查看、创建、编辑和停用应用。各个应用的标识符（应用 ID）也在此页面展示。
用量统计：该页面可以查看账号下使用各个产品功能所消耗的用量。
资源包管理：该页面可以查看各类型资源包的使用情况。
License 管理：该页面可以查看已绑定视立方视频 License 。

应用

在应用身份下，云点播控制台的使用方法和未开通应用功能时几乎完全一致，用户可以查看和操作归属于该应用的点播资源。两者的主要区别在于应用没有自己的计费配置。

服务端 API 使用说明

在开通云点播应用功能后，开发者在使用 云点播服务端 API 时需要指定访问的是哪个应用的资源。

在服务端 API 中指定应用

云点播服务端 API 已经升级为 云 API 3.0 版本，用户可以在各个 API 的 `SubAppId` 参数中指定所要访问的应用。如果要访问的是默认应用，既可以填写默认应用的标识符，也可以不填。

在服务端 API 2017 中指定应用

服务端 API 2017 也支持应用，使用时需要在请求中增加一个 `SubAppId` 参数（注意大小写），该参数与服务端 API 2017 的 公共请求参数 平级，其值为应用 ID。如果要访问的是默认应用，既可以填写默认应用的标识符，也可以不填。

说明：

服务端 API 2017 的文档并未暴露 `SubAppId` 参数，但不影响使用该参数。

`SubAppId` 参数也要参与服务端 API 的签名计算，计算规则不变。

文件上传说明

在开通云点播应用功能后，开发者在上传媒体文件到云点播时需要指定上传到哪个应用下。

直播录制

直播录制 支持录制到指定的应用，指定方法为在直播推流参数中增加 `vod_sub_app_id=xxx`（`xxx` 指应用 ID）。如果要录制到主应用，不带该参数即可。

服务端上传

服务端上传 支持上传到指定的应用，具体的参数填写方法见下方链接。如果要上传到默认应用，既可以填写默认应用的标识符，也可以不填。

SDK 方式

Java SDK
PHP SDK
Python SDK
Node.js SDK
Golang SDK

服务端 API 方式

使用 API 方式进行上传时涉及到 ApplyUpload 和 CommitUpload 两个接口，具体用法请参见 在服务端 API 中指定应用。
我们强烈建议使用 SDK 来进行上传。

客户端上传

客户端上传 支持上传到指定的应用，指定方法为在 客户端上传签名 中增加一个参数：`vodSubAppId=xxx`（`xxx` 指应用 ID）。如果要上传到默认应用，既可以填写默认应用的标识符，也可以不填。

说明：

`vodSubAppId` 参数也要参与客户端上传签名计算，计算规则不变。

URL 拉取上传

URL 拉取视频上传支持上传到指定的应用。
控制台方式：具体用法请参见 控制台使用说明。
服务端 API 方式：使用 PullUpload 接口，具体用法请参见 在服务端 API 中指定应用。

权限管理

云点播已接入访问管理 CAM，支持应用维度的授权，详情请参见 访问管理。

FAQ

开通应用功能后，是否影响线上原有的业务逻辑？

不影响。应用体系在设计时考虑了兼容性，所有的服务端 API 接口在不指定应用 ID 时，默认为操作默认应用。

开通应用功能是否收费？

每个用户可以免费创建20个应用（包含默认应用），超过20个部分应用个数需另外收费，同时每个应用所产生的消耗都将计入该云点播账号，并按云点播 计费逻辑 进行计费。

我们使用应用功能来实现业务隔离，那么各业务该如何进行内部结算/成本分摊？

如上文 限制 的说明，云点播只为整个账号出一份汇总的账单。如果开发者内部多个业务需要进行成本分摊，那么可以基于云点播提供的应用维度的统计数据，自行定义并计算内部成本分摊。

开发者被停服会对应用有什么影响？

开发者的云点播服务发生 欠费停服 时，该账号下所有的应用都会被停服。

我可以把归属于某个应用的视频迁移到另一个应用吗？

应用之间的资源是隔离的，无法迁移。

错误码

最近更新时间：2021-10-29 15:32:00

错误码列表

视频处理类错误码

错误码	含义
InvalidInput	输入参数不合法，请检查输入参数。
InvalidInput.InvalidTimeOffset	输入参数不合法：指定的时间点不合法。
InvalidInput.DefinitionNotExist	输入参数不合法：指定的模板 ID 不存在。
InvalidInput.ConfigurationUnsupported	输入参数不合法：配置无效，包括但不限于： • 用户未注册。 • 输入参数取值非法（格式、取值范围等）。 • 参数模板配置有问题。 • 未指定视频处理任务。
InvalidInput.TaskDuplicated	输入参数不合法：任务重复。
InvalidInput.PermissionDenied	输入参数不合法：没有权限，请申请产品功能权限后使用。
InvalidInput.ResultFileSizeTooLarge	输入参数不合法：输入多文件拼接后结果文件过大。
SourceFileError	源文件错误：如视频数据损坏，请确认源文件是否正常。
SourceFileError.NoVideoMedia	源文件错误：没有视频轨画面。
SourceFileError.NoVideoResolution	源文件错误：无法获取源文件的分辨率。
SourceFileError.ContentMalformed	源文件错误：输入内容存在问题，如文件不存在、文件损坏、媒体文件无法解码。
SourceFileError.ContentUnsupported	源文件错误：输入的格式有问题，如不受支持的文件格式、文件大小、文件时长。
SourceFileError.DownloadNotAccessible	源文件错误：尝试下载输入文件时，这些文件无法访问，请检查源的可用性。
InternalError	内部服务错误，建议重试。