数据万象

实践教程

腾讯云

文档目录

实践教程
临时密钥生成及使用指引

最近更新时间：2024-07-22 10:32:11

> **注意：**
> - 使用临时密钥授权访问时，请务必根据业务需要，按照最小权限原则进行授权。如果您直接授予所有资源 `(resource:*)`，或者所有操作 `(action:*)` 权限，则存在由于权限范围过大导致的数据安全风险。
> - 您在申请临时密钥时，如果指定了权限范围，那么申请到的临时密钥也只能在权限范围内进行操作。例如您在申请临时密钥时，指定了可以往存储桶 examplebucket-1-1250000000 上传文件的权限范围，那么申请到的密钥**不能**将文件上传到 examplebucket-2-1250000000，也**不能**从 examplebucket-1-1250000000 中下载文件。

临时密钥

临时密钥（临时访问凭证）是通过 CAM 云 API 提供的接口，获取到权限受限的密钥。
CI API 可以使用临时密钥计算签名，用于发起 CI API 请求。
CI API 请求使用临时密钥计算签名时，需要用到获取临时密钥接口返回信息中的三个字段，如下：

- TmpSecretId
- TmpSecretKey
- Token

使用临时密钥的优势

Web、iOS、Android 使用 CI 时，通过固定密钥计算签名方式不能有效地控制权限，同时把永久密钥放到客户端代码中有极大的泄露风险。如若通过临时密钥方式，则可以方便、有效地解决权限控制问题。
例如，在申请临时密钥过程中，可以通过设置权限策略 policy 字段，限制操作和资源，将权限限制在指定的范围内。
有关 CI 的 API 授权策略，请参见：

- 常见场景的临时密钥权限策略示例
- 授权数据处理场景（数据万象）权限策略示例

获取临时密钥

获取临时密钥，可以通过提供的 COS STS SDK 方式获取，也可以直接请求 STS 云 API 的方式获取。

> **注意：**
> 举例使用的是 Java SDK，需要在 GitHub 上获取 SDK 代码（版本号）。若提示找不到对应 SDK 版本号，请确认是否在 GitHub 上获取到对应版本的 SDK。

COS STS SDK

COS 针对 STS 提供了 SDK 和样例，目前已有 Java、Nodejs、PHP、Python、Go 等多种语言的样例。具体内容请参见 COS STS SDK。各个 SDK 的使用说明请参见 Github 上的 README 和样例。各语言 GitHub 地址如下表格所示：

语言类型	代码仓库说明	代码示例地址
Java	使用文档	示例地址
.NET	使用文档	示例地址
Go	使用文档	示例地址
NodeJS	使用文档	示例地址
PHP	使用文档	示例地址
Python	使用文档	示例地址

> ⚠ **注意：**
> STS SDK 为了屏蔽 STS 接口本身版本间的差异性，返回参数结构不一定与 STS 接口完全一致，详情请参见 Java SDK 文档 。

假设您使用的是 Java SDK，请先下载 Java SDK，然后运行如下获取临时密钥示例：

代码示例

```
// 根据 github 提供的 maven 集成方法导入 java sts sdk，使用 3.1.1 及更高版本
// <dependency>
//     <groupId>com.qcloud</groupId>
//     <artifactId>cos-sts_api</artifactId>
//     <version>3.1.1</version>
// </dependency>
public static void main(String[] args) {
    TreeMap<String, Object> config = new TreeMap<String, Object>();
    try {
        //这里的 SecretId 和 SecretKey 代表了用于申请临时密钥的永久身份（主账号、子账号等），子账号需要具有操作存储桶
的权限。
        String secretId = System.getenv("secretId");//用户的 SecretId，建议使用子账号密钥，授权遵循最小权限指
引，降低使用风险。子账号密钥获取可参见 https://cloud.tencent.com/document/product/598/37140
        String secretKey = System.getenv("secretKey");//用户的 SecretKey，建议使用子账号密钥，授权遵循最小权限
指引，降低使用风险。子账号密钥获取可参见 https://cloud.tencent.com/document/product/598/37140
        // 替换为您的云 api 密钥 SecretId
        config.put("secretId", secretId);
        // 替换为您的云 api 密钥 SecretKey
        config.put("secretKey", secretKey);

        // 初始化 policy
        Policy policy = new Policy();

        // 设置域名:
        // 如果您使用了腾讯云 cvm，可以设置内部域名
        //config.put("host", "sts.internal.tencentcloudapi.com");

        // 临时密钥有效时长，单位是秒，默认 1800 秒，目前主账号最长 2 小时（即 7200 秒），子账号最长 36 小时（即
129600）秒
        config.put("durationSeconds", 1800);
        // 换成您的 bucket
        config.put("bucket", "examplebucket-1250000000");
        // 换成 bucket 所在地区
        config.put("region", "ap-chongqing");

        // 开始构建一条 statement
        Statement statement = new Statement();
        // 声明设置的结果是允许操作
        statement.setEffect("allow");
        /**
         * 密钥的权限列表。必须在这里指定本次临时密钥所需要的权限。
         * 权限列表请参见 https://cloud.tencent.com/document/product/436/31923
         * 规则为 {project}:{interfaceName}
         * project ： 产品缩写  cos相关授权为值为cos,数据万象(数据处理)相关授权值为ci
         * 授权所有接口用*表示，例如 cos:*,ci:*
         * 添加一批操作权限 ：
         */
        statement.addActions(new String[]{
                // 创建媒体处理任务
                "ci:CreateMediaJobs",
```

```
                // 文件压缩
                "ci:CreateFileProcessJobs",
                // cos资源上传
                "cos:PutObject"
        });

        /**
         * 这里改成允许的路径前缀，可以根据自己网站的用户登录态判断允许上传的具体路径
         * 资源表达式规则分对象存储(cos)和数据万象(ci)两种
         * 数据处理、审核相关接口需要授予ci资源权限
         *  cos : qcs::cos:{region}:uid/{appid}:{bucket}/{path}
         *  ci  : qcs::ci:{region}:uid/{appid}:bucket/{bucket}/{path}
         * 列举几种典型的{path}授权场景：
         * 1、允许访问所有对象："*"
         * 2、允许访问指定的对象："a/a1.txt", "b/b1.txt"
         * 3、允许访问指定前缀的对象："a*", "a/*", "b/*"
         *  如果填写了“*”，将允许用户访问所有资源；除非业务需要，否则请按照最小权限原则授予用户相应的访问权限范围。
         *
         * 示例：授权examplebucket-1250000000 bucket目录下的所有资源给cos和ci 授权两条Resource
         */
        statement.addResources(new String[]{
                "qcs::cos:ap-chongqing:uid/1250000000:examplebucket-1250000000/*",
                "qcs::ci:ap-chongqing:uid/1250000000:bucket/examplebucket-1250000000/*"});

        // 把一条 statement 添加到 policy
        // 可以添加多条
        policy.addStatement(statement);
        // 将 Policy 示例转化成 String，可以使用任何 json 转化方式，这里是本 SDK 自带的推荐方式
        config.put("policy", Jackson.toJsonPrettyString(policy));

        Response response = CosStsClient.getCredential(config);
        System.out.println(response.credentials.tmpSecretId);
        System.out.println(response.credentials.tmpSecretKey);
        System.out.println(response.credentials.sessionToken);
    } catch (Exception e) {
        e.printStackTrace();
        throw new IllegalArgumentException("no valid secret !");
    }
}
```

常见问题和解答

JSONObject 包冲突导致 NoSuchMethodError?

使用3.1.1及以后的版本。

使用临时密钥访问 CI

CI API 使用临时密钥访问 CI 服务时，通过 x-cos-security-token 字段传递临时 sessionToken，通过临时 SecretId 和 SecretKey 计算签名。以 COS Java SDK 为例，使用临时密钥访问万象文件处理接口示例如下：

> **说明：**
> 运行如下示例前，请前往 Github 项目 获取 Java SDK 安装包。

```
// 根据 github 提供的 maven 集成方式导入 cos xml java sdk
public static void main(String[] args) {
    String tmpSecretId = "COS_SECRETID";    // 替换为 STS 接口返回给您的临时 SecretId
    String tmpSecretKey = "COS_SECRETKEY";  // 替换为 STS 接口返回给您的临时 SecretKey
    String sessionToken = "Token";  // 替换为 STS 接口返回给您的临时 Token
```

```
    BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKey,
sessionToken);
    // 2 设置 bucket 的地域
    // clientConfig 中包含了设置 region, https(默认 http), 超时, 代理等 set 方法, 使用可参见源码或者常见问题 Java
SDK 部分
    Region region = new Region("ap-chongqing"); //COS_REGION 参数：配置成存储桶 bucket 的实际地域，例如 ap-
beijing，更多 COS 地域的简称请参见 https://cloud.tencent.com/document/product/436/6224
    ClientConfig clientConfig = new ClientConfig(region);
    // 3 生成 cos 客户端
    COSClient cosClient = new COSClient(cred, clientConfig);
    //1.创建任务请求对象
    FileProcessRequest request = new FileProcessRequest();
    //2.添加请求参数 参数详情请见api接口文档
    request.setBucketName("demo-1234567890");
    request.setTag(FileProcessJobType.FileCompress);
    FileCompressConfig fileCompressConfig = request.getOperation().getFileCompressConfig();
    fileCompressConfig.setFormat("zip");
    fileCompressConfig.setFlatten("0");
    fileCompressConfig.setIgnoreError("true");
    List<KeyConfig> keyConfigList = fileCompressConfig.getKeyConfigList();
    KeyConfig keyConfig = new KeyConfig();
    keyConfig.setKey("1.jpg");
    keyConfig.setRename("rename-1.jpg");
    keyConfigList.add(keyConfig);
    MediaOutputObject output = request.getOperation().getOutput();
    output.setBucket("demo-1234567890");
    output.setRegion("ap-chongqing");
    output.setObject("output/demo.zip");
    //3.调用接口,获取任务响应对象
    try {
        FileProcessJobResponse response = cosClient.createFileProcessJob(request);
        System.out.println(response);
    } catch (CosServiceException e) {
        //失败，抛出 CosServiceException
        e.printStackTrace();
    } catch (CosClientException e) {
        //失败，抛出 CosClientException
        e.printStackTrace();
    }
    // 关闭客户端
    cosClient.shutdown();
}
```

API 授权策略使用指引

最近更新时间：2024-10-28 16:06:21

> **注意**
> 在给子用户或协作者授予 API 操作权限时，请务必根据业务需要，按照最小权限原则进行授权。如果您直接授予子用户或者协作者所有资源 `(resource:*)` ，或所有操作 `(action:*)` 权限，则存在由于权限范围过大导致数据安全风险。

概述

数据万象（Cloud Infinite，简称CI）使用临时密钥服务时，不同的 API 操作需要不同的操作权限，而且可以同时指定一个操作或一序列操作。
CI 的API 授权策略（policy）是一种 JSON 字符串。例如，授予 APPID 为1250000000，地域为 ap-beijing，存储桶为 examplebucket-1250000000 ，路径前缀为 doc 的媒体处理、文件压缩、文档预览任务创建操作的权限如下所示：

```
{
    "version": "2.0",
    "statement": [{
            "action": [
                //创建文件处理任务
                "ci:CreateFileProcessJobs",
                //创建文档预览任务
                "ci:CreateDocProcessJobs",
                //创建媒体任务
                "ci:CreateMediaJobs",
            ],
            "effect": "allow",
            "resource": [
                "qcs::ci:ap-beijing:uid/1250000000:bucket/examplebucket-1250000000/doc/*"
            ]
        }
    ]
}
```

授权策略（policy）元素说明

名称	描述
version	策略语法版本，默认为2.0。
effect	有 allow（允许）和 deny（显式拒绝）两种情况。
resource	授权操作的具体数据，可以是任意资源、指定路径前缀的资源、指定绝对路径的资源或它们的组合。 规则为: qcs::ci:{region}:uid/{appid}:bucket/{bucket}/{path} 列举几种典型的授权场景： 1、允许访问所有对象："*" 2、允许访问指定的对象："a/a1.txt", "b/b1.txt" 3、允许访问指定前缀的对象："a*", "a/*", "b/*" **注意：**若路径为中文，则保持中文输入即可。例如 `examplebucket-1250000000/文件夹/文件名.txt` 。
action	此处是指 CI 的 API，根据需求指定一个或者一序列操作的组合或所有操作(`*`)，例如 action 为 `ci:CreateMediaJobs` ，**请注意区分英文大小写。**
condition	约束条件，可以不填，具体说明请参见 condition 说明。

有关 CI 的 API 授权策略，请参见 数据万象全局访问管理 。

Service API

媒体处理任务

以媒体转码接口为例，API 接口为 `CreateMediaJobs`，若授予其操作权限，则策略的 action 为 ci: `CreateMediaJobs`。
媒体任务中涉及到以下几种权限的接口：

- `cos:GetObject` 获取cos资源权限，
- `ci:CreateMediaJobs` 创建任务，
- `ci:CreateMediaTemplate` 创建模板，
- `ci:UpdateMediaTemplate` 更新模板，
- `ci:DescribeMediaJob` 任务查询，
- `ci:DescribeMediaJobs` 任务列表查询，
- `ci:CancelMediaJob` 任务取消。

示例

授予查询存储桶列表操作权限的策略详细内容如下：

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "cos:GetObject",
        "ci:CreateMediaJobs",
        "ci:CreateMediaTemplate",
        "ci:UpdateMediaTemplate",
        "ci:DescribeMediaJob",
        "ci:DescribeMediaJobs",
        "ci:CancelMediaJob"
      ],
      "effect": "allow",
      "resource": [
        "qcs::ci:ap-beijing:uid/1250000000:bucket/examplebucket-1250000000/*"
      ]
    }
  ]
}
```

推荐使用 SDK

数据万象 SDK 提供了完整的临时密钥使用 Demo、集成服务接口、计算签名等能力。您可通过 SDK 方便快捷地调用接口，点此查看 SDK 文档。

图片处理实践
通过 Android 和 iOS 快速接入 AVIF 实践

最近更新时间：2024-12-17 14:32:53

AVIF 简介

AVIF 是一种基于 AV1 视频编码的新一代图像格式，相对于 JPEG、WEBP 这类图片格式来说，它的压缩率更高，并且画面细节更好。AVIF 格式适用于网络环境不稳定、流量不足等情况下访问图片的场景。AVIF 格式图片在保持图片质量不变的情况，尽可能的减小图片大小，以达到节省图片存储空间、减少图片访问流量、提升图片访问速度的效果。最关键的是，AV1 由谷歌发起的 AOM（开放媒体联盟）推动，在 VP9 的基础上继续演进，无专利授权费用（而且腾讯也是 AOM 的创始成员）。

App 显示 AVIF 图片

由于 AVIF 目前已在 iOS16、Android12 以上得到了原生支持，但要想覆盖所有主流机型，单靠原生支持肯定是不够的。因此需要客户端开发时集成 AVIF 解码器自行解码。一般做法是使用下面的解码库，自行编译 Android 和 iOS 解码器产物，以及写一些 JNI 代码，如果您的 App 使用 Glide、SDWebImage 等图片库，还需再按照图片库的要求进行封装集成。

- 业内开源编解码库：开源编解码库。
- 腾讯自研编解码库：本文的数据万象 AVIF SDK 基于该编解码库。

显然，这种方法有不少的工作量，那么有没有更快的方法？您可以选择接入数据万象 AVIF SDK 提高工作效率。本文将为您介绍如何快速集成 AVIF 解码器，兼容所有机型。

数据万象 AVIF 图片 SDK

Android 集成

使用 Glide 图片库

1. 安装 Glide 和 AVIF SDK。

```
implementation 'com.qcloud.cos:avif:1.1.0'
implementation 'com.github.bumptech.glide:glide:version'
annotationProcessor 'com.github.bumptech.glide:compiler:version'
```

2. 注册解码器 GlideModule。

```
// 注册自定义 GlideModule
// 开发者应该创建此类注册相关解码器
// 类库开发者可以继承 LibraryGlideModule 创建类似的注册类
@GlideModule
public class MyAppGlideModule extends AppGlideModule {
 @Override
 public void registerComponents(@NonNull Context context, @NonNull Glide glide, Registry registry) {
     /*------------------解码器 开始-------------------------*/
     //注册 AVIF 静态图片解码器
     registry.prepend(Registry.BUCKET_BITMAP, InputStream.class, Bitmap.class, new
StreamAvifDecoder(glide.getBitmapPool(), glide.getArrayPool()));
     registry.prepend(Registry.BUCKET_BITMAP, ByteBuffer.class, Bitmap.class, new
ByteBufferAvifDecoder(glide.getBitmapPool()));
     //注册 AVIF 动图解码器
     registry.prepend(InputStream.class, AvifSequenceDrawable.class, new
StreamAvifSequenceDecoder(glide.getBitmapPool(), glide.getArrayPool()));
     registry.prepend(ByteBuffer.class, AvifSequenceDrawable.class, new
ByteBufferAvifSequenceDecoder(glide.getBitmapPool()));
     /*------------------解码器 结束-------------------------*/
 }
```

```
}
```

3. 使用 Glide 加载图片。

   像普通 jpg、png 图片那样加载图片即可，请参见 Glide 官方文档。

```
Glide.with(context).load(url).into(imageView);
```

使用 Fresco 图片库

1. 安装 Fresco 和 AVIF SDK。

```
implementation 'com.qcloud.cos:avif:1.1.0'
implementation 'com.facebook.fresco:fresco:version'
// 如果需要支持 avif 动图解码器 则需要加上 fresco:animated-base 依赖
implementation 'com.facebook.fresco:animated-base:version'
```

2. 配置解码器。

```
// 解码器配置
ImageDecoderConfig imageDecoderConfig = new ImageDecoderConfig.Builder()
      // 配置 AVIF 静态解码器
     .addDecodingCapability(
             AvifFormatChecker.AVIF,
             new AvifFormatChecker(),
             new FrescoAvifDecoder())
     // 配置 AVIF 动图解码器
     .addDecodingCapability(
             AvisFormatChecker.AVIS,
             new AvisFormatChecker(),
             new FrescoAvisDecoder())
     .build();
// 配置 Image Pipeline
ImagePipelineConfig config = ImagePipelineConfig.newBuilder(context)
     .setImageDecoderConfig(imageDecoderConfig)
     .build();
// 初始化 Fresco
Fresco.initialize(context, config);
```

3. 使用 Fresco 加载图片。

   像普通 jpg png 图片那样加载图片即可，请参见 Fresco 官方文档。

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="130dp"
    android:layout_height="130dp"
    fresco:placeholderImage="@drawable/my_drawable"
  />


Uri uri = Uri.parse("https://xxx.com/test.avif");
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
draweeView.setImageURI(uri);
```

iOS 集成

1. 安装 SDWebImage 和 AVIF SDK。

在您工程 Podfile 文件中添加模块：

```
pod 'CloudInfinite/SDWebImage-CloudInfinite'
pod 'CloudInfinite/AVIF'
```

在终端执行安装命令：

```
pod install
```

2. **使用 SDWebImage 直接加载 AVIF 图片。**

   SDWebImage-CloudInfinite 模块在 App 启动时已自动将 AVIF 解码器加入到 SDWebImage 解码器队列中，在加载解码器时自动找到 AVIF 解码器来解码图片。支持动图，无需额外操作。使用时与 SDWebImage 使用没有任何区别。

   - Objective-C：

     ```
     [imageView sd_setImageWithURL:[NSURL URLWithString:@"AVIF 图片链接"]];
     ```

   - Swift：

     ```
     UIImageView() .sd_setImage(with: NSURL.init(string: "AVIF 图片链接"))
     ```

数据万象 AVIF SDK 其他功能

基础解码器

用于直接将 AVIF 数据解码为 bitmap、UIImage，以及判断图片数据是否为 AVIF 格式。

- Android：

```
import com.tencent.qcloud.image.avif.Avif;

// 图片的字节数组
byte[] buffer = new byte[XXX];
// 是否是 AVIF 格式
boolean isAvif = Avif.isAvif(buffer);
// 是否是 AVIF 动图
boolean isAvis = Avif.isAvis(buffer);

// 原图解码
Bitmap bitmap = Avif.decode(buffer);

// 宽度等比解码
// 目标宽度
int dstWidth = 500;
Bitmap bitmap = Avif.decode(buffer, dstWidth);

// 区域缩放解码
// 区域左上角x坐标
int x = 0;
// 区域左上角y坐标
int y = 0;
// 区域宽度
int width = 100;
// 区域高度
int height = 100;
// 缩放比，大于1的时候才生效，小于等于1的情况下不作缩放
int inSampleSize = 2;
```

```
Bitmap bitmap = Avif.decode(buffer, x, y, width, height, inSampleSize);
```

- iOS：

```
#import "AVIFDecoderHelper.h"
#import "UIImage+AVIFDecode.h"

//判断是否是 AVIF 格式以及动图格式
// data为图片NSData类型数据
BOOL isAVIF = [AVIFDecoderHelper isAVIFImage:data];

//解码 AVIF 图片
// data为图片NSData类型数据
UIImage * image = [UIImage AVIFImageWithContentsOfData:data];

/ data为图片NSData类型数据
// 缩小两倍 并指定解码的范围( rect 以原图为基准)
UIImage * image = [UIImage AVIFImageWithContentsOfData:imageData scale:2 rect:CGRectMake(x, y, width,
height)];
```

Android 超大图采样图片库

1. 安装 subsampling-scale-image-view 和 AVIF SDK。

```
implementation 'com.qcloud.cos:avif:1.1.0'

implementation 'com.davemorrissey.labs:subsampling-scale-image-view:3.10.0'

// AndroidX 请使用
// implementation 'com.davemorrissey.labs:subsampling-scale-image-view-androidx:3.10.0'
```

2. 获取 SubsamplingScaleImageView 控件并注册解码器。

```
SubsamplingScaleImageView subsamplingScaleImageView =
findViewById(R.id.subsampling_scale_image_view);

// 设置 AVIF 图片解码器
subsamplingScaleImageView.setBitmapDecoderClass(AvifSubsamplingImageDecoder.class);
subsamplingScaleImageView.setRegionDecoderClass(AvifSubsamplingImageRegionDecoder.class);
```

3. 使用 subsampling-scale-image-view 加载图片像普通 jpg、png 图片那样加载图片即可，请参见 subsampling-scale-image-view 官方文档 。

```
// 加载 uri 图片
subsamplingScaleImageView.setImage(ImageSource.uri(uri));

// 加载 assets 图片
subsamplingScaleImageView.setImage(ImageSource.asset("test.avif"));

// 加载 resource 图片
subsamplingScaleImageView.setImage(ImageSource.resource(R.raw.avif));
```

总结

数据万象 AVIF SDK 封装了 AVIF 解码器、Android iOS 常用的图片库生态，直接使用数据万象 AVIF SDK 即可帮您轻松实现将 AVIF 图片显示到 App 中。更加详细的使用说明，请参见 数据万象 Android SDK、数据万象 iOS SDK。

通过图片压缩实现业务降本增效

最近更新时间：2024-12-10 16:21:02

简介

各大企业、平台随着业务的发展，数据量不断扩大，尤其是 Web 中最关键的部分——图片业务，很多人选择将图片存储到 对象存储（Cloud Object Storage，COS）中，这也让客户的存储空间以及流量迅速增大。腾讯云 数据万象（Cloud Infinite，CI）提供了高效、便捷、性价比高的 图片压缩 功能，帮助开发者们在图片质量不变的情况下，大幅度减小图片体积，达到节省存储空间、节省图片访问流量的目的。本文以 AVIF 图片压缩为例，为开发者介绍如何通过 AVIF 图片压缩为业务实现降本增效。
AVIF 图片的兼容性如下：

平台	avif	webp	jpg、png
微信小程序-Android	支持	支持	支持
微信小程序-iOS	支持	支持	支持
Chrome 浏览器	支持	支持	支持
Android 系统	Android12以上版本支持	Android4.3以上版本支持	支持
iOS 系统	iOS16以上版本支持	iOS14以上版本支持	支持
Windows 系统	Windows10以上版本支持	不支持	支持

> **说明：**
> 腾讯云数据万象提供集成 AVIF 解码器的 iOS、Android 终端 SDK，可帮助您快速接入和使用 AVIF。

费用相关

AVIF 图片压缩属于图片高级压缩能力，关于图片压缩费用，请参见 图片高级压缩费用 。

前提条件

- 已登录 数据万象控制台，并开通数据万象服务。
- 已创建和绑定存储桶，详情请参见 存储桶操作 。
- 已 开通图片高级压缩 功能。

操作说明

本文提供两种使用方式达到降本增效的效果，两种方式各有优劣，可根据自身业务情况进行选择。
方式一：访问图片时实时压缩，提高访问速度、降低访问带宽；
方式二：上传图片时实时压缩，减少压缩次数，访问压缩图片时可以提高访问速度、降低访问带宽。

操作步骤

方式一：访问图片时实时压缩

1. 获取图片在 COS 中的 url 链接；
2. 访问图片链接时，在链接后面，添加 avif 压缩参数：`imageMogr2/format/avif`，参数与链接之间用 `?` 连接；
3. 假设原图链接为：`https://example-1250000000.cos.ap-shanghai.myqcloud.com/test.png`，则压缩后图片的访问链接为：`https://example-1250000000.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/avif`

如果您使用了 CDN 分发图片，可以在 COS 控制台 中的**域名与传输管理**页面，开启图片自适应压缩功能，开启后系统会根据 accept 请求头自动判断终端是否支持 avif 格式的图片，若支持则进行实时压缩，不支持则返回原图。

方式二：上传图片时实时压缩

您可以通过在 COS 的上传接口 PUT Object 中添加图片处理请求头，来进行压缩上传，详情请参见 图片上传时处理。

优劣对比

方式	优势	劣势
下载时压缩	1. 接入便捷，无需大的开发 2. 按需压缩，按量付费，无访问时不会产生额外的成本	针对图片数量多，访问频次低的场景，压缩费用可能高于节省的流量费，建议用户通过 CDN 分发图片。
上传时压缩	一次压缩永久保存，访问时无额外压缩费用	1. 如果原图与压缩图各存一份，短期内是通过存储换计算的节省方式，但存储费用长期存在，长期来看存储的费用将高于压缩费用。 2. 如果仅保存压缩图，avif 格式没有 jpg、png 等格式通用，在部分场景中可能存在无法预览的情况。

图文混合水印实战

最近更新时间：2024-12-03 10:15:12

操作场景

数据万象基础图片处理的水印功能支持在图片上添加 图片水印 或 文字水印 。但在现实的业务场景中，经常出现一张图片上既有固定的 Logo 水印，也有动态变化的文字水印（例如用户名）的情况。针对上述场景，我们提供了如下方案供您参考，您可结合实际业务场景，选择合适的接入方案。

方案优势对比

方案	优点	缺点
方案一	接入简单，实现方便。	水印尺寸无法跟随图片大小进行动态变换，无法进行平铺操作。
方案二	在图片尺寸多变的场景下，会达到比方案一更好的自适应效果。	接入流程更为繁琐，且会收取两次处理费用。

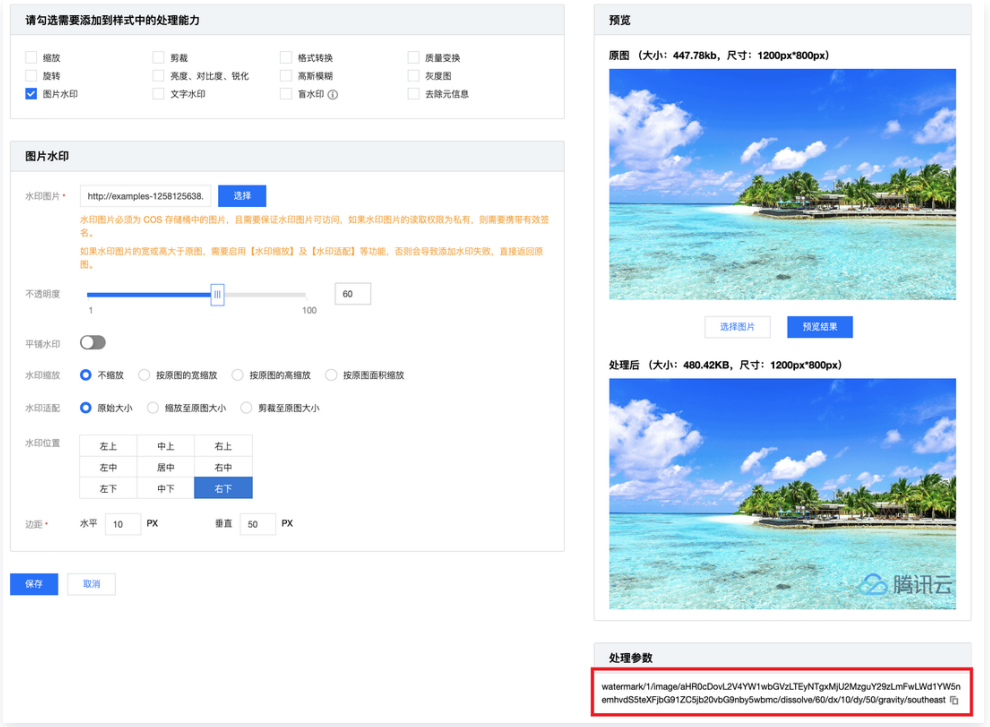
操作方案

方案一：使用管道操作符实现一条 URL 同时打上两种水印

数据万象的基础图片处理功能支持使用 管道操作符 "|"实现一次请求、多次处理，且只计算一次处理和流量费用。使用这种方式可大幅减少重复请求带来的时延和额外费用。

操作步骤

1. 参见 图片水印 API 文档，定义图片水印的参数。
   如您不熟悉 API 参数，可参考 样式管理 文档，通过控制台新增样式，生成参数。



如上图红框所示，处理参数如下：

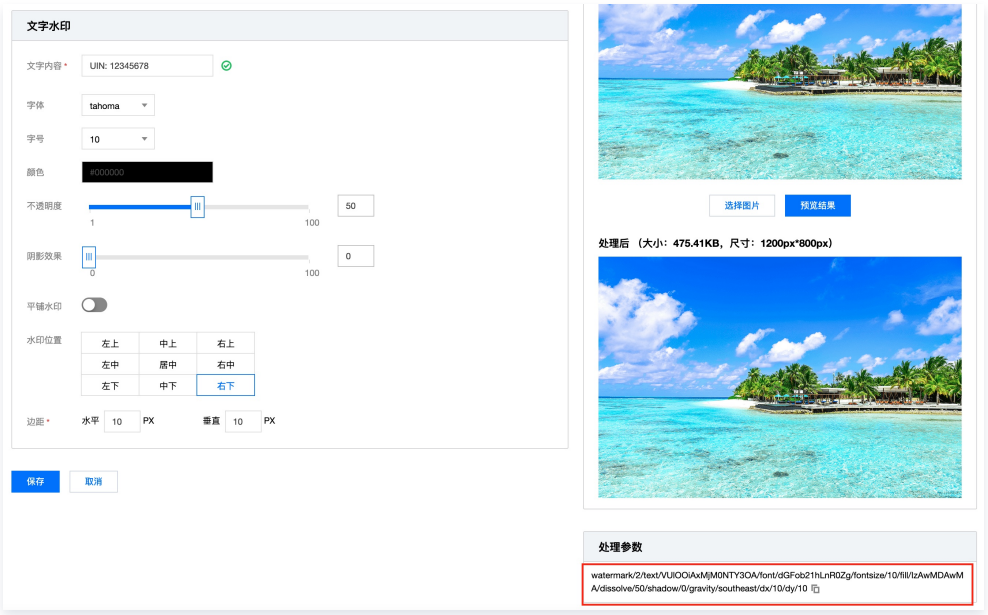
```
watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vbG9nby5nby5
wbmc/dissolve/60/dx/10/dy/50/gravity/southeast
```

> **说明：**

> 此处的编码 `aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vbG9nby5wbmc` 为水印图片链接（即存储在对象存储 bucket 上的图片链接）经过 URL 安全的 base64 编码后生成。

2. 参见 文字水印 API 文档，定义文字水印的参数。

   如您不熟悉 API 参数，可参考 样式管理 通过控制台新增样式，生成参数。




```
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAwMDAwMA/dissolve/50
/gravity/southeast/dx/10/dy/10
```

> **说明：**
> 此处编码 `VUlOOiAxMjM0NTY3OA` 为文字信息 `UIN: 12345678` 经过 URL 安全的 base64 编码后生成。

3. 使用管道操作符拼接图片水印和文字水印的两段参数：

```
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAwMDAwMA/dissolve/50
/gravity/southeast/dx/10/dy/10|watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5n
emhvdS5teXFjbG91ZC5jb20vbG9nby5wbmc/gravity/southeast/dx/10/dy/50/dissolve/60
```

4. 将拼接的参数拼接至图片的下载链接后方：

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAwMDAwMA/dissolve/50
/gravity/southeast/dx/10/dy/10|watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5n
emhvdS5teXFjbG91ZC5jb20vbG9nby5wbmc/gravity/southeast/dx/10/dy/50/dissolve/60
```

即可获得混合水印的图片：

为了缩短 URL 长度，我们可参见 样式管理 文档，在控制台将图片水印（保持不变）的部分新增为样式 `watermark1`：

这样，链接即可缩短为：

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png/watermark1?
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAwMDAwMA/dissolve/50
/gravity/southeast/dx/10/dy/10
```

之后需要更改文字内容时，您只需将链接中 `VUlOOiAxMjM0NTY3OA` 部分替换为更新后的 base64 编码即可实现。例如 `UIN: 88888888` 的编码为 `VUlOOiA4ODg4ODg4OA`，此时只需将链接更改为如下内容，即可实现文字内容的替换。

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png/watermark1?
watermark/2/text/VUlOOiA4ODg4ODg4OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAwMDAwMA/dissolve/50
/gravity/southeast/dx/10/dy/10
```

方案二：将文字和图片水印打印到透明图片上，再将其作为图片水印

1. 准备一张400px * 400px尺寸的透明 PNG 图片，上传至存储桶中。详情可参见 上传文件 文档。
   例如：`https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/transparent.png`
2. 参见**方案一的步骤1 – 步骤3**，生成图片水印和文字水印两段参数，并将其拼接。
3. 将拼接的参数拼接至透明 PNG 图片的下载链接后方：

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/transparent.png?
watermark/2/text/VUlOOiAxMjM0NTY/font/SGVsdmV0aWNhLmRmb250/fontsize/48/fill/IzAwMDAwMA/dissolve/60/gr
avity/south/dx/0/dy/60|watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5nemhvdS5t
eXFjbG91ZC5jb20vbG9nby5wbmc/gravity/center/dx/0/dy/0/dissolve/71
```

其效果如下：

4. 将这张透明图片作为水印图片，对原图进行打水印操作即可：

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?
watermark/1/image/aHR0cDovL2V0ZXJuYXV4LTEzMDE0NTM1NTAuY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vdHJhbnN
wYXJlbnQucG5nP3dhdGVybWFyay8yL3RleHQvVlVsT09pQXhNak0wTlRZL2ZvbnQvU0dWc2RtVjBhV05oTG1SbWIyNTAvZm9udHNp
emUvNDgvZmlsbC9JekF3TURBd01BL2Rpc3NvbHZlLzYwL2dyYXZpdHkvc291dGgvZHgvMC9keS82MHx3YXRlcm1hcmsvMS9pbWFnZ
S9hSFIwY0RvdkwyVjBaWEp1WVhWNExURXpNREUwTlRNMU5UQXVZMjl6TG1Gd0xXZDFZVzVuZW1odmRTNXRlWEZqYkc5MVpDNWpiMj
B2Ykc5bmJ5NXdibWMvZ3Jhdml0eS9jZW50ZXIvZHgvMC9keS8wL2Rpc3NvbHZlLzcx/gravity/southeast/dx/0/dy/0/dissol
ve/90
```

其效果如下：

您还可以通过 `scatype` 参数使水印图尺寸根据图片大小等比例缩放，并通过 `batch` 参数设置平铺。

```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?
watermark/1/image/aHR0cDovL2V0ZXJuYXV4LTEzMDE0NTM1NTAuY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vdHJhbnN
wYXJlbnQucG5nP3dhdGVybWFyay8yL3RleHQvVlVsT09pQXhNak0wTlRZL2ZvbnQvU0dWc2RtVjBhV05oTG1SbWIyNTAvZm9udHNp
emUvNDgvZmlsbC9JekF3TURBd01BL2Rpc3NvbHZlLzYwL2dyYXZpdHkvc291dGgvZHgvMC9keS82MHx3YXRlcm1hcmsvMS9pbWFnZ
S9hSFIwY0RvdkwyVjBaWEp1WVhWNExURXpNREUwTlRNMU5UQXVZMjl6TG1Gd0xXZDFZVzVuZW1odmRTNXRlWEZqYkc5MVpDNWpiMj
B2Ykc5bmJ5NXdibWMvZ3Jhdml0eS9jZW50ZXIvZHgvMC9keS8wL2Rpc3NvbHZlLzcx/scatype/3/spcent/30/gravity/southe
ast/dx/0/dy/0/dissolve/90/batch/1/degree/45
```

其效果如下：

使用数据万象快速制作商品海报

最近更新时间：2025-03-04 20:09:53

简介

本文将介绍如何结合 腾讯云数据万象（CI）的 商品抠图 和 海报合成 能力，快速制作精美的商品海报。制作商品海报首先需要从图片中获取商品主体（商品抠图）、然后用获取到的商品主体进行海报合成。

业务场景

使用数据万象智能商品抠图功能，搭配丰富的海报模板，10秒钟就可以产出一张精美海报，适用于电商活动海报、线上直播邀请函、团购营销等各类需批量海报制作的场景。

准备工作

- 已创建和绑定存储桶，详情请参见 存储桶操作。
- 已 开通媒体服务 功能。
- 上传图片。

操作步骤

步骤1：商品抠图

1. 获取图片对象地址，格式为：`https://test-12XXXXXXX.cos.ap-chongqing.myqcloud.com/test.jpg`。
2. 对图片进行抠图，从而获取商品主体，可参考 商品抠图 API，在图片地址后拼接 `ci-process=GoodsMatting`，即：`https://test-12XXXXXXX.cos.ap-chongqing.myqcloud.com/test.jpg?ci-process=GoodsMatting`。

> **注意：**
> 图片地址需要携带签名，详情请参见 请求签名。

3. 将抠图结果存储在存储桶中，例如：`https://test-12XXXXXXX.cos.ap-chongqing.myqcloud.com/product.jpg`

> **说明：**
> - 商品抠图的计费说明请参见 商品抠图费用。
> - 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问。

商品抠图效果示例：

步骤2：海报合成

提交海报合成任务，返回任务 ID。参见 提交任务接口。接口请求参数格式为：

```
<Request>
 <Tag>PosterProduction</Tag>
 <Operation>
     <Output>
         <Region>ap-chongqing</Region>
         <Bucket>test-12XXXXXXX</Bucket>
         <Object>output/out.jpg</Object>
     </Output>
     <PosterProduction>
         <TemplateId>t10461fe2bd5a649db9022452ec71exxxx</TemplateId>
         <Info>
             <main>https://test-12XXXXXXX.cos.ap-chongqing.myqcloud.com/product.jpg</main>
             <text_main>demo<text_main>
         </Info>
     </PosterProduction>
 </Operation>
 <QueueId>p2911917386e148639319e13c285cxxxx</QueueId>
 <CallBack>http://callback.demo.com</CallBack>
 <CallBackFormat>JSON<CallBackFormat>
</Request>
```

说明：

- PosterProduction.TemplateId 为海报合成模板 ID，推荐使用默认 海报模板。如需使用自定义海报模板，请参见 设计中心文档。
- PosterProduction.Info 为模板的可替换信息，将上一步提取出来的商品图片（`https://test-12XXXXXXX.cos.ap-chongqing.myqcloud.com/product.jpg`）替换到模板中，生成对应的商品海报。

- 海报合成的费用请参见 智能海报生成费用。

步骤3：获取海报合成结果

- 提交任务会返回 JobId 作为任务唯一标识，通过 JobId 查询海报合成任务，请参见 查询任务接口。
- 返回的 State 为 Success 代表已经合成成功，读取到合成的海报地址为：`https://${Operation.Output.Bucket}.cos.${Operation.Output.Region}.myqcloud.com/${Operation.Output.Object}` 。
- 推荐使用 API Explorer调试 。

最终制作的海报效果示例如下：

Demo 体验

- 具体代码可参考 cos-demo 商品海报合成。
- 您可使用腾讯云对象存储控制台，在 智能工具箱 栏目中使用示例文件或自行上传文件，体验商品抠图及海报合成的实际效果。

小程序业务节省图片流量

最近更新时间：2024-12-10 16:21:02

简介

在移动互联网时代，越来越多的人开始用微信小程序来进行各种活动和交流，其中图片的使用非常普遍。然而，图片的高清化、大图化等问题也导致了图片在传输过程中所占用的流量越来越大。腾讯云 数据万象（Cloud Infinite，CI）提供了高效、便捷、性价比高的 图片压缩 功能，保证图片质量的情况下，大幅度减小图片大小，帮助用户节省流量。

数据万象提供的图片压缩能力，能将图片转换为 WebP/AVIF 等压缩格式，这些格式相较于传统的 jpg/png 格式，更加高效，能够在保证图片质量的情况下，大幅度减小图片大小。AVIF 格式的图片相较于其他图片格式，具有压缩率更高，压缩效果更佳的优点，但是目前存在兼容性问题。本文将会为开发者介绍，在微信小程序中如何通过数据万象提供的图片自适应压缩能力让您的小程序无缝兼容 AVIF/WebP 。

AVIF 兼容性说明

AVIF 作为一种前沿的压缩图像格式，在部分较老的设备或应用程序上可能无法正常查看，您在微信小程序中可以通过 accept 头部或 UA /版本来判断是否支持：

- Android 系统小程序：微信版本>=8.0.0 且 Android 系统版本>=12
- IOS 系统小程序：需要 IOS 系统的版本>=16.0

为了解决微信小程序中 AVIF 图片格式的兼容性问题，可以采用下述方案。

前提条件

- 登录 数据万象控制台，并开通数据万象服务。
- 已创建和绑定存储桶，可参考 存储桶操作。
- 将待压缩的图片上传至存储桶中。

实践方案

基础：在小程序中应用图片自适应压缩

数据万象推出的图片自适应压缩，基于万象的图片压缩能力，可以根据终端能力进行最优图片格式压缩，解决前端兼容问题。开启后系统会根据 accept 请求头自动判断终端是否支持预览相应压缩格式的图片，若支持则进行实时压缩，若全都不支持则返回原图。推荐您开启 WebP 与 AVIF 格式，如果不支持 AVIF 图片格式，则可以自动加载 WebP 图片格式，如 WebP 也不支持则自动加载原图。

> **说明：**
> 使用图片自适应压缩，必须通过 CDN 分发图片。

操作步骤如下：

1. 开启自定义 CDN 加速域名，详情可查看 开启自定义 CDN 加速域名。
2. 开启图片自适应压缩功能，推荐开启 WebP 与 AVIF 格式。

3. 获取图片的 CDN 链接。使用 CDN 分发图片，图片链接结构形式为：`< CDN 域名>/<对象键>` 。假设 CDN 域名为：`https://www.cdndomain.com` ，对象键为：`test.png` ，则图片的访问链接为：`https://www.cdndomain.com/test.png` ；
4. 小程序上使用 image 标签加载图片。

```
<image mode="aspectFit" src="https://www.cdndomain.com/test.png"></image>
```

预览效果如下图所示：

以示例图片为例，下表展示了 AVIF 图片和 WebP 图片的压缩率。

图片格式	图片大小	压缩率
原图	1.37MB	-
AVIF 压缩图片	29.05KB	97.9%
WebP 压缩图片	64.59KB	95.3%

> **注意：**
> 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名。

微信小程序中，WebP 图片格式的兼容性较好：

- Android 系统小程序：完全兼容；
- IOS 系统小程序：需要 iOS 系统版本高于 7.0.6。

使用图片自适应压缩，可以满足您业务中大部分图片的压缩需求，但如果您担心用户设备太老，既不支持 AVIF 图片格式，也不支持 WebP 图片格式，想要进一步保证所有图片都能进行压缩，那么在使用图片自适应压缩功能的基础之上，推荐您使用数据万象的图片极智压缩功能。

更多：使用数据万象的图片极智压缩功能

数据万象提供的极智压缩能力，可以在不改变分辨率、不改变图片格式的情况下，对 JPG、PNG 格式的图片进行高效的压缩，大幅压缩图片的体积，且几乎不影响图片画质。

操作步骤如下：

1. 在控制台中开启 图片极智压缩 功能。
2. 当小程序上不支持**图片自适应压缩**的图片格式而返回了原图时，图片会自动进行极智压缩（无需其他操作）。

通过上述操作，即可基本保证在所有场景下都能让图片压缩，节省图片的访问流量。

> ⚠ **注意：**
> 极智压缩目前支持压缩的格式为：PNG、JPG、JPEG、GIF。

费用相关

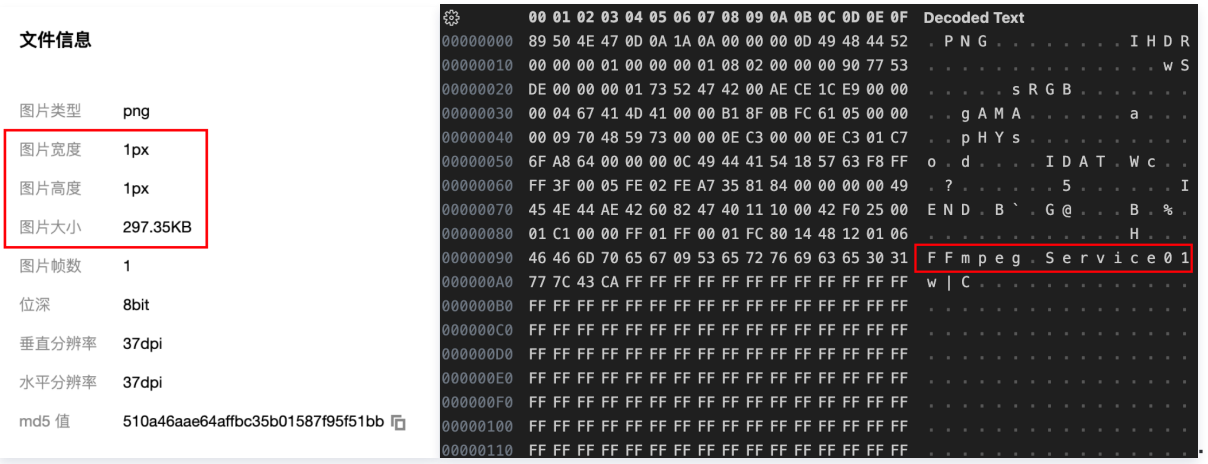
图片压缩费用，请参见 图片处理费用 。

通过异常图片检测防止暗刷流量

最近更新时间：2025-01-09 17:43:22

简介

用户使用对象存储 COS 来搭建个人网站或图床服务，享受高效稳定可靠的数据存储服务的同时，也面临着流量盗刷的问题。COS 提供了多种 防盗刷手段 来帮助开发者合理配置存储桶，降低因类似问题带来的大额资金损失的风险。本文介绍的图片异常检测能力，通过有效识别图片中是否隐含视频等文件的方式，精准检测出异常图片文件，通过删除或者转移异常图片文件，精准打击黑产盗刷行为。
以图床盗链为例，黑客将一段 MPEG-Ts 数据嵌入到 PNG 图片，将伪装后的图片上传到图床获取下载 URL。黑产专用的 M3U8 播放器，从图床下载图片去除伪装后播放。这种图片分辨率很低，体积却很大，且能够正常进行图片解码，缩略图也能正常生成，伪装性极强，能绕过大部分上传检查限制。



针对存储在对象存储 COS 上的图片，数据万象提供了图片异常检测功能，可以检测图片中是否隐含其他类型的可疑文件，该功能有以下三种使用方式：

使用方式	配置类型	说明
图片上传至 COS 后自动检测	通过控制台配置使用	通过配置工作流的方式，当图片上传至配置好的存储桶路径时，会自动触发异常图片检测任务，并对有异常的图片做自动删除等操作。
对 COS 存量图片进行批量检测	通过控制台配置使用	通过配置批量任务的方式，对存储桶中指定范围的图片（如过去一年的A目录中的图片），创建批量异常图片检测任务，创建完成后，后台将自动对这批图片进行检测，并对有异常的图片做自动删除等操作。
使用接口检测 COS 中的单个图片	通过API接口使用	同步检测单个图片中是否包含可疑内容，详情请查看 异常图片检测接口 。

限制及费用说明

- 支持检测的图片格式：JPG、JPEG、PNG、BMP、GIF，以及纯 TS 视频流（更改为任意后缀）。
- 静态图体积限制：可检测的原图大小不超过32MB，宽高不超过50000像素且总像素不超过2.5亿像素。
- 动图体积限制：动图的宽 x 高 x 帧数不超过2.5亿像素。
- 动图帧数限制：帧数限300帧。
- 异常图片检测功能为收费项，由数据万象收取，详细的计费说明请参见数据万象 图片处理费用 。

前提条件

- 已登录 数据万象控制台，并开通数据万象服务。
- 已创建和绑定存储桶，详情请参见 存储桶操作 。

使用方式

方式一：图片上传至 COS 后自动检测

如果您的业务存在上传图片的场景，建议您开启图片上传自动触发异常图片检测功能。您可以通过配置工作流的方式，图片上传至配置好的存储桶路径时，会自动触发异常图片检测任务，并对有异常的图片做自动删除等操作，避免违规数据肆意传播。您可以参考以下步骤进行配置：

1. 进入存储桶管理页面，在左侧导航栏中，选择**任务与工作流 > 工作流管理**，进入工作流管理页面，单击**创建工作流**。

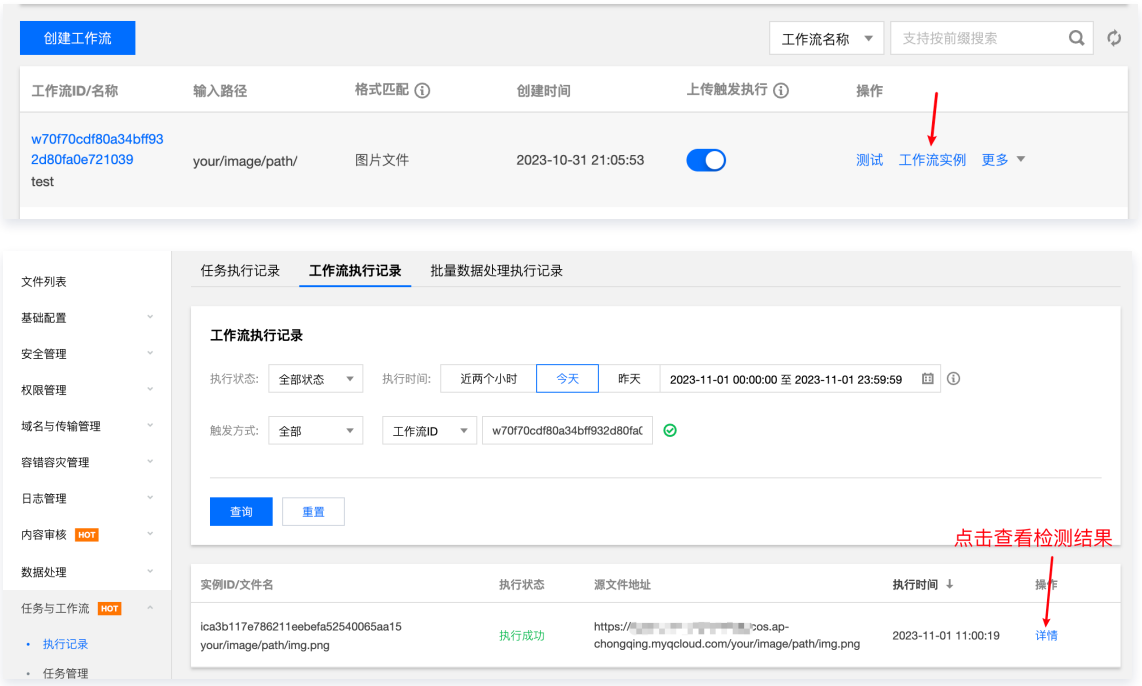
2. 在创建工作流页面，配置如下信息，并创建工作流。

- **工作流名称**：必填项，仅支持中文、英文大小写[A-Z,a-z]、数字[0-9]、下划线(_)和短横线(-)，长度不能超过128个字符。
- **输入路径**：选填项，如果不填写，则对输入存储桶的所有路径生效。工作流启用后，当文件上传至该路径时，工作流将被自动触发。
- **格式匹配**：选择图片文件。
- **配置工作流**：按以下操作配置异常图片检测功能：
  - 单击输入右侧的“+”，选择**异常图片检测**。
  - 在异常图片检测配置弹窗中，按需要选择后续处理的设置，如弹窗顶部提示您需要开通图片处理异步处理服务，请点击开通。
  - 单击**确定**，完成配置。

- **回调设置：可配置回调 URL 获取处理结果，如果异常图片检测的后续处理选择了不处理，建议配置回调URL。**

3. **工作流默认为未启用状态，单击该工作流对应的状态按钮，即可启用工作流。工作流启用后，将在5分钟内生效。工作流生效后，后续上传的文件都将自动进行异常图片检测处理。**

4. 上传图片文件后，单击**工作流实例**查看工作流的执行记录，进入当前执行记录的**详情**，查看异常图片检测的结果。

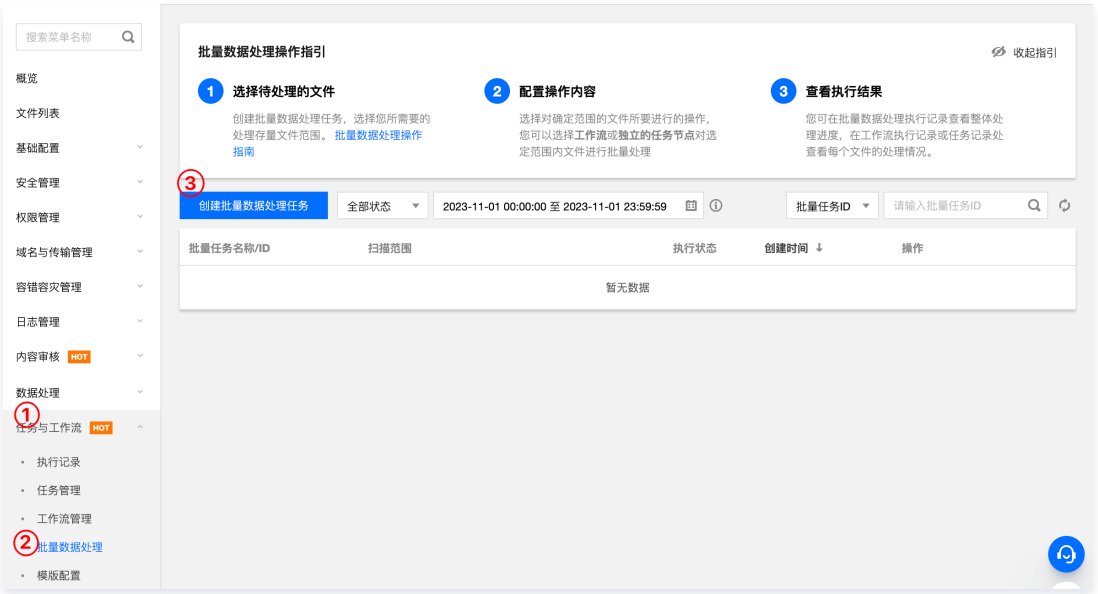


如下所示，检测 img.png 图片时发现存在 297.24KB的异常内容：

方式二：对 COS 存量图片进行批量检测

如果您在 COS 上有旧的图片数据仍然存在被黑产盗刷的风险，建议您创建存量图片批量检测任务，对 COS 上的历史图片进行一次全面的清理，将存在异常的图片进行转移或删除处置。您可以按照以下步骤进行历史数据检测：

1. 进入存储桶管理页面，在左侧导航栏中，选择**任务与工作流 > 批量数据处理**，进入批量数据处理页面，单击**创建批量数据处理任务**。

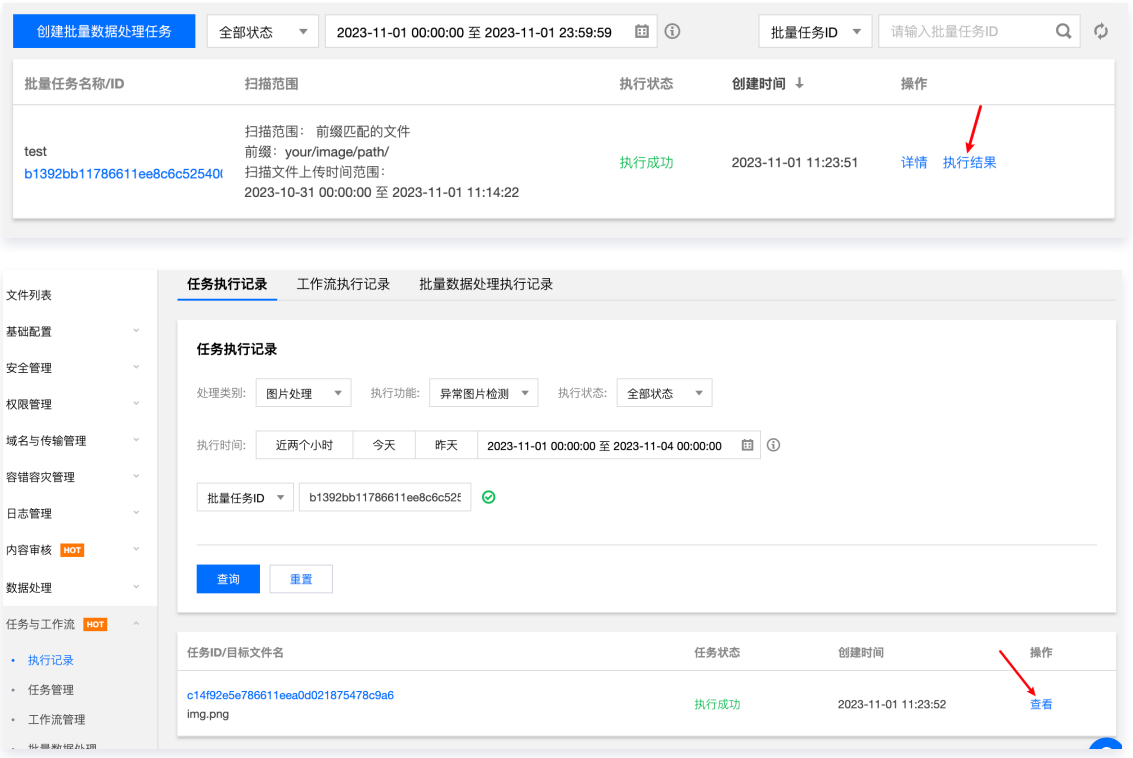


2. 在创建批量数据处理任务页面，配置如下输入设置，并创建批量处理任务。

- **任务名称**：必填项，仅支持中文、英文大小写[A-Z,a-z]、数字[0-9]、下划线(_)和短横线(-)，长度不能超过128个字符。
- **范围**：确定批量处理的数据范围，支持以下几项：
  - 存储桶文件列表：支持按时间、前缀两个维度筛选当前存储桶中的文件。
  - COS 清单报告：选择扫描由 COS 清单功能生成的清单列表，清单列表必须为当前存储桶内的文件。
  - URL 列表文件：选择扫描指定的 url 列表文件，支持 txt 格式，url 列表必须为当前存储桶内的文件。

- **处理类型**：为指定范围内的数据设置操作，如您已配置过异常图片检测的工作流，可选择该**工作流**；如您还未创建过工作流，可选择**独立任务**，任务类型选择**图片处理-异常图片检测**，按需求配置后续处理设置。
- **回调 URL**：可配置回调 URL 获取处理结果，如果异常图片检测的后续处理选择了不处理，**建议配置回调 URL**。

3. 单击**执行结果**，前往任务执行记录页面查看批量执行情况，进入当前任务记录的**详情**，查看异常图片检测的结果。



如下所示，扫描存量图片 img.png 时，发现该图存在 297.24KB的异常内容：

方式三：使用接口检测 COS 中的单个图片

如果您希望在自己的业务代码中使用异常图片检测功能，可以使用接口检测 COS 中的某个图片是否隐含其他类型的可疑文件，接口详情请查看 异常图片检测接口，同时提供了 SDK 供开发者使用。

AI 内容识别实践
自动翻译文件内容

最近更新时间：2024-11-15 21:50:12

简介

数据万象翻译功能，支持对 pdf/docx/pptx/xlsx/txt 等多种格式文档进行多语言类型翻译，最大程度上保留文档的样式与排版，适用于跨境电商、教育培训、跨文化交流等场景。
本文将介绍如何使用 腾讯云数据万象（CI）提供的翻译能力，快速翻译您的文档文件，支持多种语言类型和文档类型。

> **说明：**
> 本文介绍的翻译场景适用于文件翻译，如您有短文本的翻译需求，请参见 实时文字翻译。

前提条件

- 已创建和绑定数据万象存储桶，详情请参见 存储桶操作。
- 进入存储桶详情，开启 **数据处理 > 内容识别 > 机器翻译** 功能。
- 上传待翻译的文件。

操作步骤

步骤一：初始化 COS SDK 并配置相关信息

```
// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '*******',
  SecretKey: '**********',
});
```

> **说明：**
> - 注意：该初始化方式仅供联调测试使用，为了安全起见，请勿在生产环境直接暴露密钥。
> - 生产环境请参考各语言 SDK 签名实现，详情请参见 SDK 签名实现 。

步骤二：创建翻译任务

提交翻译任务，参数详情参见 提交任务接口。接口请求参数格式为1：

```
// 存储桶配置请在 cos 控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
// 源文档相关配置
const InputConf = {
   Bucket: 'test-125********',
   Region: 'ap-shanghai',
   FileName: 'input.pdf',
};
// 翻译结果文档相关配置
const OutputConf = {
   Bucket: 'test-125********',
   Region: 'ap-shanghai',    /* 注意：需与源文档所在存储桶为同地域 */
   FileName: 'output.pdf',
};
const host = InputConf.Bucket + '.ci.' + InputConf.Region + '.myqcloud.com';

// 创建翻译任务请求及参数格式
```

```
const body = COS.util.json2xml({
  "Request": {
    "Tag": "Translation", /* 创建任务的 Tag：Translation ，必须*/
    "Input": {
      "Object": InputConf.FileName, /* 需要翻译的文档文件，存储桶里的路径，必须 */
      "Lang": "zh", /* 源文档语言类型，支持简（繁）体中文、英语、德语、法语、日语、俄语、韩语、阿拉伯语等多种语言，必须
*/
      "Type": "pdf", /* 源文档类型，支持 pdf、docx、pptx、xlsx、txt、xml、html、markdown、jpg、png 等多种文档格
式，必须 */
    },
    "Operation": {
      "Output": {
        "Region": OutputConf.Region, /* 存储桶的地域，必须 */
        "Bucket": OutputConf.Bucket, /* 存储结果的存储桶，必须 */
        "Object": OutputConf.FileName /* 输出结果的文件名，必须 */
      },
      "Translation": {
        "Lang": "en", /* 目标文档语言类型，支持多种语言，必须 */
        "Type": "pdf" /* 目标文档类型，必须 */
      },
    }
  }
});
// 创建翻译任务
cos.request({
  Bucket: InputConf.Bucket,
  Region: InputConf.Region,
  Method: 'POST',
  Url: 'https://' + host + '/jobs',
  Key: '/jobs', /** 固定值，必须 */
  ContentType: 'application/xml', /** 固定值，必须 */
  Body: body
}, (err, data) => {
  console.log(err || data);
})
```

说明：

- 翻译计费规则参见 内容识别费用 – 文字翻译。
- 通过子账号使用时，需要授予相关的权限，详情请参见 授权粒度详情 文档。
- 接口返回参数请参见响应参数，JobsDetail 节点下为翻译任务接口响应信息。其中 JobId 为关键信息，步骤三中查询翻译任务时会用到。

步骤三：获取翻译结果

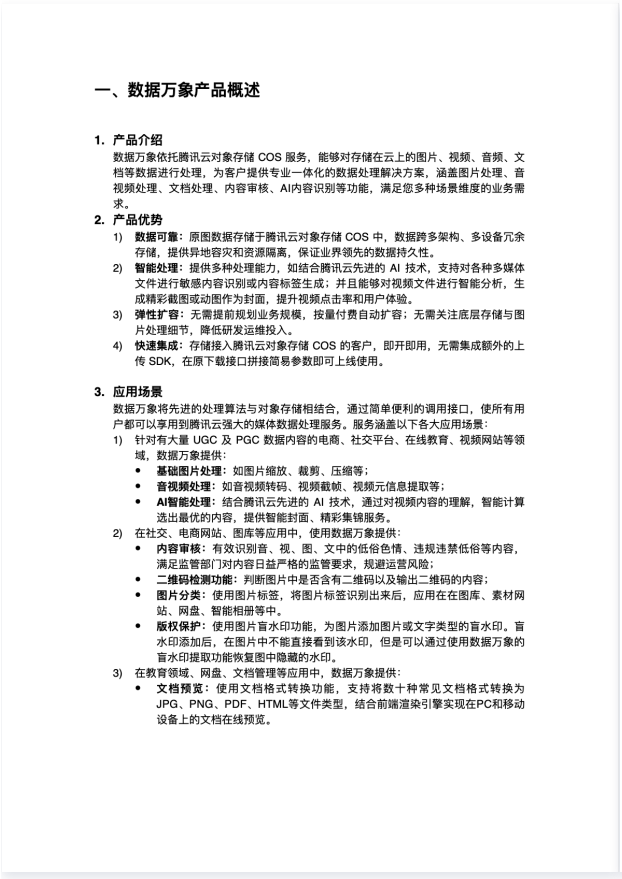
步骤二中提交任务后返回的 JobId 作为任务唯一标识，通过 JobId 查询翻译任务，详情请参见 查询任务接口。
可定时查询任务的状态，当返回的 State 为 Success 时代表文件翻译成功，翻译后的文件地址为：
`https://${Operation.Output.Bucket}.cos.${Operation.Output.Region}.myqcloud.com/${Operation.Output.Object}` 。

```
const getResult = (jobId) => {
  // 查询任务状态及参数格式
  cos.request({
    Bucket: InputConf.Bucket,
    Region: InputConf.Region,
    Method: 'GET',
    Url: 'https://' + host + `/jobs/${jobId}`,
    Key: `/jobs/${jobId}` /** 固定值，必须 */
  }, (err, data) => {
    if(err) {
```

```
        console.log(JSON.stringify(err));
        return;
      }
      if (data?.Response?.JobsDetail?.State === 'Success') {
        console.log("success");
      } else if (data?.Response?.JobsDetail?.State === 'Failed') {
        console.log("Failed");
      }
    })
}
```

说明：
推荐使用 API Explorer调试。

以 pdf 文档为例，翻译前后的效果对比如图：



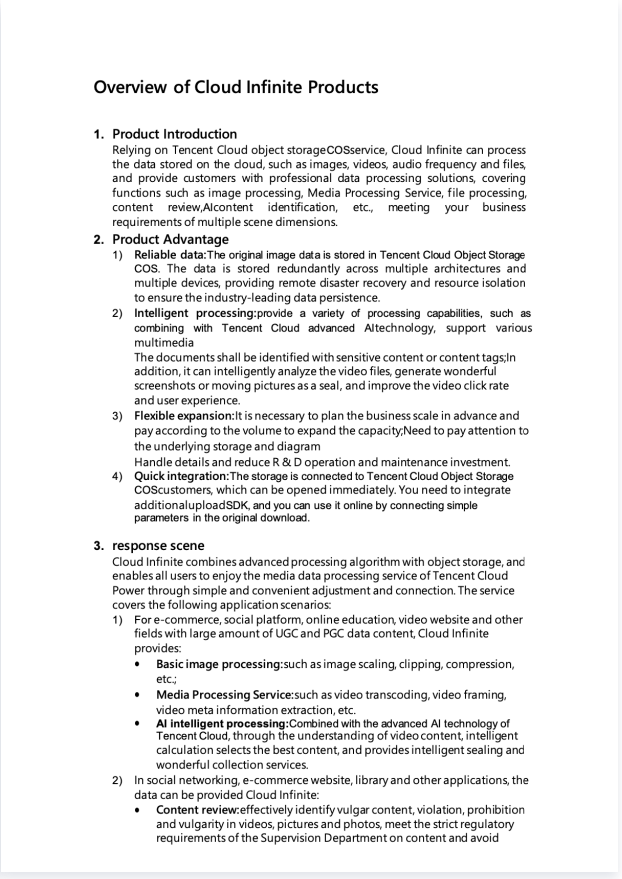
一、数据万象产品概述

1. 产品介绍
数据万象依托腾讯云对象存储 COS 服务，能够对存储在云上的图片、视频、音频、文档等数据进行处理，为客户提供专业一体化的数据处理解决方案，涵盖图片处理、音视频处理、文档处理、内容审核、AI内容识别等功能，满足您多种场景维度的业务需求。
2. 产品优势
1) 数据可靠：原图数据存储于腾讯云对象存储 COS 中，数据跨多架构、多设备冗余存储，提供异地容灾和资源隔离，保证业界领先的数据持久性。
2) 智能处理：提供多种处理能力，如结合腾讯云先进的 AI 技术，支持对各种多媒体文件进行敏感内容识别或内容标签生成；并且能够对视频文件进行智能分析，生成精彩截图或动图作为封面，提升视频点击率和用户体验。
3) 弹性扩容：无需提前规划业务规模，按量付费自动扩容；无需关注底层存储与图片处理细节，降低研发运维投入。
4) 快速集成：存储接入腾讯云对象存储 COS 的客户，即开即用，无需集成额外的上传 SDK，在原下载接口拼接简易参数即可上线使用。

3. 应用场景
数据万象将先进的处理算法与对象存储相结合，通过简单便利的调用接口，使所有用户都可以享用到腾讯云强大的媒体数据处理服务。服务涵盖以下各大应用场景：
1) 针对有大量 UGC 及 PGC 数据内容的电商、社交平台、在线教育、视频网站等领域，数据万象提供：
- 基础图片处理：如图片缩放、裁剪、压缩等；
- 音视频处理：如音视频转码、视频截帧、视频元信息提取等；
- AI智能处理：结合腾讯云先进的 AI 技术，通过对视频内容的理解，智能计算选出最优的内容，提供智能封面、精彩集锦服务。

2) 在社交、电商网站、图库等应用中，使用数据万象提供：
- 内容审核：有效识别音、视、图、文中的低俗色情、违规违禁低俗等内容，满足监管部门对内容日益严格的监管要求，规避运营风险；
- 二维码检测功能：判断图片中是否含有二维码以及输出二维码的内容；
- 图片分类：使用图片标签，将图片标签识别出来后，应用在在图库、素材网站、网盘、智能相册等中。
- 版权保护：使用图片盲水印功能，为图片添加图片或文字类型的盲水印。盲水印添加后，在图片中不能直接看到该水印，但是可以通过使用数据万象的盲水印提取功能恢复图中隐藏的水印。

3) 在教育领域、网盘、文档管理等应用中，数据万象提供：
- 文档预览：使用文档格式转换功能，支持将数十种常见文档格式转换为JPG、PNG、PDF、HTML等文件类型，结合前端渲染引擎实现在PC和移动设备上的文档在线预览。



Overview of Cloud Infinite Products

1. Product Introduction
Relying on Tencent Cloud object storageCOSservice, Cloud Infinite can process the data stored on the cloud, such as images, videos, audio frequency and files, and provide customers with professional data processing solutions, covering functions such as image processing, Media Processing Service, file processing, content review,AIcontent identification, etc., meeting your business requirements of multiple scene dimensions.
2. Product Advantage
1) Reliable data:The original image data is stored in Tencent Cloud Object Storage COS. The data is stored redundantly across multiple architectures and multiple devices, providing remote disaster recovery and resource isolation to ensure the industry-leading data persistence.
2) Intelligent processing:provide a variety of processing capabilities, such as combining with Tencent Cloud advanced AItechnology, support various multimedia
The documents shall be identified with sensitive content or content tags;In addition, it can intelligently analyze the video files, generate wonderful screenshots or moving pictures as a seal, and improve the video click rate and user experience.
3) Flexible expansion:It is necessary to plan the business scale in advance and pay according to the volume to expand the capacity;Need to pay attention to the underlying storage and diagram
Handle details and reduce R & D operation and maintenance investment.
4) Quick integration:The storage is connected to Tencent Cloud Object Storage COScustomers, which can be opened immediately. You need to integrate additionaluploadSDK, and you can use it online by connecting simple parameters in the original download.

3. response scene
Cloud Infinite combines advanced processing algorithm with object storage, and enables all users to enjoy the media data processing service of Tencent Cloud Power through simple and convenient adjustment and connection. The service covers the following application scenarios:
1) For e-commerce, social platform, online education, video website and other fields with large amount of UGC and PGC data content, Cloud Infinite provides:
- Basic image processing:such as image scaling, clipping, compression, etc.;
- Media Processing Service:such as video transcoding, video framing, video meta information extraction, etc.
- AI intelligent processing:Combined with the advanced AI technology of Tencent Cloud, through the understanding of video content, intelligent calculation selects the best content, and provides intelligent sealing and wonderful collection services.

2) In social networking, e-commerce website, library and other applications, the data can be provided Cloud Infinite:
- Content review:effectively identify vulgar content, violation, prohibition and vulgarity in videos, pictures and photos, meet the strict regulatory requirements of the Supervision Department on content and avoid

费用相关

- 翻译费用
- 对象存储相关费用

示例代码

详细代码可参见 cos-demo。

图片瑕疵修复

最近更新时间：2024-10-17 21:07:02

简介

图片修复功能支持指定图片中需要修复的区域，并对其中的主体进行识别，通过 AI 算法智能填充与周围区域相似的图片纹理，从而快速、准确地修复图片中的瑕疵，提高工作效率、降低成本。本文将介绍如何使用 腾讯云数据万象（CI）的 图像修复 能力，快速涂抹修复图片瑕疵。

业务场景

- 摄影后期制作时，修复图片瑕疵及背景杂物，帮助摄影师节省时间和精力。
- 修复老照片中的划痕、污渍等瑕疵。
- 电商平台中的商品图片修复，提升商品图片的质量和美观度，吸引更多的消费者。

前提条件

- 创建数据万象存储桶，详情请参见 存储桶操作。
- 上传待处理的图片 到存储桶中。

步骤一：初始化 Canvas 画布

根据原图大小来初始化 Canvas 画布，原图为存储在存储桶中的图片：

```
// 原图存储在Bucket中，原图地址为：
const imgSrc = `https://${config.Bucket}.cos.${config.Region}.myqcloud.com/${config.FileName}`;

const img = new Image();
img.crossOrigin = 'anonymous'; // 跨域设置，必须
img.onload = () => {
    // 获取图片尺寸初始化画布
    const context = canvas.getContext('2d');
    originWidth = img.width;
    originHeight = img.height;
    canvas.width = originWidth;
    canvas.height = originHeight;
    context.clearRect(0, 0, originWidth, originHeight);
};
img.src = imgSrc;
```

说明：

- 若存储桶为私有读，则对象地址需要携带签名，详情请参见 请求签名。
- 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问。

原图效果示例：

步骤二：制作包含涂抹修复区域的遮罩图

下面以鼠标标记涂抹区域为例，介绍制作遮罩图的方法。

```
// 使用鼠标标记涂抹区域
function setCanvasDraw(context) {
    const context = canvas.getContext('2d');
    let isDrawing = false;
    let y = 0;
    // 画线条标记涂抹区域
    const drawLine = (context, x0, y0, x1, y1) => {
      context.beginPath();
      context.strokeStyle = 'white';     // 涂抹区域为白色
      context.lineWidth = 30;
      context.moveTo(x0, y0);
      context.lineTo(x1, y1);
      context.lineJoin = 'round';
      context.lineCap = 'round';
      context.stroke();
    };

    // 鼠标按下：开始涂抹
    canvas.addEventListener('mousedown', event => {
      x = event.layerX;
      y = event.layerY;
      isDrawing = true;
    });

    // 鼠标移动：持续涂抹
    canvas.addEventListener('mousemove', event => {
      if (isDrawing) {
        drawLine(context, x, y, event.layerX, event.layerY);
        x = event.layerX;
```

```
        y = event.layerY;
      }
    });
    // 鼠标抬起：结束涂抹
    canvas.addEventListener('mouseup', event => {
      if (isDrawing) {
        drawLine(context, x, y, event.layerX, event.layerY);
        x = 0;
        y = 0;
        isDrawing = false;
      }
    });
}
```

遮罩图示例（白色为涂抹区域）：

> **注意：**
> 遮罩图需与原图相同尺寸，背景为黑色，需要涂抹的区域为白色。

最后将 Canvas 遮罩图上传到存储桶中，详情请参见 上传图片 。

步骤三：修复图片并展示

获取原始图片地址，格式为：

```
https://${config.Bucket}.cos.${config.Region}.myqcloud.com/source.jpg
```

获取遮罩图地址，格式为：

```
https://${config.Bucket}.cos.${config.Region}.myqcloud.com/mask.jpg
```

获取修复后的图片，有三种方法：

- 下载时处理，适用于直接下载并展示修复后图片的场景。使用方法可参见 图像修复 API，在图片地址后拼接 ci-process=ImageRepair 、访问签名q-sign-algorithm=<signature> 和 MaskPic=遮罩图地址 URL 安全的 Base64 编码，即：

```
https://${config.Bucket}.cos.${config.Region}.myqcloud.com/source.jpg?q-sign-algorithm=
<signature>&ci-process=ImageRepair&MaskPic=aHR0cDovL2V4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

- 上传时处理，适用于将修复图持久化存储的场景。使用方法可参见 上传时处理 ，在上传原图的时候在请求头中加入处理参数，即：

```
PUT /source.jpg HTTP/1.1
Host: ${config.Bucket}.cos.${config.Region}.myqcloud.com
Date: Wed, 28 Oct 2022 20:32:00 GMT
Authorization:XXXXXXXXXXX
Pic-Operations: {"is_pic_info":1,"rules":[{"fileid":"result.jpg","rule":"ci-
process=ImageRepair&MaskPic=aHR0cDovL2V4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"}]}
Content-Length: XXX

[Image Object]
```

- 云上数据处理，适用于原图已存储在存储桶中，并且需要将修复后的图片存入到存储桶中的场景。使用方法可参见 云上数据处理 ：

```
POST /source.jpg?image_process HTTP/1.1
Host: ${config.Bucket}.cos.${config.Region}.myqcloud.com
Date: Wed, 28 Oct 2022 20:32:00 GMT
Authorization: XXXXXXXXXX
Pic-Operations: {"is_pic_info":1,"rules":[{"fileid":"result.jpg","rule":"ci-
process=ImageRepair&MaskPic=aHR0cDovL2V4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"}]}
Content-Length: XXX
```

图片修复后效果如下：

说明：

- 图片修复计费规则参见 图像修复费用 。

- 图片存储在存储桶中，涉及到 对象存储相关费用 。

Demo 体验

- 具体代码可参见 cos-demo 图片瑕疵涂抹修复 。
- 您可使用腾讯云对象存储控制台，在 智能工具箱 栏目中使用示例图片或自行上传图片，体验图片修复的实际效果。

图像超分实践

最近更新时间：2024-10-17 20:10:11

图像超分技术介绍

图像超分技术是一种通过使用深度学习算法，将低分辨率图像转换为高分辨率图像的计算机视觉方法。这种技术可以有效地恢复图像中的细节和纹理，提高图像质量，使图像看起来更加清晰。超分技术广泛应用于摄影、视频处理、医疗成像等领域，提高了视觉效果和信息的准确性。
腾讯云数据万象图像服务将图片放大为原图的 2 倍，支持PNG、JPEG、BMP等图片格式，支持多种超分模型，可以根据业务需要选择最适合的超分模型，达到最佳的效果。接口格式可以查看 数据万象图像超分 文档。

应用场景

图像超分可以应用于很多场景，只要需要提高图像清晰度和质量的场景都可以考虑使用图像超分技术。

- 电商领域：展示清晰的产品图片可以提升用户购买意愿；
- 医疗行业：高清晰度的医学影像可以帮助医生更准确地诊断病情；
- 智能城市建设：高清晰度的监控画面可以提高安全性。

使用方式

数据万象图像超分有多种使用方式，开发者可根据业务场景选用：

使用方式	适用场景
下载时，处理桶里的图片	图片在 COS 桶中，需要调用 API 实时处理
下载时，处理第三方 Url	图片不在 COS 桶中，需要调用 API 实时处理
上传时，处理图片并转存到 COS	图片上传时进行超分处理，并将结果存储在 COS，需要调用 API 主动触发
云上处理，处理并转存到 COS	图片在 COS 桶中，希望处理后将结果也存储在 COS中

以下通过示例和操作指引，具体介绍这几种使用方式：

下载时，处理桶里的图片

在控制台复制文件地址，链接加上 Url 参数 `?ci-process=AISuperResolution`，在浏览器访问能看到图片放大为原图的2倍，细节也更清晰。访问链接的 `ci-process` 参数需要加入签名计算里。
示例如下：

```
http://examples-1251000004.cos.ap-shanghai.myqcloud.com/sample.jpeg?ci-process=AISuperResolution&q-ak=xxx&...&q-signature=xxx
```

下载时，处理第三方 Url

存储桶域名根链接加上图像超分参数和第三方 Url 参数 `/?ci-process=AISuperResolution&detect-url=<detect-url>`
其中 `ci-process`、`detect-url` 参数需要加入签名计算里。
示例如下：

```
http://examples-1251000004.cos.ap-shanghai.myqcloud.com/?ci-process=AISuperResolution&detect-url=http%3A%2F%2Fexamples-1251000004.cos.ap-shanghai.myqcloud.com%2Fsample.jpeg&q-ak=xxx&...&q-signature=xxx
```

上传时，处理并转存到COS

上传时处理，上传图片到存储桶时 Header加上 `Pic-Operations` 参数，实现图像超分处理并将处理后的图片存储到 COS。参数格式例子：
`Pic-Operations: {"is_pic_info":1,"rules": [{"fileid": "xxx.super.jpg","rule": "ci-process=AISuperResolution"}]}`
可以调用熟悉语言的 COS SDK 做文件上传时处理，以 Nodejs 为例上传时处理示例如下：

```
cos.putObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'img/1.jpg',
    Body: file,
    Headers: {
        // 指定 rule 为图像超分处理，处理后另存为 img/1.super.jpg
        'Pic-Operations': '{"is_pic_info":1,"rules": [{"fileid": "img/1.super.jpg","rule": "ci-process=AISuperResolution"}]}',
    },
}, function (err, data) {
    console.log(err || data);
});
```

云上处理，处理并转存到COS

原图已经在存储桶里，可以调用云上处理 API 对原图做图像超分处理，转存为另一个图片。
调用云上处理时 Header 加上 `Pic-Operations` 参数。参数格式例子：
`Pic-Operations: {"is_pic_info":1,"rules": [{"fileid": "xxx.super.jpg","rule": "ci-process=AISuperResolution"}]}`
可以调用熟悉语言的 COS SDK 做文件上传时处理，以 Nodejs 为例上传时处理示例如下：

```
cos.request({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'img/1.jpg',
    Method: 'POST',
    Action: 'image_process',
    Headers: {
        // 通过 AISuperResolution 接口使用图像超分功能，对已有的文件 img/1.jpg 处理为 img/1.super.jpg
        'Pic-Operations': '{"rules": [{"fileid": "img/1.super.jpg", "rule": "ci-process=AISuperResolution"}]}',
    },
}, function (err, data) {
    console.log(err || data);
});
```

Demo体验

1. 下载 代码示例 到本地，修改 server/app.js 里 config 的参数，指定密钥、存储桶信息。
2. 安装 nodejs，cos-demo 目录下执行 `npm i` 安装依赖，并执行 `npm run dev` 启动调试服务器，在浏览器访问 http://127.0.0.1:3000/image-super-resolution 。
3. 在页面上点击上传文件，体验示例。

> **说明：**
> 1. 体验demo，需要到腾讯云 COS控制台 创建存储桶，并进入存储桶详情->数据处理->图片处理开启数据万象并绑定当前存储桶。
> 2. 密钥可前往腾讯云 CAM控制台 获取。

demo效果如下：

图像超分辨率示例

COS图像超分辨率文档链接

使用方式一、下载时，处理桶里的对象：

选择本地文件

使用方式二、下载时，传入 Url 处理：

https://test-125[redacted].cos.ap-guangzhou.myqcloud.c 提交 Url 处理

使用方式三、上传时，传参处理和转存：

选择本地文件

文件上传成功！

原图：

图片超分辨率图处理后的图：

费用相关

- 图像超分根据**图片处理次数**收费，费用参考 **计费说明** 。
- **对象存储相关费用**

媒体处理实践
音视频处理概述

最近更新时间：2024-12-04 15:58:22

简介

音视频转码提供音频、视频等媒体类文件的转码能力，是文件码流转换成另一个码流的过程。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。
本文将介绍如何使用音视频转码及媒体智能处理服务，将您存储在 COS 上的音视频文件，通过高效、智能的方式，转换成适合在移动端、PC、TV 上播放的格式，实现多端流畅高清播放体验。

适用场景

多终端适配

针对不同用户不同终端需要提供不同格式的媒体文件，将视频转换成流畅、标清、高清及超清等，用户可以根据当前网络环境选择合适码率的视频播放。同时提供多样的压缩功能以提高压缩效率、减小文件体积，从而减少卡顿并节省存储空间和流量费用。

智能编辑

支持视频添加片头片尾、精彩片段剪辑、自动化抠图、人声和背景声分离等二次创作场景；也可以采用超分辨率、色彩和细节增强、SDR to HDR 变换、音频降噪等技术对老旧片源进行修复和美化。

版权保护

可在转码时为视频添加特定图文 LOGO 来说明视频版权归属，也可以添加数字水印，以作版权追溯使用。

功能支持

提供了基础处理、智能编辑、画质优化和版权保护等方面的处理能力：

具体功能说明如下：

功能	说明
格式转换	音频、视频等媒体类文件的转码能力，支持常见的音视频封装格式：avi、mp4、mkv、flv、hls、ts、mp3、aac、flac，支持H.264、H.265、AV1、VP8、VP9 等多种编码格式。
倍速转码	适用于时长较长、文件较大、对转码时效性要求高的传媒场景和数据量大的超高清视频生产，倍速转码最高可以达到30倍速的转码性能。
极速高清	提供让视频更小更清晰的转码能力，保证低带宽的同时带给用户视觉上更佳体验。
广电格式转码	支持 XAVC 、ProRes 等特殊格式转码。
自适应码率	将单个源视频一步生成多码率自适应文件的能力，让视频适配不同播放终端与网络情况，支持hls和dash打包格式。
HDR 转 SDR	使转换后视频的画面细节最大程度贴近原视频，适配不同类型的终端设备，避免画面出现失真、灰暗的情况。
截帧	对视频某一时间节点的截图功能，支持同步和异步截帧，支持将多个截帧拼成雪碧图。
转动图	将视频格式文件转为动图格式文件，支持gif和webp格式。
视频拼接	可将指定的音视频片段拼接在音视频文件的开头或结尾，生成一个新的文件。
视频分段	可将指定的音视频文件切分成若干个片段。
智能封面	智能分析视频帧的质量、精彩程度、内容相关度，提取最优帧生成截图作为封面。

精彩集锦	精准地提取视频中的精彩片段，对视频的内容、动作姿态、场景进行多维度识别与聚合，以匹配专业编辑的水准迅速剪辑生成视频中精彩集锦。
视频标签	对视频中视觉、场景、行为、物体等信息进行分析，输出视频的多维度内容标签。
人声分离	将指定的音视频文件中人声与背景声进行分离生成独立音频素材，便于后期实现其他风格艺术加工。
音频降噪	语音智能降噪，听到想听的声音。
语音合成	通过深度学习技术，将文本转换成自然流畅的语音。目前有多种音色可供选择。
语音识别	对音视频中的音频进行识别，支持中文普通话、英语和粤语。
质量评分	采用无参考评价方式对视频清晰度、信噪比、色彩、亮度等进行综合分析输出视频质量分值。
画质增强	超分辨率：通过识别视频的内容与轮廓高清重建视频的细节与局部特征，通过一系列低分辨率的图像来得到一幅高分辨率的图像 细节和色彩增强：色彩智能美化、饱和度调节；细节智能优化，曲线增强。 SDR to HDR：伽马曲线智能变换，亮度增强。
水印 LOGO	将静态或动态图片、文字水印添加至视频中，支持水印漂浮效果。
数字水印	将图片、字符串隐藏在视频、图文中，水印不易被探知和再次修改，同时也不会破坏视频载体的完整性与可观赏性。
视频加密	通过 HLS 标准加密和 DASH 加密方式对视频数据进行加密，保障视频安全。
视频元信息	获取视频、音频、字幕类等媒体文件的元信息。

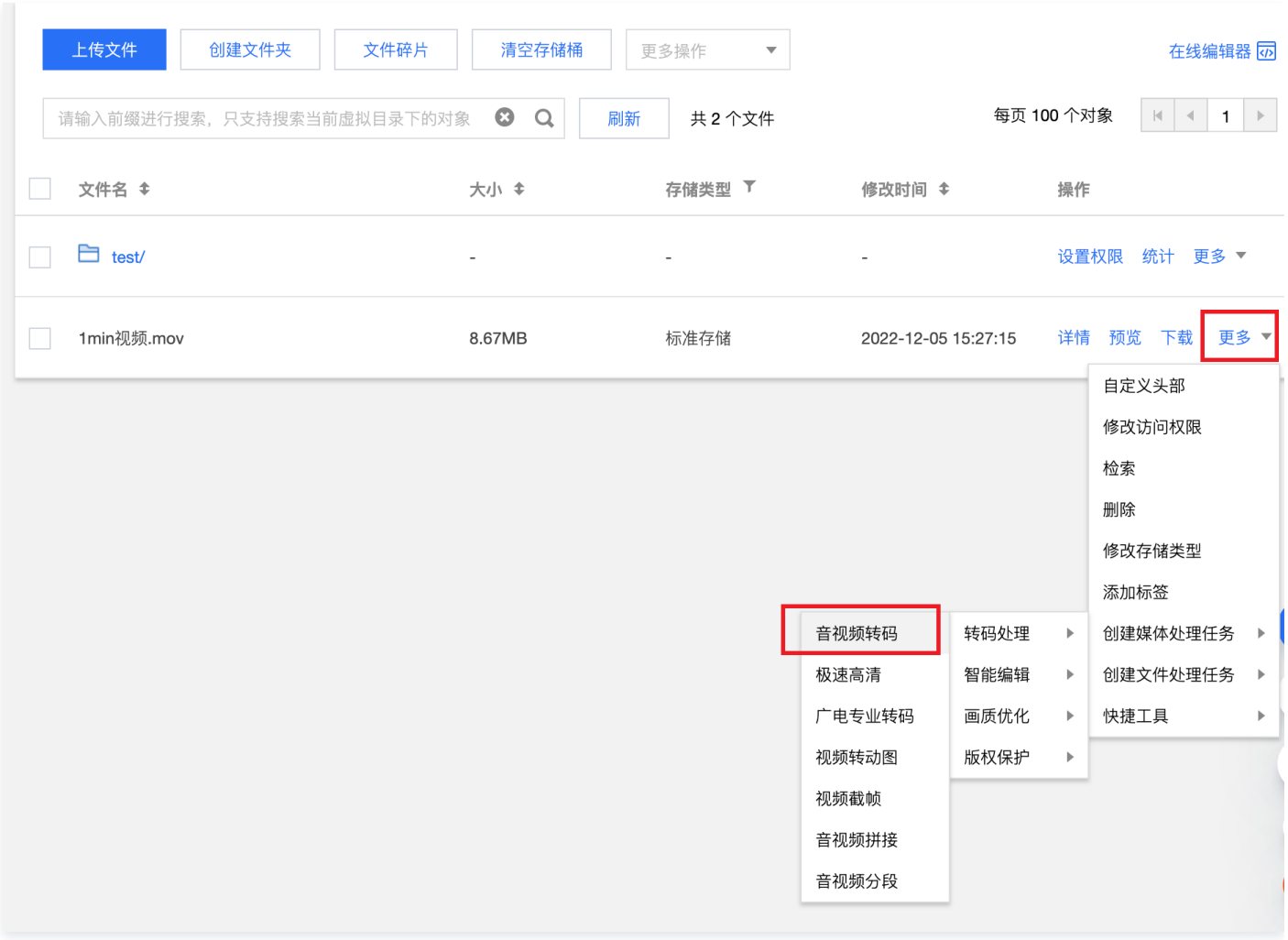
使用方式

以音视频普通转码功能为例，根据场景需要提供以下三种使用方式，这三种方式也可以组合使用：

方式一：创建单个处理任务

适用于对单个媒体文件进行转码处理的场景，步骤如下：

1. 选择待处理的COS媒体文件，并创建**音视频转码**任务。



2. 配置处理参数，具体参数参考 创建音视频转码任务 。

> **说明：**
> - 您也可以使用 提交音视频转码任务接口 来创建单个任务。
> - 转码任务的计费说明请参见 媒体处理费用 。

方式二：创建批量处理任务

适用于对于存储在cos中的存量媒体文件，进行统一转码处理的场景。步骤如下：

1. 选择一个文件夹，创建批量数据处理任务，可参考 批量数据处理 。

2. 在控制台上查看批量任务的执行情况。

> **说明：**
> - 您也可以使用 批量任务接口 来创建批量处理任务。
> - 转码任务的计费说明请参见 媒体处理费用 。

方式三：上传文件触发任务执行

适用于持续有新的媒体文件上传到COS的场景，将转码处理配置到工作流中，设置文件上传触发工作流执行，确保新上传的视频都会经过转码处理。步骤如下：

1. 选择一个文件路径创建工作流

2. 配置工作流参数，详情可参见 配置工作流 。
3. 查看执行结果
4. 可以在控制台上查看执行结果，可参见 查看执行实例 。
5. 工作流支持自定义设置回调 URL，在工作流开启状态下，每上传一个文件会触发工作流执行一个实例，实例执行完成后系统会向该 URL 发送 HTTP POST 请求，请求体中包含 回调通知内容 ，以便及时了解工作流实例处理的进展和状态。

> **说明：**
> - 您也可以使用 创建工作流接口 来创建一个工作流。
> - 转码任务的计费说明请参见 媒体处理费用 。

视频截帧

最近更新时间：2024-12-04 16:46:12

概览

视频截帧是指从视频中提取出一帧或多帧静态图像，在视频处理中，截帧可以用于视频预览、封面生成、缩略图制作等多种应用场景。本文将着重介绍基于 COS SDK 调用腾讯云数据万象（CI）视频截帧 API 实现图片截取功能，并提供多种截帧模式供开发者选择，助力用户快速应对实际应用场景需求。

使用场景

截取单张图片

提取视频中的特定的某一帧图片，可用于视频封面制作，视频文件预览和广告封面等场景。

截取多张图片

通过对截帧开始时间点、截帧间隔、截帧数量、输出图片尺寸、输出格式等进行自定义设置，提取视频中的一组图片，可用于视频采样分析、视频编辑、动图封面制作等场景。

雪碧图

将截取的多张图片按照一定的规律排列在一张图片上形成的一个大图。雪碧图可以通过一次请求加载多个图片，从而减少页面请求次数，提高网页性能。在视频播放器中，雪碧图可以用来实现视频预览功能，以便用户能够快速预览指定时间点的视频内容。雪碧图还可以用于游戏开发中的动画制作，将多个动画帧按照一定的规律排列在一张图片上，从而实现动画的展示。

执行方式

同步截帧

同步截帧可以获取媒体文件某个时间的截图，接口同步返回截图内容。此方式实现简单，能快速获取结果，但是仅支持截取单张图片。

异步截帧

异步截帧支持更多的参数配置，支持截取单张图片、多张图片和雪碧图。创建截帧任务后，异步处理视频文件，截图的结果在任务执行完成后通过消息回调或查询任务接口返回给用户。此方式支持更多的参数配置和处理方式，并且支持在 工作流 中使用，方便用户对新上传的文件和存量文件进行截帧处理。

前提条件

- 已创建和绑定 CI 存储桶，详情请参见 存储桶操作。
- 已 开通媒体处理 功能。
- 上传视频文件
- 初始化 COS Javascript SDK 并配置相关信息

```
<!--COS SDK-->
<script src="https://cdn.jsdelivr.net/npm/cos-js-sdk-v5/dist/cos-js-sdk-v5.min.js"></script>

// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '**********************************',
  SecretKey: '**********',
});

// 存储桶配置请在cos控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
const config = {
  // 需要替换成您自己的存储桶信息
```

```
  Bucket: '***-125********' /* 存储桶，必须 */,
  Region: '**-*****' /* 存储桶所在地域，必须字段 */,
  FileName: 'demo.mp4' /* 视频文件名 */,
};
```

同步截帧操作步骤

直接获取截图

```
// 同步截帧请求参数详见https://cloud.tencent.com/document/product/460/49283
cos.request(
  {
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: config.FileName /* 存储桶内的媒体文件，必须字段 */,
    Query: {
      'ci-process': 'snapshot' /** 固定值，必须 */,
      time: 1 /** 截图的时间点，单位为秒，必须 */,
      // width: 0, /** 截图的宽，非必须 */
      // height: 0, /** 截图的高，非必须 */
      // format: 'jpg', /** 截图的格式，支持 jpg 和 png，默认 jpg，非必须 */
      // rotate: 'auto', /** 图片旋转方式，默认为'auto'，非必须 */
      // mode: 'exactframe', /** 截帧方式，keyframe：截取指定时间点之前的最近的一个关键帧、exactframe：截取指定时
间点的帧，默认为 exactframe ，非必须 */
    },
    RawBody: true,
    // 可选返回文件格式为blob
    DataType: 'blob',
  },
  function (err, data) {
    console.log(err || data);
  }
);
```

异步截帧操作步骤

步骤一：配置截帧参数

- 视频截帧任务接口详见 请求参数 。
- 视频截帧模式配置详见 参数详情 。

不同的使用场景需要使用不同的参数配置：

场景 1：截取单张图片

```
const snapshot = {
  Mode: 'KeyFrame' /* 表示关键帧模式 */,
  Start: '5' /* 截图开始时间,单位为秒 */,
  Count: '1' /* 截图数量，必须 */,
};
```

场景 2：截取多张图片

支持不同的多帧截取方式，可以根据自己的使用场景来配置：

- **间隔模式**，间隔固定的时间截取多帧。

```
const snapshot = {
  Mode: 'Interval' /* 表示间隔模式  */,
  Start: '1' /* 截图开始时间,单位为秒 */,
  Count: '10' /* 截图数量，必须 */,
  TimeInterval: '5' /* 截图时间间隔，可选。未设置TimeInterval时表示截取所有帧共Count张图片 */,
};
```

- **平均模式**，按平均间隔截取共 X 张图片。

```
const snapshot = {
  Mode: 'Average' /* 表示平均模式 */,
  Start: '1' /* 截图开始时间,单位为秒 */,
  Count: '10' /* 截图数量，必须 */,
};
```

- **关键帧模式**，截取 X 张关键帧，与其他帧相比，关键帧包含更多的信息，画质更好。

```
const snapshot = {
  Mode: 'KeyFrame' /* 表示关键帧模式 */,
  Count: '10' /* 截图数量，必须 */,
};
```

场景 3：雪碧图

以下示例是截取25个关键帧，组合成一张 5 * 5 雪碧图：

```
const snapshot = {
  Mode: 'KeyFrame' /* 截图模式,interval 表示间隔模式 Average 表示平均模式 KeyFrame 表示关键帧模式 */,
  Start: '5' /* 开始时间,单位为秒 */,
  Count: '25' /* 截图数量，必须 */,
  SnapshotOutMode: 'OnlySprite' /* 截图输出模式参数, OnlySnapshot 表示仅输出截图模式 OnlySprite 表示仅输出雪碧图
模式 SnapshotAndSprite 表示输出截图与雪碧图模式*/,
  SpriteSnapshotConfig: {
    Color: 'Black' /* 背景颜色，必须,支持颜色详见 https://www.ffmpeg.org/ffmpeg-utils.html#color-syntax */,
    Columns: '5' /* 雪碧图列数，必须*/,
    Lines: '5' /* 雪碧图行数，必须 */,
  },
};
```

步骤二：创建视频截帧任务

视频截帧任务接口参数详见 请求参数 。

```
//需在地址前拼接/jobs，即：`https://<BucketName-APPID>.ci.<Region>.myqcloud.com/jobs
const host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
const url = 'https://' + host + '/jobs';
//使用cos sdk 发起视频截帧任务请求
const body = COS.util.json2xml({
  Request: {
    Tag: 'Snapshot' /* 创建任务的 Tag：Snapshot ,必须*/,
    Input: {
      Object: `${config.FileName}` /* 需要截帧的视频文件，存储桶里的路径 */,
    },
    Operation: {
      Snapshot: snapshot /* 使用 ‘步骤 3’ 的参数配置 ,必须 */,
```

```
      Output: {
        Bucket: config.Bucket /*      存储结果的存储桶 ，必须 */,
        Region: config.Region /*      存储结果存储桶地域 ，必须 */,
        Object: `result-${Number}.jpg` /* 结果文件的名称，${Number}为多张截图的序号，必须 */,
        SpriteObject: `sprite.jpg` /* 雪碧图的名字 ， 雪碧图模式下必须 */,
      },
    },
  },
});
cos.request(
  {
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'POST',
    Url: url,
    Key: '/jobs' /** 固定值，必须 */,
    ContentType: 'application/xml' /** 固定值，必须 */,
    Body: body,
  },
  (err, data) => {
    console.log(err || data);
  }
);
```

- 请求方式为 POST，Content-Type 为 application/xml，Tag 为 Snapshot 视频截帧，Input.Object 为 准备工作中上传的 视频文件,Operation.Output 为结果输出地址，需要注意截取多张图片时， Output.Object 文件名称中需要包含${Number}作为截图序号。
- 接口返回参数参考 响应参数，JobsDetail 节点下为任务接口响应信息。其中 JobId 为关键信息，步骤四中查询视频截帧任务结果时会用到。

步骤三：获取截帧文件

查询视频截帧任务 执行是否完成，获取截帧图片文件：

```
// 需在对象地址前面拼接 jobs/<jobId>，即：`https://<BucketName-APPID>.ci.<Region>.myqcloud.com/jobs/<jobId>`
// jobId 即为刚刚创建的任务 ID
const host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
const url = 'https://' + host + '/jobs/' + JobId;
cos.request(
  {
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Url: url,
    Key: '/jobs/' + JobId /** 固定值，必须 */,
    ContentType: 'application/xml' /** 固定值，必须 */,
  },
  (err, data) => {
    if (err) {
      // 视频截帧任务查询失败，请在console查看报错信息;
      console.log(JSON.stringify(err));
      return;
    }
    const resp = data.Response || {};
    //判断视频截帧任务是否执行中
    if (resp.JobsDetail.State !== 'Success') {
      console.log('...视频截帧任务执行中');
      return;
```

```
    }
    //任务执行完成 截图文件地址为
    const srtUrl = `https://${config.Bucket}.cos.${config.Region}.myqcloud.com/result-0.jpg`;
  }
);
```

- 返回体 Content-Type 为 application/xml，其中 State 为 Success 代表已经完成视频截帧，读取到通过视频截帧的截图文件地址为 `https://${config.Bucket}.cos.${config.Region}.myqcloud.com/result-${Number}.jpg`。
- 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问 。
- 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名 。

费用相关

- 视频截帧费用
- 对象存储相关费用

Demo 体验

具体代码可参考 cos demo，您也可使用腾讯云对象存储控制台，在 智能工具箱 栏目中体验视频截帧实际效果，也可以使用 TCPlayer 播放 COS 视频文件。

视频添加水印

最近更新时间：2025-04-03 14:19:02

概览

视频水印是指在视频中添加的一种标识，通常是一些文字、图片或者 Logo 等，用来标识视频的来源或者作者，以及保护视频的版权。数据万象支持将静态图片、文字等多种水印格式，支持同时为媒体资源添加多个水印，满足用户不同场景下的需求。本文将介绍如何基于 COS SDK 调用腾讯云数据万象（CI）视频水印 API 实现视频添加水印功能，并提供多种水印类型供开发者选择。

适用场景

版权保护

在视频文件中添加视频的来源、作者或 Logo 标识，减少视频被盗可能性。

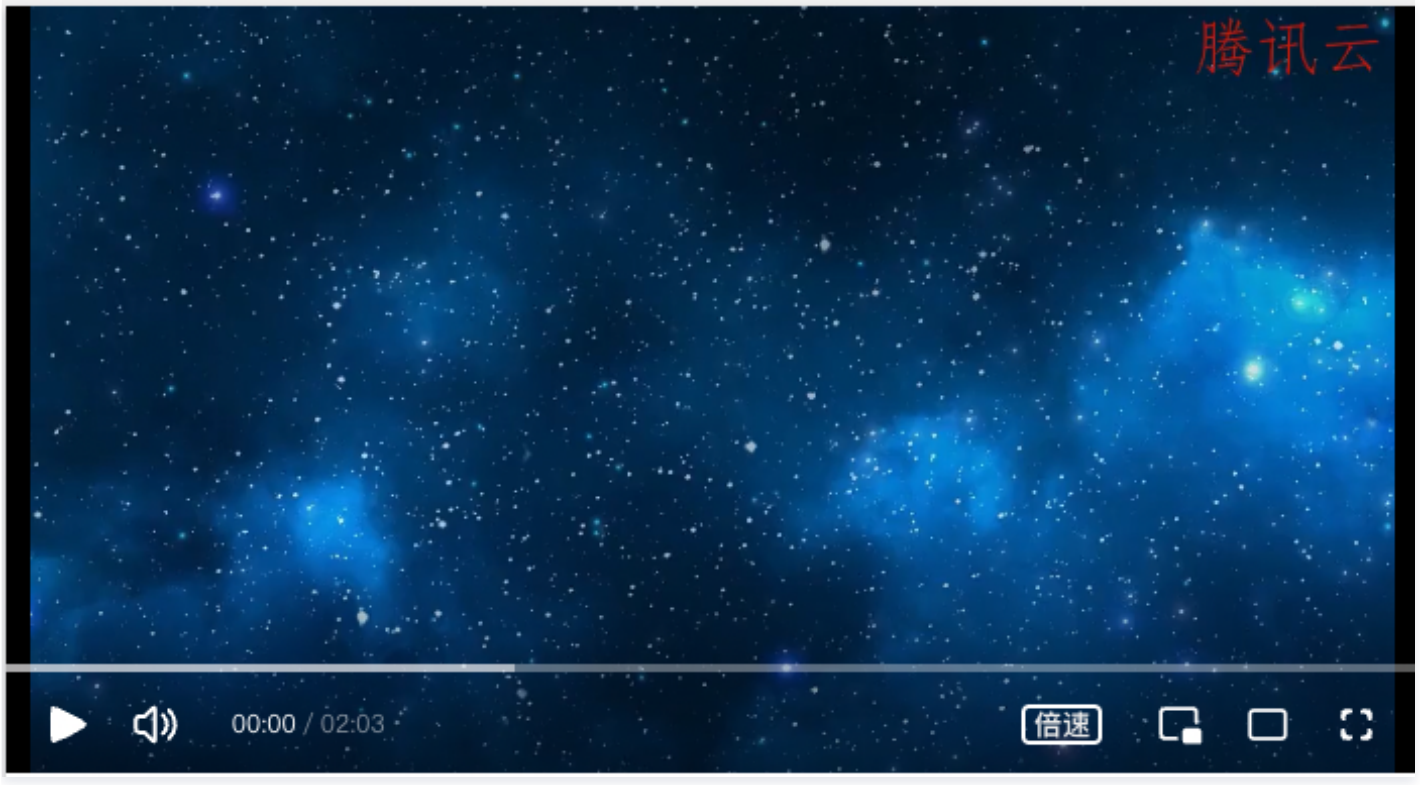
广告宣传

在视频文件中添加自己的 Logo 或品牌标识，以提高品牌知名度和宣传效果；也可以在视频文件中添加一些文字或图形标识，以增加视频的艺术效果和视觉冲击力。

支持的水印类型

文字水印

在视频中添加一个或多个文字标识，文字水印通常会出现在视频的角落或底部，支持设置文字水印的位置、字体、字号、颜色和透明度等属性。效果如下：



图片水印

在视频中添加一个或多个图片标识，多用于 logo 或商标展示等，支持设置图片水印的大小、位置、透明度、显示时长等属性，**支持使用 gif、apng 动图格式**，gif 和 apng 格式图片水印会在持续时间内循环播放。效果如下：

动态文字/图片水印

在视频中添加一个动态的图片或文字标识，动态水印随着视频的播放而移动，支持设置水印从左上角到右下角循环移动的横向速度和纵向速度。支持叠加设置三个动态的文字或图片水印。动态水印效果如下：

准备工作

- 已创建和绑定存储桶，详情请参见 存储桶操作。

- 已 开通媒体处理 功能。
- 上传视频文件。

视频添加水印操作步骤

以下介绍了如何使用 COS Javascript SDK 给视频同时添加三种不同类型的水印。

步骤 1：初始化 COS Javascript SDK 并配置相关信息

```
<!--COS SDK-->
<script src="https://cdn.jsdelivr.net/npm/cos-js-sdk-v5/dist/cos-js-sdk-v5.min.js"></script>

// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '**********************************',
  SecretKey: '**********************************',
});
// 存储桶配置请在 cos 控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
// 源文件相关配置
const InputConf = {
  Bucket: '***-125********',
  Region: '**-*****',
  FileName: 'demo.mp4',
  WaterMark: 'demo.png'    // 图片水印地址
};

// 水印结果文件相关配置，注意：需与源文件所在存储桶为同地域
const OutputConf = {
  Bucket: '***-125********',
  Region: '**-*****',
  FileName: 'demo.mp4',
};
```

步骤 2：创建视频水印任务

视频水印任务支持设置图片水印、文字水印和动态水印，详情参考 请求参数 。

图片水印参数：

```
const imageWaterMarkUrl = InputConf.Bucket + '.cos.' + InputConf.Region + '.myqcloud.com/' +
InputConf.WaterMark
const picWatermark = {
    "Type": "Image", // 水印类型 Image：图片水印
    "Dx": "10", // 水平偏移
    "Dy": "10", // 垂直偏移
    "LocMode": "Absolute", // 偏移方式 Relativity：按比例，Absolute：固定位置
    "Pos": "TopRight", // 基准位置 TopRight、TopLeft、BottomRight、BottomLeft、Left、Right、Top、Bottom、
Center
    "Image": { // 图片水印参数
        "Height": 10, // 高
        "Width": 10, // 宽
        "Mode": "Original", //尺寸模式 1. Original：原有尺寸,2. Proportion：按比例,3. Fixed：固定大小
        "Transparency": "75", // 透明度
        "Background": false, // 是否为背景图
        "Url": imageWaterMarkUrl // 水印图地址，支持 GIF,PNG,MOV,APNG,JPG 格式（gif 和 apng 格式图片水印会在持
续时间内循环播放）
    }
```

```
}
```

文字水印参数：

```
const txtWatermark = {
    "Type": "Text", // 水印类型 Text：文字水印
    "Dx": "10", // 水平偏移
    "Dy": "10", // 垂直偏移
    "LocMode": "Absolute", // 偏移方式 Relativity：按比例，Absolute：固定位置
    "Pos": "BottomRight", // 基准位置 TopRight、TopLeft、BottomRight、BottomLeft、Left、Right、Top、
Bottom、Center
    "Text": { // 文字水印参数
        "FontColor": "0xFF0000", // 字体颜色
        "FontSize": 32, // 字体大小
        "FontType": "simfang.ttf", // 字体类型
        "Text": "腾讯云", // 水印内容，长度不超过64个字符，仅支持中文、英文、数字、_、-和*
        "Transparency": "75" // 透明度
    },
}
```

动态文字水印参数：

```
const dynamicPicWatermark = {
    "Type": "Text", // 水印类型 Image：图片水印
    "SlideConfig": { // 水印滑动配置，配置该参数后水印位移设置不生效
        "SlideMode": "Default", // 滑动模式，Default：默认不开启、ScrollFromLeft：从左到右滚动
        "XSlideSpeed": "1", // 横向滑动速度
        "YSlideSpeed": "1" // 纵向滑动速度
    },
    "Text": { // 文字水印参数
        "FontColor": "0xFF0000", // 字体颜色
        "FontSize": 32, // 字体大小
        "FontType": "simfang.ttf", // 字体类型
        "Text": "腾讯云", // 水印内容
        "Transparency": "75" // 透明度
    },
}
```

构造提交视频水印任务接口并发起请求：

```
// 需在地址前拼接/jobs，即：`https://<BucketName-APPID>.ci.<Region>.myqcloud.com/jobs
const host = InputConf.Bucket + '.ci.' + InputConf.Region + '.myqcloud.com';
const url = 'https://' + host + '/jobs';

// 使用cos sdk 发起视频水印任务请求
// 水印参数，包括文字水印、图片水印、动画水印、滑动水印、图文混合水印添加
const body = COS.util.json2xml({
  Request: {
          "Tag": "Watermark", /* 创建任务的类型是 Watermark ,必须*/
          "Input": {
            "Object": InputConf.FileName, /* 需要的视频文件，存储桶里的路径，必须 */
          },
          "Operation": {
            "Output": {
              "Region": OutputConf.Region, /* 存储桶的地域，必须 */
              "Bucket": OutputConf.Bucket, /* 存储结果的存储桶，必须 */
              "Object": OutputConf.FileName /* 输出结果的文件名，必须 */
```

```
            },
            "Watermark": [picWatermark, txtWatermark, dynamicPicWatermark]
          }
        },
});

cos.request(
  {
    Method: 'POST',
    Url: 'https://' + host + '/jobs',
    Key: '/jobs', /** 固定值，必须 */
    ContentType: 'application/xml', /** 固定值，必须 */
    Body: body
  },
  (err, data) => {
    console.log(err || data);
  }
);
```

- 请求方式为 POST，Content-Type 为 application/xml，Tag 为 Watermark，Input.Object 为准备工作中上传的视频文件，Operation.Output 为结果输出地址可以填准备工作中创建的存储桶，需要注意 Output.Object 文件名称后缀名应为 **.mp4格式**，Operation.Watermark 为视频水印配置参数。
- **接口响应** 参数，JobsDetail 节点下为 **视频水印任务** 信息。其中 JobId 为关键信息，步骤3中 **构造查询视频水印任务链接** 会用到。

步骤 3：获取视频水印文件

查询视频水印任务 执行是否完成，并获取结果文件。

```
// 需在对象地址前面拼接 jobs/<jobId>，即：`https://<BucketName-APPID>.ci.<Region>.myqcloud.com/jobs/<jobId>`
// jobId 即为刚刚创建的任务 ID
const host = InputConf.Bucket + '.ci.' + InputConf.Region + '.myqcloud.com';
const url = 'https://' + host + '/jobs/' + JobId;
cos.request(
  {
    Method: 'GET',
    Url: url,
    Key: '/jobs/' + JobId, /** 固定值，必须 */
    ContentType: 'application/xml', /** 固定值，必须 */
  },
  (err, data) => {
    if (err) {
      // 视频水印任务查询失败，请在 console 查看报错信息
      console.log(JSON.stringify(err));
      return;
    }
    const resp = data.Response || {};
    // 判断视频水印任务是否执行中
    if (resp.JobsDetail.State !== 'Success') {
      console.log('...视频水印任务执行中');
      return;
    }
    //任务执行完成 水印文件地址为
    const srtUrl =
`https://${OutputConf.Bucket}.cos.${OutputConf.Region}.myqcloud.com/${OutputConf.FileName}`;
  }
);
```

返回体 Content-Type 为 application/xml，其中 State 为 Success 代表已经完成视频水印任务，读取到视频水印文件地址为 `https://${OutputConf.Bucket}.cos.${OutputConf.Region}.myqcloud.com/${OutputConf.Object}`。
添加三种水印之后，视频播放效果如下：

费用相关

- 视频添加水印将收取转码费用。
- 对象存储相关费用。

Demo 体验

具体代码可参考 cos demo，您也可使用腾讯云对象存储控制台，在 智能工具箱 栏目中体验视频水印实际效果。

视频自动添加字幕

最近更新时间：2024-11-15 21:50:12

概览

本文将介绍如何使用 COS SDK 调用数据万象的 语音识别 能力生成视频字幕文件，并使用 腾讯云播放器（TCPlayer） 播放挂载了字幕的视频。

业务场景

适用于短视频制作时，原始视频无字幕，需要自动识别视频语音内容并生成字幕的场景。可应用于 PGC/UGC 平台、视频网站、短视频应用、资讯平台等对媒体内容制作有较高智能化和时效性需求的行业。

准备工作

- 已创建和绑定存储桶，详情请参见 存储桶操作。
- 已 开通语音识别 功能。
- 上传视频文件
- 在页面中引入 COS SDK 与 TCPlayer 相关脚本文件：

```
<!--播放器样式文件-->
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.6.0/tcplayer.min.css"
rel="stylesheet" />
<!--播放器脚本文件-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.6.0/tcplayer.v4.6.0.min.js">
</script>
<!--COS SDK-->
<script src="https://cdn.jsdelivr.net/npm/cos-js-sdk-v5/dist/cos-js-sdk-v5.min.js"></script>
```

- 设置播放器容器节点:
  在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<video id="player-container" width="414" height="270" preload="auto" playsinline webkit-playsinline>
</video>
```

生成字幕文件

步骤 1：初始化 COS SDK 并配置相关信息

```
// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '************************************',
  SecretKey: '**********',
});
```

步骤 2：创建语音识别任务

构造提交语音识别任务接口并发起请求 请求参数：

```
// 存储桶配置请在 cos 控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
```

```
const config = {
  // 需要替换成您自己的存储桶信息
  Bucket: '***-125********' /* 存储桶，必须 */,
  Region: '**-*****' /* 存储桶所在地域，必须字段 */,
  FileName: 'demo.mp4' /* 文件名 */,
};
//需在地址前拼接/asr_jobs，即：`https://<BucketName-APPID>.ci.<Region>.myqcloud.com/
const host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
const url = 'https://' + host + '/asr_jobs';
//使用 cos sdk 发起语音识别任务请求
const body = COS.util.json2xml({
  Request: {
    Tag: 'SpeechRecognition' /* 创建任务的 Tag：SpeechRecognition ，必须*/,
    Input: {
      Object:config.FileName /* 需要语音识别的视频文件，存储桶里的路径 */,
    },
    Operation: {
      SpeechRecognition: {
        EngineModelType: '16k_zh_video' /* 引擎模型类型：16k 音视频领域 ，必须 */,
        ChannelNum: 1 /* 语音声道数，必须 */,
        ResTextFormat: 1 /* 识别结果返回形式，必须 */,
        OutputFileType: 'srt',
      },
      Output: {
        Bucket: config.Bucket /*      存储结果的存储桶 ，必须 */,
        Region: config.Region /*      存储结果存储桶地域 ，必须 */,
        Object: `demo.srt` /* 结果文件的名称 ，必须 */,
      },
    }
  },
});
cos.request(
  {
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'POST',
    Url: url,
    Key: '/asr_jobs' /** 固定值，必须 */,
    ContentType: 'application/xml' /** 固定值，必须 */,
    Body: body,
  },
  (err, data) => {
    console.log(err || data);
  }
);
```

- 请求方式为 POST，Content-Type 为 application/xml，Tag 为 SpeechRecognition 语音识别，Input.Object 为 准备工作中上传的 视频文件,Operation.Output 为结果输出地址可以填准备工作中创建的存储桶，需要注意 Output.Object 文件名称后缀名应为 **.srt** 格式，Operation.SpeechRecognition 为语音识别配置参数。
- 接口响应参数，JobsDetail 节点下为下面 **获取提语音识别任务接口接口响应** 任务信息。其中 JobId 为关键信息，下面 **构造查询语音识别任务链接** 会用到。

步骤 3：获取字幕文件

查询语音识别任务 执行是否完成，获取字幕文件。

```
// 存储桶配置请在 cos 控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
```

```
const config = {
  // 需要替换成您自己的存储桶信息
  Bucket: '***-125********' /* 存储桶，必须 */,
  Region: '**-*****' /* 存储桶所在地域，必须字段 */,
  FileName: 'demo.srt' /* 文件名 */,
};
// 需在对象地址前面拼接 ai_jobs/<jobId>，即：`https://<BucketName-APPID>.ci.
<Region>.myqcloud.com/ai_jobs/<jobId>`
// jobId 即为刚刚创建的任务 ID

const host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
const url = 'https://' + host + '/asr_jobs/' + JobId;
cos.request(
  {
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Url: url,
    Key: '/asr_jobs/' + JobId /** 固定值，必须 */,
    ContentType: 'application/xml' /** 固定值，必须 */,
  },
  (err, data) => {
    if (err) {
      // 语音识别任务查询失败，请在 console 查看报错信息;
      console.log(JSON.stringify(err));
      return;
    }
    const resp = data.Response || {};
    //判断语音识别任务是否执行中
    if (resp.JobsDetail.State !== 'Success') {
      console.log('...语音识别任务执行中');
      return;
    }
    //任务执行完成 字幕文件地址为
    const srtUrl = `https://${config.Bucket}.cos.${config.Region}.myqcloud.com/${config.FileName}`;
  }
);
```

- 返回体 Content-Type 为 application/xml，其中 State 为 Success 代表已经完成语音识别，读取到通过语音识别的字幕文件地址为 https://${Operation.Output.Bucket}.cos.${Operation.Output.Region}.myqcloud.com/${Operation.Output.Object}。

使用 TCPlayer 播放

1. 获取上面 **准备工作** 创建的视频文件地址：
   `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4`
2. 获取上面 **生成字幕文件** 创建的字幕文件地址：
   `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.srt`
3. 初始化播放器，并设置视频地址和字幕文件：

```
// TODO: 使用 Web 播放器时，为获取更佳的兼容性，可将普通 srt 格式字幕文件转换为 webvtt 格式
const getWebvvtUrl = url => {
  return fetch(url)
    .then(response => response.text())
    .then(text => {
      const arr = text.split('\n');
      text = arr.slice(0, arr.length - 4).join('\n');
      const vvtText = 'WEBVTT\n' + '\n' + text.replace(/,/g, '.') + '\n';
      console.log(vvtText);
```

```
        return URL.createObjectURL(new Blob([vvtText]));
      });
};
// srt 文件格式转 webvvt 格式，非必须，TCPlayer 播放器字幕文件暂只支持 webvvt 格式
const webvvtUrl = await getWebvvtUrl(`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.srt`);
// 初始化播放并设置播放地址及字幕文件
const Player = TCPlayer('player-container', {});
Player.src(`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4`);
Player.on('ready', function () {
  // 添加语音识别任务生成的字幕文件
  const subTrack = Player.addRemoteTextTrack(
    {
      src: webvvtUrl, // 字幕文件
      kind: 'subtitles',
      srclang: 'zh-cn',
      label: '中文',
      default: 'true',
    },
    true
  );
});
```

- 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问 。
- 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名 。

费用相关

- 语音识别费用
- 对象存储相关费用

Demo 体验

- 具体代码可参考cos-demo 视频自动添加字幕
- 使用 TCPlayer 播放 COS 视频文件。
- 通过数据万象 语音识别 并生成字幕文件能力，使用 COS 音视频播放器体验馆 体验给视频设置字幕实际效果。

视频混音实践

最近更新时间：2024-12-03 14:56:12

概览

视频混音是将多个音频轨道与视频轨道结合在一起，创建一个完整视听体验的过程。它可以用于合并不同的音频源，如背景音乐、对话、音效等，以增强视频的质量和吸引力。本文将介绍如何基于 COS Javascript SDK 调用腾讯云数据万象（CI）音视频转码 API 实现从视频 A 中抽离出其音频，并且混入到视频 B 中覆盖原视频音频，从而实现二次创作等需求。

适用场景

短视频二次创作

在视频文件中混入别的音频文件，利用搞怪的配音对视频进行恶搞创作，或将原音频替换为混音后的音频，来实现不同的后期效果。

前提条件

- 已创建和绑定 CI 存储桶，详情请参见 存储桶操作。
- 已 开通媒体处理 功能。
- 上传视频文件 到存储桶中。

操作步骤

步骤一：初始化 COS SDK 并配置相关信息

初始化 COS Javascript SDK 并配置相关信息：

```
<!--COS SDK-->
<script src="https://cdn.jsdelivr.net/npm/cos-js-sdk-v5/dist/cos-js-sdk-v5.min.js"></script>

// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '**********************************',
  SecretKey: '**********',
});

// 存储桶配置请在cos控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-shanghai'
const config = {
  // 需要替换成您自己的存储桶信息
  Bucket: "******125********", // 存储桶
  Region: "**-**********", // 存储桶所在地域
  FileName: "demo1.mp4", // 源文件
  MixFileName: "demo2.mp4", // 混音文件
  ResultName: "demo3.mp4", // 混音之后自定义文件名
};
```

说明：

- 注意：该初始化方式仅供联调测试使用，为了安全起见，请勿在生产环境直接暴露密钥。
- 生产环境请参考各语言 SDK 签名实现，详情请参见 SDK 签名实现 。

步骤二：提交音视频转码任务

提交音视频转码任务，参数详情参见 提交任务接口 。接口请求参数格式为：

```
/**
 * 支持使用cos桶文件，如果文件为私有读，需要使用cos.getObjectUrl方法得到一个带有签名信息的url
```

```
 * 如果为其他在线资源，可忽略此步骤
 */
const AkUrl = await cos.getObjectUrl({
  Bucket: config.Bucket,
  Region: config.Region,
  Key: config.MixFileName,
  Sign: true,
});

const key = `jobs`; // 固定值，必须
const host = `${config.Bucket}.ci.${config.Region}.myqcloud.com`;
const url = `https://${host}/${key}`;
const body = COS.util.json2xml({
  Request: {
    // 创建任务的Tag
    Tag: "Transcode",
    // 待操作的文件信息
    Input: {
      // 源文件路径
      Object: config.FileName,
    },
    // 操作规则
    Operation: {
      // 转码参数
      Transcode: {
        Container: {
          Format: "mp4",
        },
        Video: {
          Codec: "H.264",
        },
        Audio: {
          Codec: "aac",
        },
        // 混音参数
        AudioMix: {
          AudioSource: AkUrl, // 混音文件路径
          Replace: true, // 是否保留被混音视频的源音频
        },
      },
      // 结果输出配置
      Output: {
        // 存储桶的地域
        Region: config.Region,
        // 存储结果的存储桶
        Bucket: config.Bucket,
        // 输出结果的文件名
        Object: config.ResultName,
      },
    },
  },
});

const res = await cos.request({
  Method: "POST", // 固定值
  Key: key, // 固定值
  Url: url, // 请求的url
  Body: body, // 请求体参数
  ContentType: "application/xml", // 固定值
```

```
});
```

说明：

- 转码计费规则，请参见 音视频转码费用 。
- 通过子账号使用时，需要授予相关的权限，详情请参见 授权粒度详情 文档。
- 接口返回参数请参见 响应参数，JobsDetail 节点下为转码任务接口响应信息。其中 JobId 为关键信息，步骤三中查询翻译任务时会用到。

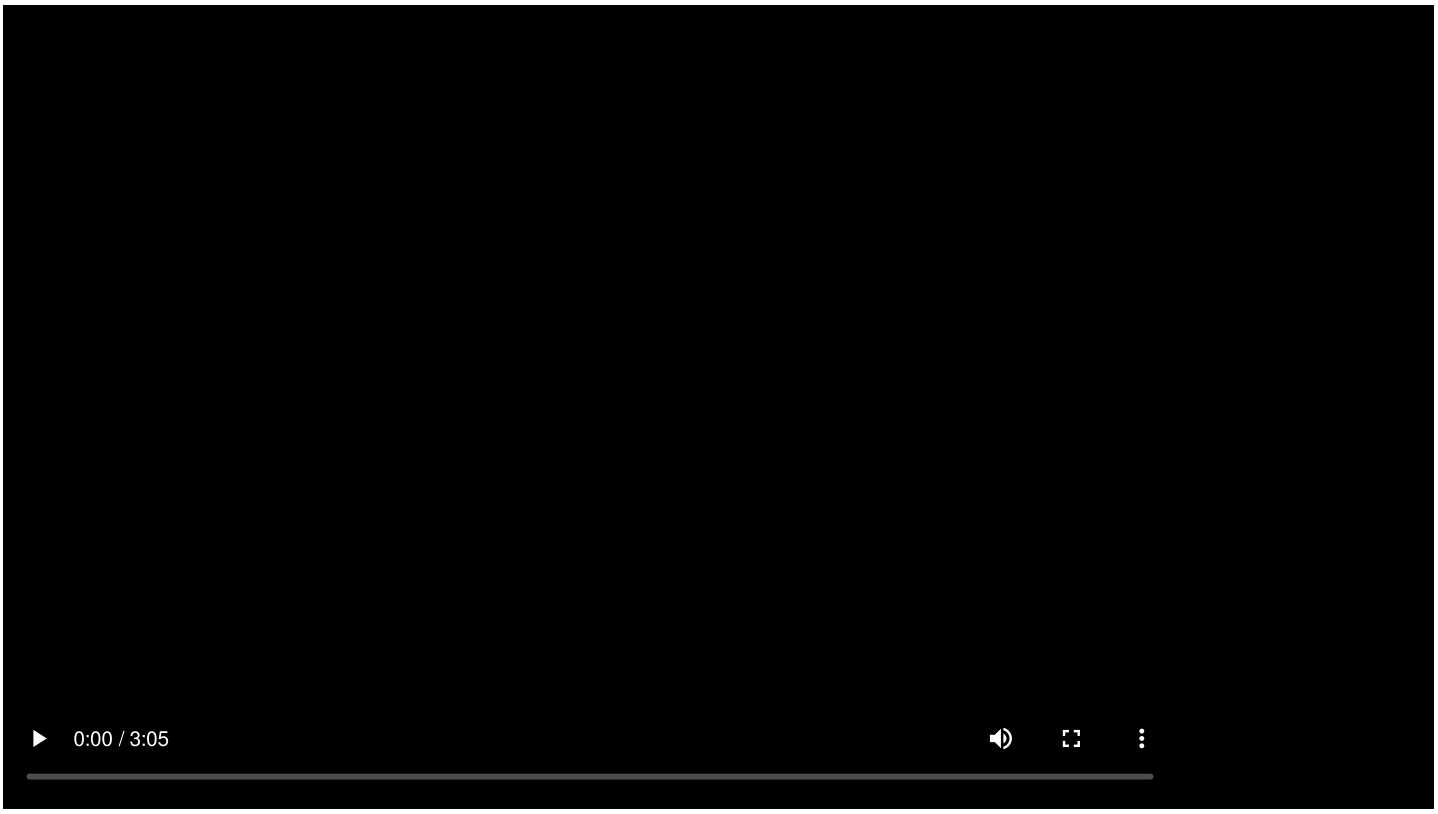
步骤三：查询音视频转码任务执行结果

步骤二中提交任务后返回的 JobId 作为任务唯一标识，通过 JobId 查询翻译任务，参数详情请参见 查询任务接口 。
可定时查询任务的状态，当返回的 State 为 Success 时代表文件转码成功，混音之后的视频文件地址为：
`https://${config.Bucket}.cos.${config.Region}.myqcloud.com/${config.ResultName}` 。
轮询任务执行结果的代码可参考：

```
// 轮询任务执行结果
function queryTranceTrack(jobId) {
  setTimeout(() => {
    const key = `jobs/${jobId}`; // jobId: 需要查询的jobId;
    const host = config.Bucket + ".ci." + config.Region + ".myqcloud.com";
    const url = `https://${host}/${key}`;
    cos.request(
      {
        Bucket: config.Bucket,
        Region: config.Region,
        Method: "GET",
        Url: url,
        Key: key /** 固定值，必须 */,
        ContentType: "application/xml" /** 固定值，必须 */,
      },
      async (err, data) => {
        if (err) {
          msgText.innerHTML = "任务查询失败，请在console查看报错信息";
          console.log(JSON.stringify(err));
          return;
        }
        const resp = data.Response || {};
        //判断任务是否在执行中
        if (resp.JobsDetail.State !== "Success") {
          msgText.innerHTML = "任务执行中...";
          queryTranceTrack(jobId);
          return;
        }
        // 任务执行完成 初始化播放器
        msgText.innerHTML = "任务完成";
      }
    );
  }, 2000);
}
```

说明：
推荐使用 API Explorer调试。

混音后的视频效果如下：



费用相关

- 对象存储相关费用
- 音视频转码费用

Demo 体验

具体代码可参考 cos demo 。

HLS 加密视频播放
HLS 视频加密播放实践

最近更新时间：2024-09-06 17:09:51

本文主要介绍数据万象 HLS 加密方案，包含视频加密和视频播放的过程。

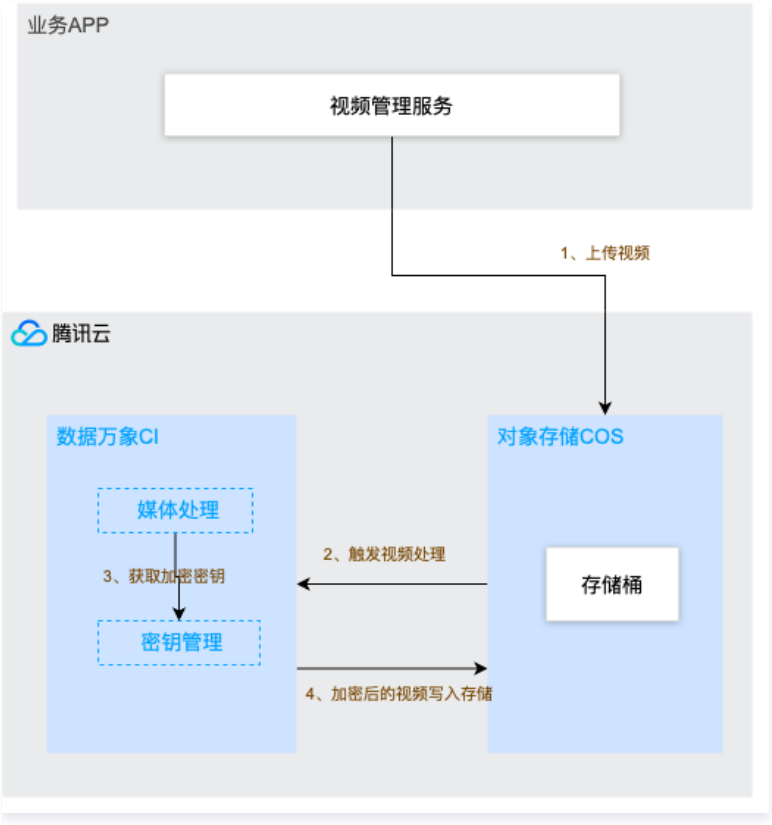
视频加密能力简介

视频加密是指对视频中的内容进行加密处理，加密后的视频无法分发给无访问权限的用户观看，即使视频被下载到本地，视频本身也是被加密的，无法正常播放和二次分发，从而保障您的视频版权不受到非法侵犯，有效防止视频泄露和盗链问题。视频加密可广泛用于在线教育及财经等领域。

方案架构

整体方案架构包含视频加密与播放加密后的视频两大部分流程说明。

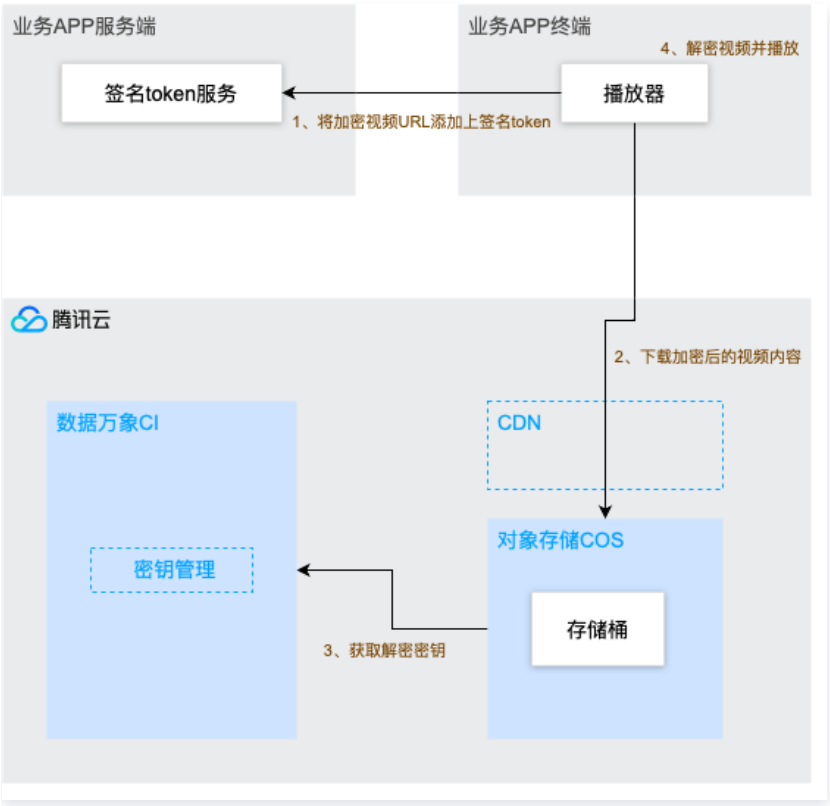
视频加密流程



视频加密

1. 上传视频：业务APP通过控制台、服务端 API 等方式，将视频上传到COS。
2. 触发视频处理：上传视频后，触发视频处理。触发后，视频在转码的过程中进行加密。
3. 获取加密密钥：视频转码并加密，从数据万象密钥管理模块获取加密密钥。
4. 加密后的视频写入存储：视频转码并加密后，输出的视频内容被写入到COS存储中。

播放加密后的视频



解密视频并播放

1. 将加密视频URL添加上签名token：向业务侧APP服务发起请求，在播放地址后拼接签名token。
2. 下载加密后的视频内容：接收业务侧播放器请求，如有CDN存在，CDN回源COS，下载加密后的视频内容。
3. 获取解密密钥：业务侧播放器携带含签名token的url从数据万象密钥管理模块请求解密密钥。
4. 解密并播放：播放器获取密钥后，解密视频并播放。

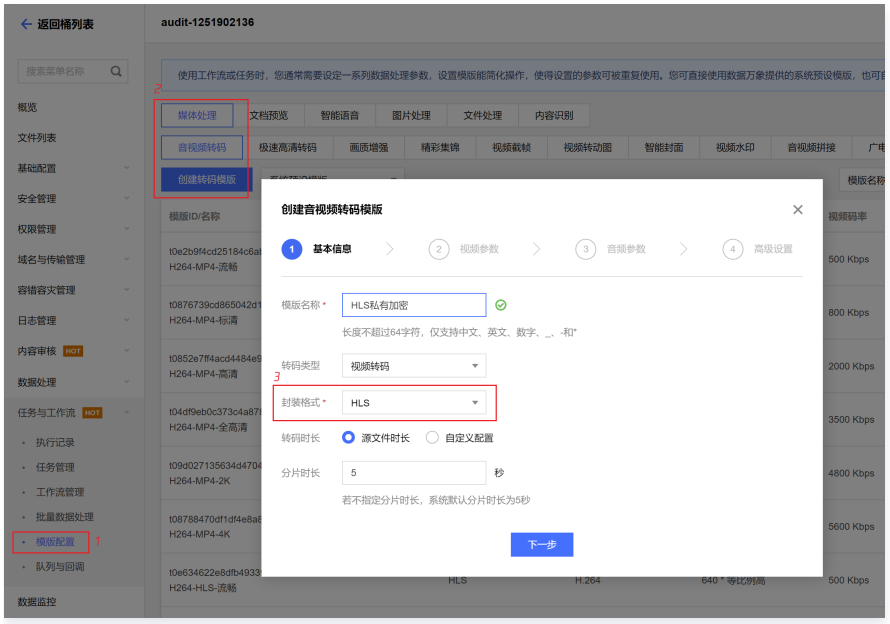
> **说明：**
> 业务方需开通腾讯云数据万象服务（CI）、存储服务（COS）。

接入指引

基于上述原理架构，进行如下操作步骤，便可以实现对视频的加密与解密播放。

步骤一：上传视频，转码为 HLS 加密视频

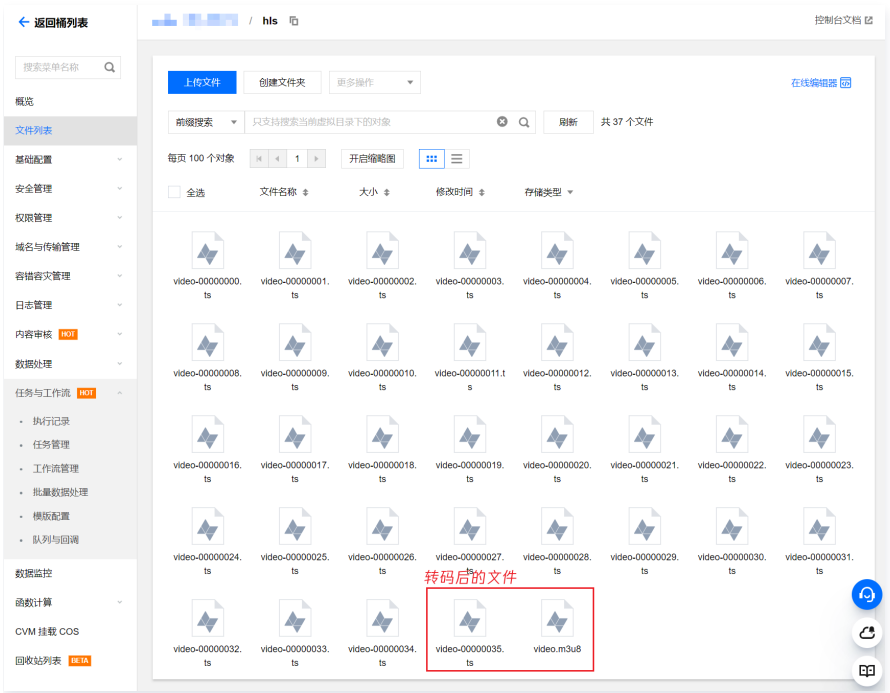
1. **上传视频**：在对象存储控制台 存储桶列表，进入要存储视频的存储桶，上传要转码的视频到存储桶。
2. **创建自定义模板**：左侧菜单找到**任务和工作流>模板配置**，选择音视频转码，单击创建转码模板，在弹窗里**封装格式**选择 HLS，并配置其他转码选项（可保留默认配置），该模板可用于后续创建任务和创建工作流。



在最后一步的高级设置里开启视频加密，点击完成。

3. **创建转码任务：到任务和工作流>任务管理点击创建任务，任务类型选择音视频转码，选择刚创建的HLS加密模板，选择目标存储桶和文件名后，点击确定。**

4. **找到转码后的视频文件**：等任务执行完成后，找到**创建任务**时填的目标路径，可看到生成后的.m3u8 和 .ts 后缀的**加密视频文件**。



5. 复制播放密钥：到 对象存储控制台，桶详情页面的**数据处理>媒体处理**，在上方选中**媒体处理**页签。编辑**开启媒体处理**，并生成和复制**播放密钥**。用于后续 token 服务搭建。

步骤二：搭建用于获取 token 的服务

HLS 加密视频播放流程里，需要开发者自行搭建服务，用于获取 HLS 播放 token 和签名。

token 计算规则

HLS 播放 token 采用 JWT（JSON Web Token），一种由 Header、PayLoad 和 Signature 组合得到的数字令牌。

token 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)

Payload 的参数说明

以上 Payload 的参数说明如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Type	无	token 类型，固定值为 CosCiToken	String	是
AppId	无	用户的 appId	String	是
BucketId	无	需要播放的文件所在的 BucketId	String	是
Object	无	需要播放的文件名	String	是
Issuer	无	token 颁发者，固定为 client	String	是
IssuedTimeStamp	无	token 颁发秒级时间戳	int	是
ExpireTimeStamp	无	token 过期秒级时间戳，默认1天过期	int	否
UsageLimit	无	token 使用次数限制，默认限制100次	int	是
ProtectContentKey	无	是否加密解密密钥（播放时解密ts视频流的密钥），1表示对解密密钥加密，0表示不对解密密钥加密。 默认为0	int	否
ProtectSchema	无	保护模式，仅支持 rsa1024 ，则表示使用 RSA 非对称加密的方式保护，公私钥对长度为	String	否（当 ProtectContentKey=1 时必选）

		1024bit		
PublicKey	无	公钥。1024 bit 的 RSA 公钥，需使用 Base64 进行编码	String	否（当 ProtectContentKey=1 时必选）

token 计算步骤

按照以上计算规则，这里提供一个计算过程的例子，假设某用户 需要播放 test-125000000 桶下的 hls_test.m3u8 加密视频，示例如下：

1. 计算 Header

Header 是 JSON 格式，表示 JWT 使用的算法信息，固定使用如下内容：

```
{
    // 加密的算法，固定为 HS256
    "alg": "HS256",
    // 类型，固定为 JWT
    "typ": "JWT"
}
```

2. 计算 Payload

PayLoad 是 JSON 格式，是播放器签名参数的内容，格式例如：

```
{
    // 固定为 CosCiToken，必填参数
    Type: "CosCiToken",
    // app id，必填参数
    AppId: "",
    // 播放文件所在的BucketId，必填参数
    BucketId: "",
    // 需要播放的文件名
    Object: "",
    // 固定为client，必填参数
    Issuer: "client",
    // token颁发时间戳，必填参数
    IssuedTimeStamp: Number,
    // token过期时间戳，非必填参数，默认1天过期
    ExpireTimeStamp: Number,
    // token使用次数限制，非必填参数，默认限制100次
    UsageLimit: 100,
    // 保护模式，填写为 rsa1024 ，则表示使用 RSA 非对称加密的方式保护，公私钥对长度为 1024 bit
    ProtectSchema: "rsa1024",
    // 公钥。1024 bit 的 RSA 公钥，需使用 Base64 进行编码
    PublicKey: "",
    // 是否加密解密密钥（播放时解密ts视频流的密钥），1表示对解密密钥加密，0表示不对解密密钥加密。
    ProtectContentKey: 0
}
```

示例：

```
{
  Type: "CosCiToken",
  AppId: "125000000",
  BucketId: "test-125000000",
  Object: "hls_test.m3u8",
  Issuer: "client",
  IssuedTimeStamp: 1697094000,
  ExpireTimeStamp: 1697122800,
```

```
  UsageLimit: 3,
  ProtectSchema: "rsa1024",
  PublicKey:
"NS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlHZk1BMEdDU3FHU0liM0RRRUJBUVVBQTRHTkFEQ0JpUUtCZ1FDa1M1TU1QS0k0eStudnU1V0E3amdGV0ZzYQoybXpqK0d1WWVjcVZreDJNaE1kbmZDcG0yL1JqNDdMbi9pYnFlSUJGNmxWZnZTbTdJTlBXNHNYWG9KMS9UK1lhClM0WHowTVBuSjZOemJHQTZSYm5FdTNlMkx4UnZXNTBPZXJaUnFwNlpOY21JR3prZHVjRXhhMDh4VmkzVHdGNGEKQjc4ZXFMdmlRSm5uTkFYWEl3SURBUUFCCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==",
  ProtectContentKey: 1
}
```


经过 base64UrlEncode 后的结果是：


```
eyJUeXBlIjoiQ29zQ2lUb2tlbiIsIkFwcElkIjoiMTI1MDAwMDAwIiwiQnVja2V0SWQiOiJ0ZXN0LTEyNTAwMDAwMCIsIk9iamVjdCI6Imhsc190ZXN0Lm0zdTgiLCJJc3N1ZXIiOiJjbGllbnQiLCJJc3N1ZWRUaW1lU3RhbXAiOjE2OTcwOTQwMDAsIk5vdEJlZm9yZVRpbWVTdGFtcCI6MCwiRXhwaXJlVGltZVN0YW1wIjoxNjk3MTIyODAwLCJSYW5kb20iOjAsIlVzYWdlTGltaXQiOjMsIlByb3RlY3RTY2hlbWEiOiJyc2ExMDI0IiwiUHVibGljS2V5IjoiTlMwdExTMUNSVWRKVGlCUVZVSk1TVU1nUzBWWkxTMHRMUzBLVFVsSFprMUJNRWREVTNGSFUwbGlNMFJSUlVKQlVWVkJRVFJIVGtGRVEwSnBVVXRDWjFGRGExTTFUVTFRUzBrMGVTdHVkblUxVjBFM2FtZEdWMFp6WVFveWJYcHFLMGQxV1dWamNWWnJlREpOYUUxa2JtWkRjRzB5TDFKcU5EZE1iaTlwWW5GbFNVSkdObXhXWm5aVGJUZEpUbEJYTkhOWVdOS1MxbENlM0WHowTVBuSjZOemJHQTZSYm5FdTNlMkx4UnZXNTBPZXJaUnFwNlpOY21JR3prZHVjRXhhMDh4VmkzVHdGNGEKQjc4ZXFMdmlRSm5uTkFYWEl3SURBUUFCCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==",
  "ProtectContentKey": 1
}
```


3. 计算 Signature

签名 Signature 是对 Header、Payload 内容，用播放密钥 playKey 算一个哈希值签名放在 token 里，用于保证参数不被篡改。计算签名用的 playKey 是存储桶的播放密钥，需要到 COS 控制台存储桶详情里的数据处理->媒体处理，获取播放密钥。

签名计算规则是：Signature = HMACSHA256(base64UrlEncode(Header) + "." + base64UrlEncode(Payload), playKey)

示例：以播放密钥作为 playKey（即 92552049b9a64ad3b3e45af066f387cd33）进行 HMAC 计算，Signature 是：4fRr2czM2vQx8lmhQ6zChN29oFmPrmu5X5oulo01dkc。

4. 合并得到 token

用英文字符 . 合并3个字符串，得到 token。
示例：


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJUeXBlIjoiQ29zQ2lUb2tlbiIsIkFwcElkIjoiMTI1MDAwMDAwIiwiQnVja2V0SWQiOiJ0ZXN0LTEyNTAwMDAwMCIsIk9iamVjdCI6Imhsc190ZXN0Lm0zdTgiLCJJc3N1ZXIiOiJjbGllbnQiLCJJc3N1ZWRUaW1lU3RhbXAiOjE2OTcwOTQwMDAsIk5vdEJlZm9yZVRpbWVTdGFtcCI6MCwiRXhwaXJlVGltZVN0YW1wIjoxNjk3MTIyODAwLCJSYW5kb20iOjAsIlVzYWdlTGltaXQiOjMsIlByb3RlY3RTY2hlbWEiOiJyc2ExMDI0IiwiUHVibGljS2V5IjoiTlMwdExTMUNSVWRKVGlCUVZVSk1TVU1nUzBWWkxTMHRMUzBLVFVsSFprMUJNRWREVTNGSFUwbGlNMFJSUlVKQlVWVkJRVFJIVGtGRVEwSnBVVXRDWjFGRGExTTFUVTFRUzBrMGVTdHVkblUxVjBFM2FtZEdWMFp6WVFveWJYcHFLMGQxV1dWamNWWnJlREpOYUUxa2JtWkRjRzB5TDFKcU5EZE1iaTlwWW5GbFNVSkdObXhXWm5aVGJUZEpUbEJYTkhOWVdOS1MxbENlM0WHowTVBuSjZOemJHQTZSYm5FdTNlMkx4UnZXNTBPZXJaUnFwNlpOY21JR3prZHVjRXhhMDh4VmkzVHdGNGEKQjc4ZXFMdmlRSm5uTkFYWEl3SURBUUFCCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==
```


token 服务代码实现

HLS 播放 token 是在 HLS 加密视频播放步骤里需要用于拼接播放链接的参数，采用 JWT（JSON Web Token）格式计算得出，是一种由 Header、PayLoad 和 Key 计算并组合得到的数字令牌。token 服务可以参考 Nodejs 示例、Go 示例。

步骤三：播放 HLS 加密视频

获取播放器签名 token 后，您可以分别使用 Web 端播放 HLS 加密视频、Android 端播放 HLS 加密视频、iOS 端播放 HLS 加密视频 和 小程序端播放 HLS 加密视频的播放器 Demo 进行验证，具体内容请参考 Demo 的源码。

费用说明

- 视频 HLS 加密将收取 转码费用

- 对象存储相关费用

Web 端播放 HLS 加密视频

最近更新时间：2025-04-29 16:35:21

本文主要介绍如何在 Web 端播放 HLS 私有加密视频。

前提条件

1. **创建加密视频、搭建 token 服务**，详情请参见 HLS 视频加密播实践 。
2. **配置 CORS 规则**：由于 Web 端播放会对播放地址 ajax 跨域请求，需要在 COS 控制台或 CDN 控制台配置 CORS 规则，允许播放页跨域访问资源。

Web 端播放指引

1. **下载播放代码库**：首先下载 JS 播放代码库，解压得到 cos_hls.js、jsencrypt.js 和 hls.js。
2. **引入到播放页面**：根据播放器种类，在自己页面中引入文件，目前支持三种类型（hls.js/tcplayer/video.js）。

   hls.js 引用示例：

```
<script src="./hls.js"></script>
<script src="./cos_hls_sdk.js"></script>
```

   tcplayer 引用示例：

```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v5.1.0/tcplayer.min.css"
rel="stylesheet">
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v5.1.0/libs/hls.min.1.1.7.js">
</script>
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v5.1.0/tcplayer.v5.1.0.min.js">
</script>
```

   video.js 引用示例：

```
<link href="https://vjs.zencdn.net/8.11.8/video-js.css" rel="stylesheet" />
<script src="https://vjs.zencdn.net/8.11.8/video.js"></script>
<script src="./cos_hls_sdk.js"></script>
```

3. **调用播放的方法**：前端使用 cos_hls.js 文件封装好的 cosHls 对象来播放加密的 m3u8视频文件，cosHls 对象的 play 方法参数说明如下：

参数名	说明	是否必填	类型	默认值
container	video 标签的默认值	是	String	无
playerType	播放器种类（hls.js/tcplayer/video.js）	是	String	无
playerOptions	播放器参数，在创建播放器是（hls.js/tcplayer/video.js）传入。	否	Object	无

   调用播放的代码示例：

```
<script>
// 请求服务端获取带有 token 和 签名 的 playUrl
const getPlayUrl = function (opt) {
    return new Promise(function (resolve, reject) {
        var xhr = new XMLHttpRequest();
        xhr.withCredentials = true;
        xhr.open('POST', `/hls/getPlayUrl`, true);
        xhr.setRequestHeader('Content-Type', 'application/json')
        xhr.onload = function () {
            var r = JSON.parse(xhr.responseText);
            resolve(r.playUrl);
        };
        xhr.onerror = function () {
            reject('get getPlayUrl error');
```

```
        };
        var data = {
            objectKey: opt.objectKey,
            publicKey: opt.publicKey,
            protectContentKey: opt.protectContentKey,
        };
        xhr.send(JSON.stringify(data));
    });
};
// 播放视频
const playVideo = async function () {
    const params = CosHlsSdk.createTokenParams();
    const opt = {
        // 这里根据业务需要，可以传自己业务逻辑里的视频 id，方便服务端知晓是播放哪个 objectKey
        objectKey: 'encrypt/bunny.m3u8',
        // publicKey、protectContentKey 是需要透传到服务端的参数
        publicKey: params.publicKey,
        protectContentKey: params.protectContentKey,
    };
    const playUrl = await getPlayUrl(opt);
    const { player } = CosHlsSdk.play({
        container: '#container',
        playerType: 'hls.js',
        playUrl: playUrl,
    });
    console.log(player);
};
playVideo();
</script>
```

如以上示例，hls/token 接口的参数 src 、publicKey、ProtectContentKey 不需要用户填写，SDK 会自动生成，只需要传递给 token 接口。

4. **完整的代码示例**：请参见 完整源码 。

> **注意：**
> 兼容性说明：当前示例在大部分浏览器场景下**默认是私有加密**，当检测到部分环境（iOS Safari 或 WebView）不兼容 Media Source Extensions 对象时，会自动降级为**标准加密**。

Demo 体验

您可在线体验 HLS 私有加密播放功能，单击前往 HLS 加密视频播放体验地址 。

Android 端播放 HLS 加密视频

最近更新时间：2025-05-29 16:01:42

本文主要介绍如何在 Android 端播放 HLS 私有加密视频。

前提条件

创建加密视频，搭建 token 服务，详情请参见 HLS 视频加密播实践 。

Android 端播放指引

Android 端使用 ci-assistor.aar 中封装好的 CIPlayerAssistor 和 CIMediaInfo 对象来播放 m3u8 文件，用户按照如下规则传入参数，即可实现播放功能。

1. 集成 CIPlayerAssistor SDK。
   - 集成 SDK：ci-assistor.aar
   - 依赖 nanohttpd

```
    implementation 'org.nanohttpd:nanohttpd:2.3.1'
```

2. 构造 CIMediaInfo 对象，参数说明如下。

参数名	说明	是否必填	类型	默认值
media	视频资源唯一标识，例如 hls/encrypt/test.m3u8	是	string	无
setPlayUrl	设置最终的播放 url	是	function	无

3. 使用 CIPlayerAssistor 获取最终的播放 url。

```
// 1. 初始化万象播放协助器，例如可以在 APP 启动时
CIPlayerAssistor.getInstance().init(context);

// 2. 获取最终的播放 url
String playerUrl = CIPlayerAssistor.getInstance().buildPlayerUrl(ciMediaInfo);

// 3. 注意：不再使用播放器助手后，请释放 CIPlayerAssistor 资源，例如 APP 退出时
CIPlayerAssistor.getInstance().destroy();
```

完整的 Android 端示例请参考如下代码。

```
// 初始化万象播放协助器，例如可以在 APP 启动时
CIPlayerAssistor.getInstance().init(context);

// 媒体资源标识，请替换成您业务的媒体资源标识，此处媒体资源标识仅为示例
String mediaObjectKey = "hls/encrypt/test.m3u8";

// CIMediaInfo 实例，可用于请求 PlayUrl
CIMediaInfo ciMediaInfo = new CIMediaInfo(mediaObjectKey);

// 从业务服务器获取 PlayUrl: 自行实现 getPlayUrl 方法(可参考下一块代码示例)
String playUrl = getPlayUrl(ciMediaInfo.getMedia(), ciMediaInfo.getPublicKey());

// 给 ciMediaInfo 设置获取到的 PlayUrl
ciMediaInfo.setPlayUrl(playUrl);

// 获取最终的播放 url（此链接只能播放一次，请不要重复使用，buildPlayerUrl 非耗时方法，可以多次调用）
```

```
String playerUrl = CIPlayerAssistor.getInstance().buildPlayerUrl(ciMediaInfo);

// 将 playerUrl 设置给播放器即可，demo 中演示了 exoplayer 和腾讯云播放器

// 以下以 exoplayer 作为代码示例
PlayerView playerView = view.findViewById(R.id.video_view);
ExoPlayer player = new ExoPlayer.Builder(getActivity()).build();
playerView.setPlayer(player);
MediaItem mediaItem = MediaItem.fromUri(playerUrl);
HlsMediaSource mediaSource = new HlsMediaSource.Factory(dataType ->
        new DefaultHttpDataSource.Factory().createDataSource()createMediaSource(mediaItem);
player.prepare(mediaSource);
player.play();

// 注意：不再使用播放器助手后，请释放 CIPlayerAssistor 资源，例如 APP 退出时
CIPlayerAssistor.getInstance().destroy();
```

从业务服务器获取 token 和签名的示例代码如下。

```
/**
 * 从业务服务器获取 PlayUrl
 */
private String getPlayUrl(String mediaObjectKey, String publicKey) {
    HttpURLConnection urlConnection = null;
    try {
        // 该 url 仅为示例，请替换成您业务的 url，具体实现请参考 “业务后端示例代码”
        URL url = new URL("https://cos.cloud.tencent.com/samples/hls/getPlayUrl");
        urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setRequestMethod("POST");
        urlConnection.setDoOutput(true);
        urlConnection.setRequestProperty("Content-Type", "application/json");

        JSONObject jsonObject = new JSONObject();
        // 添加键值对到 JSONObject
        jsonObject.put("objectKey", mediaObjectKey);
        jsonObject.put("protectContentKey", 1);
        jsonObject.put("publicKey", publicKey);

        // 将 JSONObject 转换为字符串
        String jsonInputString = jsonObject.toString();

        byte[] input = jsonInputString.getBytes("utf-8");
        urlConnection.getOutputStream().write(input, 0, input.length);

        int status = urlConnection.getResponseCode();
        if (status != 200) {
            throw new RuntimeException("HttpResponseCode: " + status);
        } else {
            BufferedReader br = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream(), "utf-8"));
            StringBuilder response = new StringBuilder();
            String responseLine;
            while ((responseLine = br.readLine()) != null) {
                response.append(responseLine.trim());
            }
            String responseString = response.toString();
            Log.d("getPlayUrl: ", responseString);
            JSONObject json = new JSONObject(responseString);
            String playUrl = json.getString("playUrl");
```

```
            // 在这里你可以使用 playUrl
            return playUrl;
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
    }
}
```

Demo 体验

我们提供了终端播放 HLS 私有加密视频的示例，请参见 CIPlayerAssistor Demo 。

iOS 端播放 HLS 加密视频

最近更新时间：2025-05-29 16:01:42

本文主要介绍如何在 iOS 端播放 HLS 私有加密视频。

前提条件

创建加密视频，搭建 token 服务，详情请参见 HLS 视频加密播实践 。

iOS 端播放指引

> ⚠ **注意：**
> 配置支持 HTTP 请求，需要在项目的 info.plist 文件中添加 `App Transport Security Settings -> Allow Arbitrary Loads` 并将其设置为 YES。

iOS 端使用 CIPlayerAssistor SDK 中封装好的 CIPlayerAssistor 和 CIMediaConfig 对象来播放 m3u8 文件，用户按照如下规则传入参数，即可实现播放功能。

1. 集成 CIPlayerAssistor SDK

   1.1 podfile 文件中新增如下内容并执行 `pod install` 。

```
pod 'CIPlayerAssistor'
```

   1.2 导入头文件。

```
#import <CIPlayerAssistor/CIPlayerAssistor.h>
```

2. 构造 CIPlayerAssistor 和 CIMediaConfig 对象

   CIMediaConfig 参数说明如下：

参数名	说明	是否必填	类型	默认值
fileUrl	请求m3u8接口的文件地址	是	NSString	无

   CIPlayerAssistor 参数说明如下：

参数名	说明	是否必填	类型	默认值
config	媒体文件配置信息	是	CIMediaConfig	无
url	调用业务服务返回的带签名的视频链接	是	NSString	无

iOS 端完整示例代码

```
@property (strong, nonatomic)AVPlayer *myPlayer;//播放器
@property (strong, nonatomic)AVPlayerItem *item;//播放单元
@property (strong, nonatomic)AVPlayerLayer *playerLayer;//播放界面（layer）

// 创建媒体配置对象
// fileUrl：文件链接。
CIMediaConfig * config = [[CIMediaConfig alloc]initWithFileUrl:@"https://ci-1251902136.cos.ap-chongqing.myqcloud.com/hls/encrypt/test.m3u8"];
[[CIPlayerAssistor singleAssistor] setDebug:NO];// 关闭日志打印
// CIMediaConfig 类在实例化时 自动生成了公钥 config.publicKey;
[self getToken:config.publicKey fileURL:config.fileUrl protectContentKey:1 callBack:^(NSString *_Nonnull url) {
```

```
        dispatch_async(dispatch_get_main_queue(), ^{
            // 设置授权信息，会在 url 上通过&拼接传入的 signature
            // 如果原始 url 是 cdn 的话，不用传 cos 的 signature
            [[CIPlayerAssistor singleAssistor] buildPlayerUrlWithConfig:self.config withUrl:url
buildUrlcallBack:^(NSString * _Nullable url, NSError * _Nullable error) {
                AVPlayerItem *item = [[AVPlayerItem alloc] initWithURL:[NSURL URLWithString:url]];
                self.myPlayer = [AVPlayer playerWithPlayerItem:item];
                self.playerLayer = [AVPlayerLayer playerLayerWithPlayer:self.myPlayer];
                self.playerLayer.frame = CGRectMake(0, 100, self.view.bounds.size.width, 300);
                [self.view.layer addSublayer:self.playerLayer];
                [self.myPlayer play];
            }];
        });
}];

- (void)getToken:(NSString *)publickey fileURL:(NSString *)fileURL protectContentKey:(int)protect
callBack:(void (^)(NSString * url))callBack{
    // 该 url 仅为示例，请替换成您业务的 url，具体实现请参考 “业务后端示例代码”
    NSMutableURLRequest * mrequest = [[NSMutableURLRequest alloc]initWithURL:[NSURL
URLWithString:@"https://cos.cloud.tencent.com/samples/hls/token"]];
    mrequest.HTTPMethod = @"POST";
    [mrequest setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    NSData *data = [publickey dataUsingEncoding:NSUTF8StringEncoding];
    publickey = [data base64EncodedStringWithOptions:0];
    NSString * objectKey = [NSURL URLWithString:fileURL].path;
    if ([objectKey hasPrefix:@"/"]) {
        objectKey = [objectKey substringFromIndex:1];
    }
    NSDictionary * body = @{
        @"objectKey":objectKey,
        @"publicKey":publickey,
        @"protectContentKey":@(protect)
    };
    mrequest.HTTPBody = [NSJSONSerialization dataWithJSONObject:body
options:NSJSONWritingFragmentsAllowed error:nil];
    [[[NSURLSession sharedSession]dataTaskWithRequest:mrequest completionHandler:^(NSData * _Nullable
data, NSURLResponse * _Nullable response, NSError * _Nullable error) {
        NSDictionary * result = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingAllowFragments error:nil];
        callBack(result[@"playUrl"]);
    }]resume];

}
```

Demo 体验

我们提供了终端播放 HLS 私有加密视频的示例，请参见 CIPlayerAssistor Demo 。

小程序端播放 HLS 加密视频

最近更新时间：2024-06-03 15:41:11

本文主要为您介绍如何在小程序端播放 HLS 加密视频。

前置步骤

先创建加密视频，搭建 token 服务，详情请参见 HLS 视频加密播实践 。

小程序端播放指引

由于小程序端的 video 标签是原生支持 HLS 格式，所以以下步骤主要是计算和拼接一个可以正常被 video 标签播放的地址。步骤如下：
1. 新建小程序工程代码，在 pages/index/index.wxml 里加 video 标签。

```
<video src="{{src}}"></video>
```

2. 在 pages/index/index.js 里填入以下代码示例。

```
Page({
    data: {
        // 将要播放的转码后的 m3u8 视频链接
        m3u8Url: 'https://example-1250000000.cos.ap-beijing.myqcloud.com/hls/video.m3u8?ci-process=pm3u8',
        src: '',
    },
    onReady: async function () {
        wx.request({
            method: 'POST',
            // 替换成自己实现的 hls token server
            // token server 参考文档：https://cloud.tencent.com/document/product/460/104024#5ea2a185-f626-4f4c-9011-cc7684c39baf
            // token server 代码示例：https://github.com/tencentyun/cos-demo/tree/main/server/hls-decrypt-token/
            url: 'https://example.com/samples/hls/token',
            data: JSON.stringify({
                src: this.data.m3u8Url,
                protectContentKey: 0,
            }),
            header: {
                'Content-Type': 'application/json',
            },
            dataType: 'json',
            success: (res) => {
                const {token, authorization} = res.data;
                // 拼接可被正常 HLS 标准加密方式播放的视频链接，加上签名、token等参数
                // 如果 m3u8 链接的域名是 CDN 域名，不需要拼 &${authorization} 签名
                const url = `${this.data.m3u8Url}?ci-process=pm3u8&expires=43200&tokenType=JwtToken&token=${encodeURIComponent(token)}&${authorization}`;
                this.setData({ src: url });
            },
            fail(res) {
                console.log(res);
            },
        });
    }
})
```

3. 在小程序开发工具上打开项目编译，即可正常播放 HLS 加密视频。

相关文档

HLS 视频加密

COS 音视频播放器实践
COS 音视频播放器概述

最近更新时间：2025-06-09 17:44:52

本文主要介绍 COS 音视频云端处理与端侧播放如何进行实际应用，文中的实践案例涵盖音视频处理所支持的协议、功能以及如何播放 COS 音视频文件的操作指引，并结合 腾讯云数据万象（CI） 丰富的音视频处理能力，为您提供更多的产品功能使用思路并获得更好的播放性能体验。

协议支持

音视频协议	URL 地址格式	PC 浏览器	移动端浏览器
MP3	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp3	支持	支持
MP4	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4	支持	支持
HLS (M3U8)	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8	支持	支持
FLV	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.flv	支持	支持
DASH	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mpd	支持	支持

注意：
HLS、FLV、DASH 视频在部分浏览器环境中播放需要依赖 Media Source Extensions 。

功能支持

功能	TCPlayer 播放器	DPlayer 播放器	Videojs 播放器
播放 MP4 格式视频	查看详情	查看详情	查看详情
播放 HLS 格式视频	查看详情	查看详情	查看详情
播放 FLV 格式视频	查看详情	查看详情	查看详情
播放 DASH 格式视频	查看详情	查看详情	查看详情
播放 PM3U8（私有 M3U8）视频	查看详情	查看详情	查看详情
设置封面图	查看详情	查看详情	查看详情
设置 HLS 标准加密	查看详情	查看详情	查看详情
切换清晰度	查看详情	查看详情	-
设置动态水印	查看详情	-	-
设置左上角 LOGO	-	查看详情	-
设置进度预览图	查看详情	-	-
设置字幕	查看详情	-	-
设置多语言	查看详情	-	-
设置贴片广告	查看详情	-	-

说明：
播放器兼容常见的浏览器，播放器内部会自动区分平台，并使用最优的播放方案。例如在 Chrome 等现代浏览器中优先使用 HTML5 技术实现视频播放，而手机浏览器上会使用 HTML5 技术或者浏览器内核能力实现视频播放。

使用指引

- 使用 TCPlayer 播放 COS 视频文件
- 使用 DPlayer 播放 COS 视频文件
- 使用 VideojsPlayer 播放 COS 视频文件

Demo 体验

您可在线体验 COS 音视频功能，单击前往 COS 音视频体验馆 。

COS 音视频播放器体验馆

最近更新时间：2023-06-19 11:25:34

Web 端体验

您可直接在电脑端浏览器访问 COS 音视频体验馆，体验 COS 的音视频处理功能。

移动端体验

示例代码

前往 Github 获取示例代码。

使用 TCPlayer 播放 COS 视频文件

最近更新时间：2023-06-19 11:25:34

简介

本文将介绍如何使用音视频终端 SDK（腾讯云视立方）集成的 TCPlayer 并结合 腾讯云数据万象(CI) 所提供的丰富的音视频能力，实现在 Web 浏览器播放 COS 视频文件。

集成指引

步骤1：在页面中引入播放器样式文件及脚本文件

```
<!--播放器样式文件-->
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/tcplayer.min.css"
rel="stylesheet">
<!--播放器脚本文件-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.0/tcplayer.v4.5.0.min.js">
</script>
```

说明

- 建议在正式使用播放器 SDK 时，自行部署以上相关静态资源，单击下载播放器资源 。
- 部署解压后的文件夹，不能调整文件夹里面的目录，避免资源互相引用异常。

步骤2：设置播放器容器节点

在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-playsinline>
</video>
```

说明

- 播放器容器必须为 `<video>` 标签。
- 示例中的 `player-container-id` 为播放器容器的 ID，可自行设置。
- 播放器容器区域的尺寸，建议通过 CSS 进行设置，通过 CSS 设置比属性设置更灵活，可以实现例如铺满全屏、容器自适应等效果。
- 示例中的 `preload` 属性规定是否在页面加载后载入视频，通常为了更快的播放视频，会设置为 `auto`，其他可选值：`meta`（当页面加载后只载入元数据），`none`（当页面加载后不载入视频），移动端由于系统限制不会自动加载视频。
- `playsinline` 和 `webkit-playsinline` 这几个属性是为了在标准移动端浏览器不劫持视频播放的情况下实现行内播放，此处仅作示例，请按需使用。
- 设置 `x5-playsinline` 属性在 TBS 内核会使用 X5 UI 的播放器。

步骤3：获取视频文件对象地址

1. 创建一个存储桶 。
2. 上传视频文件 。
3. 获取视频文件对象地址，格式为：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<视频格式>` 。

说明

- 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问 。
- 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名 。

步骤4：初始化播放器，并传入 COS 视频文件对象地址 URL

```
var player = TCPlayer("player-container-id", {}); // player-container-id 为播放器容器 ID，必须与 html 中一致
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COS 视频对象地址
```

功能指引

播放不同格式的视频文件

1. 获取 COS 存储桶上的视频文件对象地址。

> **说明**
> 未经转码的源视频在播放时有可能出现不兼容的情况，建议您使用转码后的视频进行播放，通过数据万象 音视频转码处理，获取不同格式视频文件。

2. 针对不同的视频格式，为了保证多浏览器的兼容性，需要引入对应的依赖。

- MP4：无需引入其他依赖。
- HLS：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.min.js 之前引入 hls.min.js。

```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.min.0.13.2m.js">
</script>
```

- FLV：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 FLV 格式的视频，需要在 tcplayer.min.js 之前引入 flv.min.js。

```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.min.1.6.2.js">
</script>
```

- DASH：DASH 视频需要加载 dash.all.min.js 文件。

```
<script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.all.min.js">
</script>
```

3. 初始化播放器并传入对象地址。

```
var player = TCPlayer("player-container-id", {}); // player-container-id 为播放器容器 ID，必须与 html 中一致
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COS 视频地址
```

获取示例代码：

- 播放 MP4 示例代码
- 播放 FLV 示例代码
- 播放 HLS 示例代码
- 播放 DASH 示例代码

播放 PM3U8 视频

PM3U8 是指私有的 M3U8 视频文件，COS 提供用于获取私有 M3U8 TS 资源的下载授权API，可参见 私有 M3U8 接口 。

```
var player = TCPlayer("player-container-id", {
    poster: "https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-process=pm3u8&expires=3600" // 私有 ts 资源 url 下载凭证的相对有效期为3600秒
});
```

获取示例代码：播放 PM3U8 示例代码

设置封面图

1. 获取 COS 存储桶上的封面图对象地址。

> ⚠ 注意
> 通过数据万象 智能封面 能力，提取最优帧生成截图作为封面，可提升内容吸引力。

2. 设置封面图。

```
var player = TCPlayer("player-container-id", {
 poster: "https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.png"
});
```

获取示例代码：设置封面图示例代码

播放 HLS 加密视频

为了保障视频内容安全，防止视频被非法下载和传播，数据万象提供了对 HLS 视频内容进行加密的功能，拥有相比于私有读文件更高的安全级别。加密后的视频，无法分发给无访问权限的用户观看。即使视频被下载到本地，视频本身也是被加密的，无法恶意二次分发，从而保障您的视频版权不受到非法侵犯。
操作步骤如下：

1. 参见 播放 HLS 加密视频 和 COS 音视频实践 | 给你的视频加把锁 流程，生成加密视频。
2. 初始化播放器并传入视频对象地址。

```
var player = TCPlayer("player-container-id", {}); // player-container-id 为播放器容器 ID，必须与 html 中
一致
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"); // hls 加密视频地址
```

获取示例代码：播放 HLS 加密视频示例代码

切换清晰度

数据万象 自适应码流 功能，可以将视频文件转码并打包生成自适应码流输出文件，帮助用户在不同网络情况下快速分发视频内容，播放器能够根据当前带宽，动态选择最合适的码率播放，详情可参见 COS 音视频实践 | 数据工作流助你播放多清晰度视频 。
操作步骤如下：

1. 通过 数据万象 自适应码流 功能，生成多码率自适应的 HLS 或 DASH 目标文件。
2. 初始化播放器并传入视频对象地址。

```
var player = TCPlayer("player-container-id", {}); // player-container-id 为播放器容器 ID，必须与 html 中
一致
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"); // 多码率视频地址
```

获取示例代码：切换清晰度示例代码

设置动态水印

播放器支持为视频添加位置与速度产生变换的水印。在使用动态水印功能时，播放器对象的引用不能暴露到全局环境，否则动态水印可以轻易去除，数据万象也支持在云端对视频进行添加动态水印等操作，详情可参见 水印模板接口 。

```
var player = TCPlayer("player-container-id", {
    plugins:{
      DynamicWatermark: {
        speed: 0.2, // 速度
        content: "腾讯云数据万象 CI", // 文案
        opacity: 0.7 // 透明度
      }
    }
  });
```

获取示例代码：设置动态水印

设置贴片广告

操作步骤如下：

1. 准备视频广告封面图以及广告链接。
2. 初始化播放器，设置广告封面图和链接，并设置广告节点。

```
var PosterImage = TCPlayer.getComponent('PosterImage');
PosterImage.prototype.handleClick = function () {
 window.open('https://cloud.tencent.com/product/ci'); // 设置广告链接
};

var player = TCPlayer('player-container-id', {
 poster: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx..png', // 广告封面图
});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');

var adTextNode = document.createElement('span');
adTextNode.className = 'ad-text-node';
adTextNode.innerHTML = '广告';

var adCloseIconNode = document.createElement('i');
adCloseIconNode.className = 'ad-close-icon-node';
adCloseIconNode.onclick = function (e) {
  e.stopPropagation();
  player.posterImage.hide();
};

player.posterImage.el_.appendChild(adTextNode);
player.posterImage.el_.appendChild(adCloseIconNode);
```

获取示例代码：设置贴片广告示例代码

设置视频进度图

操作步骤如下：

1. 通过数据万象 视频截帧 并生成雪碧图。
2. 获取步骤1生成的雪碧图和 VTT（雪碧图位置描述文件）对象地址。
3. 初始化播放器，并设置视频地址和 VTT 文件。

```
var player = TCPlayer('player-container-id', {
  plugins: {
 VttThumbnail: {
   vttUrl: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.vtt' // 进度图 VTT 文件
 },
  },
});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
```

获取示例代码：设置视频进度图示例代码

设置视频字幕

操作步骤如下：

1. 通过数据万象 语音识别 并生成字幕文件。
2. 获取步骤1生成的字幕 SRT 文件对象地址。
3. 初始化播放器，并设置视频地址和字幕 SRT 文件。

```
var player = TCPlayer('player-container-id', {});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
player.on('ready', function () {
  // 添加字幕文件
  var subTrack = player.addRemoteTextTrack({
 src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.srt', // 字幕文件
 kind: 'subtitles',
 srclang: 'zh-cn',
 label: '中文',
 default: 'true',
  }, true);
});
```

获取示例代码：设置视频字幕示例代码

设置视频多语言字幕

操作步骤如下：

1. 通过数据万象 语音识别 生成字幕文件，并同时翻译成多种语言。
2. 获取步骤1生成的多语言字幕 SRT 文件对象地址。
3. 初始化播放器，并设置视频地址和多语言字幕 SRT 文件。

```
var player = TCPlayer('player-container-id', {});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
player.on('ready', function () {
  // 设置中文字幕
  var subTrack = player.addRemoteTextTrack({
 src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/zh.srt', // 字幕文件
 kind: 'subtitles',
 srclang: 'zh-cn',
 label: '中文',
 default: 'true',
  }, true);
  // 设置英文字幕
  var subTrack = player.addRemoteTextTrack({
 src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/en.srt', // 字幕文件
 kind: 'subtitles',
 srclang: 'en',
 label: '英文',
 default: 'false',
  }, true);
});
```

获取示例代码：设置视频多语言字幕示例代码

Demo 体验

您可在线体验 COS 音视频功能，单击前往 COS 音视频体验馆 。

使用 DPlayer 播放 COS 视频文件

最近更新时间：2025-04-27 16:54:32

简介

本文将介绍如何使用 DPlayer 并结合 腾讯云数据万象（CI） 所提供的丰富的音视频能力，实现在 Web 浏览器播放 COS 视频文件。

集成指引

步骤1：在页面中引入播放器脚本文件以及按需引入部分依赖文件

```
<!-- 播放器脚本文件 -->
<script src="https://cdn.jsdelivr.net/npm/dplayer@1.26.0/dist/DPlayer.min.js"></script>
```

> **说明：**
> 建议在正式使用播放器时，自行部署以上相关静态资源。

步骤2：设置播放器容器节点

在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<div id="dplayer" style="width: 100%; height: 100%"></div>
```

步骤3：获取视频文件对象地址

1. 创建一个存储桶
2. 上传视频文件
3. 获取视频文件对象地址，格式为：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<视频格式>`。

> **说明：**
> - 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问。
> - 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名。

步骤4：初始化播放器，并传入 COS 视频文件对象地址 URL

```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
      url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4', // COS 视频对象地址
  },
});
```

功能指引

播放不同格式的视频文件

1. 获取 COS 存储桶上的视频文件对象地址。

> **说明：**
> 未经转码的源视频在播放时有可能出现不兼容的情况，建议您使用转码后的视频进行播放，通过数据万象 音视频转码处理，获取不同格式视频文件。

2. 针对不同的视频格式，为了保证多浏览器的兼容性，需要引入对应的依赖。
   - MP4：无需引入其他依赖。

- HLS：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5播放 HLS 格式的视频，需要在 tcplayer.min.js 之前引入 hls.min.js。

```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.min.0.13.2m.js"></script>
```

- FLV：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 FLV 格式的视频，需要在 tcplayer.min.js 之前引入 flv.min.js。

```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.min.1.6.2.js">
</script>
```

- DASH：DASH 视频需要加载 dash.all.min.js 文件。

```
<script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.all.min.js">
</script>
```

3. 初始化播放器并传入对象地址。

```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
   url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4', // COS 视频对象地址
  },
});
```

获取示例代码：

- 播放 MP4示例代码
- 播放 FLV 示例代码
- 播放 HLS 示例代码
- 播放 DASH 示例代码

播放 PM3U8视频

PM3U8是指私有的 M3U8视频文件，COS 提供用于获取私有 M3U8 TS 资源的下载授权 API，可参见 私有 M3U8接口 。

```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  // 关于 pm3u8 详情请查看相关文档：https://cloud.tencent.com/document/product/436/73189
  video: {
    url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-process=pm3u8&expires=3600'
// 私有 ts 资源 url 下载凭证的相对有效期为3600秒
  }
});
```

获取示例代码：播放 PM3U8示例代码

设置封面图

1. 获取 COS 存储桶上的封面图对象地址。

> ⚠ 注意：
> 通过数据万象 智能封面 能力，提取最优帧生成截图作为封面，可提升内容吸引力。

2. 初始化播放器并设置封面图。

```
const dp = new DPlayer({
```

```
  container: document.getElementById('dplayer'),
  video: {
 url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4',
 pic: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.png',
  },
});
```

获取示例代码：设置封面图示例代码

播放 HLS 加密视频

为了保障视频内容安全，防止视频被非法下载和传播，数据万象提供了对 HLS 视频内容进行加密的功能，拥有相比于私有读文件更高的安全级别。加密后的视频，无法分发给无访问权限的用户观看。即使视频被下载到本地，视频本身也是被加密的，无法恶意二次分发，从而保障您的视频版权不受到非法侵犯。
操作步骤如下：

1. 参见 播放 HLS 加密视频 和 COS 音视频实践 | 给您的视频加把锁 流程，生成加密视频。
2. 初始化播放器并传入视频对象地址。

```
 const dp = new DPlayer({
container: document.getElementById('dplayer'),
video: {
  url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8' // 加密视频地址
}
 });
```

获取示例代码：播放 HLS 加密视频示例代码

切换多清晰度

数据万象 自适应码流 功能，可以将视频文件转码并打包生成自适应码流输出文件，帮助用户在不同网络情况下快速分发视频内容，播放器能够根据当前带宽，动态选择最合适的码率播放，详情可参见 COS 音视频实践 | 数据工作流助您播放多清晰度视频 。
操作步骤如下：

1. 通过 数据万象 自适应码流 功能，生成多码率自适应的 HLS 或 DASH 目标文件。
2. 初始化播放器并传入视频对象地址。

```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
   url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8', //  多码率的HLS/DASH视频
  },
});
```

获取示例代码：切换清晰度示例代码

设置左上角 LOGO

播放器支持在左上角设置 LOGO。
操作步骤如下：

1. 获取 COS 存储桶上的 LOGO 图标对象地址。
2. 初始化播放器并设置 LOGO 图标。

```
const dp = new DPlayer({
 container: document.getElementById('dplayer'),
  video: {
   url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4',
  },
 logo: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.svg'
});
```

获取示例代码：设置左上角 LOGO 示例代码

Demo 体验

您可在线体验 COS 音视频功能，单击前往 COS 音视频体验馆 。

使用 VideojsPlayer 播放 COS 视频文件

最近更新时间：2023-06-19 11:25:35

简介

本文将介绍如何使用 VideojsPlayer 并结合 腾讯云数据万象(CI) 所提供的丰富的音视频能力，实现在 Web 浏览器播放 COS 视频文件。

集成指引

步骤1：在页面中引入播放器样式文件及脚本文件

```
<!-- 播放器样式文件 -->
<link href="https://vjs.zencdn.net/7.19.2/video-js.css" rel="stylesheet" />
<!-- 播放器脚本文件 -->
<script src="https://vjs.zencdn.net/7.19.2/video.min.js"></script>
```

> **说明**
> 建议在正式使用播放器时，自行部署以上相关静态资源。

步骤2：设置播放器容器节点

在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<video
  id="my-video"
  class="video-js"
  controls
  preload="auto"
  width="100%"
  height="100%"
  data-setup="{}"
></video>
```

步骤3：获取视频文件对象地址

1. 创建一个存储桶。
2. 上传视频文件。
3. 获取视频文件对象地址，格式为 `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<视频格式>`。

> **说明**
> - 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问。
> - 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名。

步骤4：在播放器容器内设置视频地址，传入 COS 视频文件对象地址 URL

```
<video
  id="my-video"
  class="video-js"
  controls
  preload="auto"
  width="100%"
  height="100%"
  data-setup="{}"
>
  <source
```

```
    src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
    type="video/mp4"
  />
</video>
```

功能指引

播放不同格式的视频文件

1. 获取 COS 存储桶上的视频文件对象地址。

> **说明**
> 未经转码的源视频在播放时有可能出现不兼容的情况，建议您使用转码后的视频进行播放，通过数据万象 音视频转码处理，获取不同格式视频文件。

2. 针对不同的视频格式，为了保证多浏览器的兼容性，需要引入对应的依赖。

- MP4：无需引入其他依赖。
- HLS：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.min.js 之前引入 hls.min.js。

```
  <script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.min.0.13.2m.js">
</script>
```

- FLV：如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 FLV 格式的视频，需要在 tcplayer.min.js 之前引入 flv.min.js。

```
  <script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.min.1.6.2.js">
</script>
```

- DASH：DASH 视频需要加载 dash.all.min.js 文件。

```
  <script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.all.min.js">
</script>
```

3. 初始化播放器并传入对象地址。

```
<!-- MP4 -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
  type="video/mp4"
/>

<!-- HLS -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"
  type="application/x-mpegURL"
/>

<!-- FLV -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.flv"
  type="video/x-flv"
/>

<!-- DASH -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mpd"
  type="application/dash+xml"
```

```
/>
```

获取示例代码：

- 播放 MP4 示例代码
- 播放 FLV 示例代码
- 播放 HLS 示例代码
- 播放 DASH 示例代码

播放 PM3U8 视频

PM3U8 是指私有的 M3U8 视频文件，COS 提供用于获取私有 M3U8 TS 资源的下载授权API，可参见 私有 M3U8 接口 。

```
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-process=pm3u8&expires=3600"
  type="application/x-mpegURL"
/>
```

获取示例代码：播放 PM3U8 示例代码

设置封面图

1. 获取 COS 存储桶上的封面图对象地址。

> ⚠ 注意
> 通过数据万象 智能封面 能力，提取最优帧生成截图作为封面，可提升内容吸引力。

2. 初始化播放器并设置封面图。

```
<video
  id="my-video"
  class="video-js"
  controls
  preload="auto"
  width="100%"
  height="100%"
  data-setup="{}"
  poster="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/poster.png"
>
  <source
 src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
 type="video/mp4"
  />
</video>
```

获取示例代码：设置封面图示例代码

播放 HLS 加密视频

为了保障视频内容安全，防止视频被非法下载和传播，数据万象提供了对 HLS 视频内容进行加密的功能，拥有相比于私有读文件更高的安全级别。加密后的视频，无法分发给无访问权限的用户观看。即使视频被下载到本地，视频本身也是被加密的，无法恶意二次分发，从而保障您的视频版权不受到非法侵犯。
操作步骤如下：

1. 参见 播放 HLS 加密视频 和 COS 音视频实践 | 给您的视频加把锁 流程，生成加密视频。
2. 初始化播放器并传入视频对象地址。

```
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"
  type="application/x-mpegURL"
```

```
/>
```

获取示例代码：播放 HLS 加密视频示例代码

Demo 体验

您可在线体验 COS 音视频功能，单击前往 COS 音视频体验馆 。

智能语音实践
图文点读

最近更新时间：2024-12-03 14:31:12

概述

在处理一些文本类信息时，例如图片中的英文，我们可以使用 OCR 识别技术来快速识别，并结合语音合成技术来朗读照片中的英文。本文将介绍如何使用 腾讯云数据万象（CI）的 OCR 技术识别图片中的文本内容，并通过语音合成技术将文本转换成语音播放。

应用场景

语言学习

将语言教材和词典中的英文转化为语音播放，可以为学习者提供标准的发音示范，帮助他们更好地学习语音。这种技术适用于语音学习软件、外语学习应用等场景。

书籍阅读

借助图文点读技术，儿童绘本、漫画、地图导览等可迅速将图片中的内容转化为语音，从而提升信息获取的趣味性和互动性，让用户更便捷地阅读和获取信息。

准备工作

- 创建数据万象存储桶，详情请参见 存储桶操作 。
- 在存储桶详情 – 智能语音页面，开启智能语音功能。

- 上传待处理的图片 到存储桶中。

操作步骤

步骤一：初始化 COS SDK 并配置相关信息

```
// 密钥请在访问管理控制台获取。https://console.cloud.tencent.com/cam/capi
const cos = new COS({
  SecretId: '**********************************',
  SecretKey: '**********',
});

// 存储桶配置请在cos控制台获取。https://console.cloud.tencent.com/cos/bucket
// 格式参考：Bucket: 'abc-1250000000', Region: 'ap-guangzhou'
const bucketConf = {
    Bucket: 'test-1250000000',
    Region: 'ap-guangzhou'
};
```

> **说明：**
> - 注意：该初始化方式仅供联调测试使用，为了安全起见，请勿在生产环境直接暴露密钥。
> - 生产环境请参考各语言 SDK 签名实现，详情请参见 SDK 签名实现 。

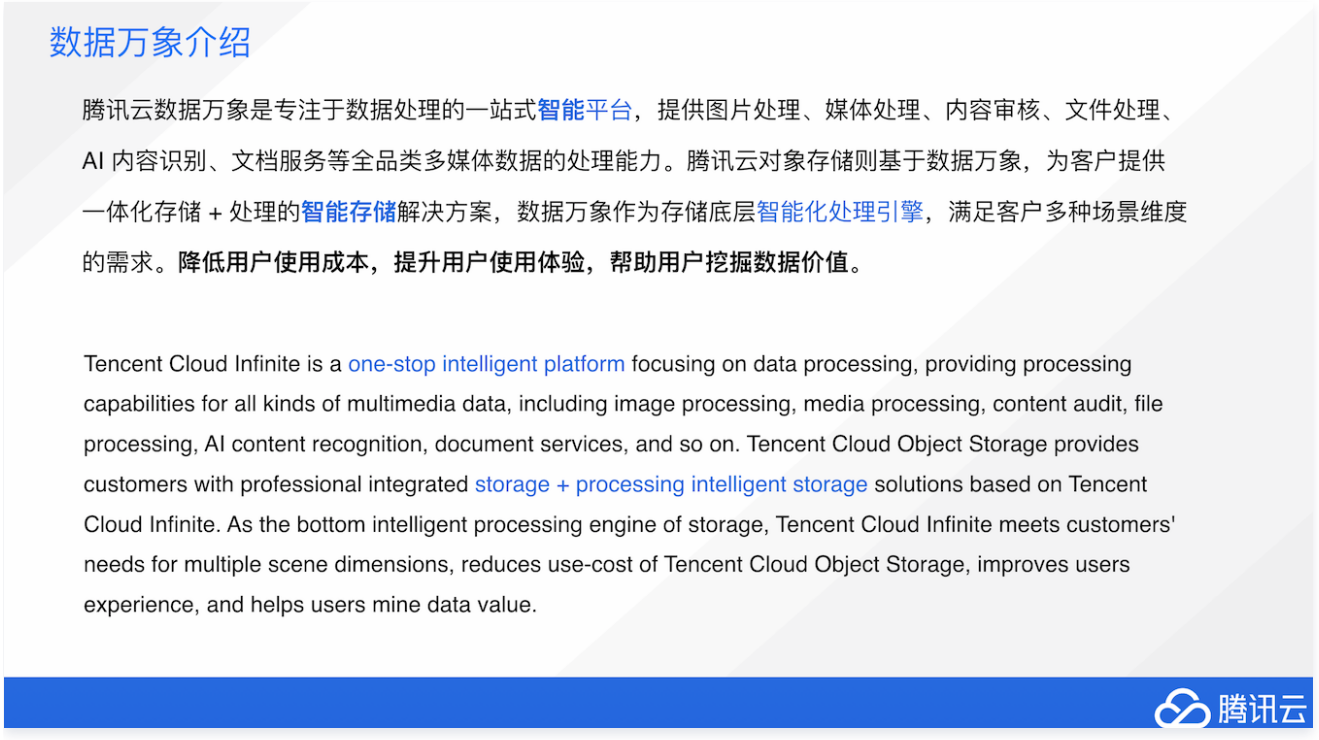
步骤二：使用 OCR 技术识别图片内容

数据万象通用文字识别功能（OCR）基于行业前沿的深度学习技术，将图片上的文字内容，智能识别为可编辑的文本。通过子账号使用时，需要授予相关的权限，详情请参见 授权粒度详情 文档。对于存储在 test-1250000000 存储桶中的图片 test.png，OCR 识别如下：

```
// 使用 OCR 功能提取图片中的文字，接口可参考 https://cloud.tencent.com/document/product/460/63227
cos.request({
    ...bucketConf,
    Key: 'test.png',     // 待识别的图片
    Query: {
        'ci-process': 'OCR',
    }
}, (ocrErr, ocrData) => {
    if (ocrErr) {
        console.log(JSON.stringify(ocrErr));
        return alert('识别图片失败');
    }
    const list = ocrData?.Response?.TextDetections?.map(text => text.DetectedText || "");
    const ocrResult = list?.join('\n') || "";
    console.log(ocrResult);
})
```

> **说明：**
> OCR计费规则参见 内容识别费用 - 通用文字识别 。

识别效果如下：



选择文件 tts-demo.png

识别的内容： 数据万象介绍 腾讯云数据万象是专注于数据处理的一站式智能平台，提供图片处理、媒体处理、内容审核、文件处理、 AI内容识别、文档服务等全品类多媒体数据的处理能力。腾讯云对象存储则基于数据万象，为客户提供 一体化存储+处理的智能存储解决方案，数据万象作为存储底层智能化处理引擎，满足客户多种场景维度 的需求。降低用户使用成本，提升用户使用体验，帮助用户挖掘数据价值。 Tencent Cloud Infinite is a one-stop intelligent platform focusing on data processing, providing processing capabilities for all kinds of multimedia data, including image processing, media processing, content audit, file processing, AI content recognition, document services, and so on. Tencent Cloud Object Storage provides customers with professional integrated storage + processing intelligent storage solutions based on Tencent Cloud Infinite. As the bottom intelligent processing engine of storage, Tencent Cloud Infinite meets customers' needs for multiple scene dimensions, reduces use-cost of Tencent Cloud Object Storage, improves users experience, and helps users mine data value. 腾讯云

步骤三：创建语音合成任务

数据万象语音合成技术通过先进的深度学习技术，将文本转换成自然流畅的语音。支持多种音色，并提供调节语速、语调、音量等功能。

- 如果识别的文本内容 `ocrResult` 长度不超过300字符，可以直接创建语音合成任务，同步获取合成的音频。
- 如果识别的文本内容 `ocrResult` 长度超过300字符，需要先将识别的内容上传到 COS 中，再创建语音合成任务。

通过子账号使用时，需要授予相关的权限，详情请参见 授权粒度详情 文档。
创建语音合成任务，参数详情参见 提交任务接口 。

```
// 上传ocrResult到cos存储桶，参考接口： https://cloud.tencent.com/document/product/436/7749
// 注意：为了避免上传出现跨域错误，需要设置跨域访问规则，简单调试可将来源Origin设置为*，参考
https://cloud.tencent.com/document/product/436/13318
cos.putObject({
    ...bucketConf,
    Key: key,    // 目标文件名
    Body: file,
}, (err, data) => {
    if (err) {
        return console.log(JSON.stringify(OcrUploadErr));
    }
    const txtUrl = `https://${bucketConf.Bucket}.cos.${bucketConf.Region}.myqcloud.com/content.txt`;

    // 创建语音合成任务，参考接口：https://cloud.tencent.com/document/product/460/84797
    cos.request({
            ...bucketConf,
            Method: 'POST',
            Url:  `https://${bucketConf.Bucket}.ci.${bucketConf.Region}.myqcloud.com/jobs`,
            Key: '/jobs', /** 固定值，必须 */
            ContentType: 'application/xml', /** 固定值，必须 */
            Body: COS.util.json2xml({
                Request: {
                    Tag: 'Tts',      // 语音合成任务
                    Operation: {
                        TtsConfig: {
                            InputType: 'Url',        // 支持输入文本文件 Url 或 utf8-字符，utf8-字符长度不超过
300，Url文件大小不超过10000个字符
                            Input: txtUrl         // 将步骤2中识别到的文本作为输入
```

```
                        },
                        TtsTpl: {
                            Mode: "Asyc",            // 支持同步和异步模式，如果文本内容长，建议选择异步模式
                            Codec: "mp3",            // 生成mp3音频，支持wav、mp3和pcm格式
                            VoiceType: "ruxue"       // 支持多种音色选择
                        },
                        Output: {
                            Bucket: bucketConf.Bucket,
                            Region: bucketConf.Region,
                            Object: 'result.mp3'      // 输出音频的地址
                        }
                    }
                }
            })
        }, (err, data) => {
            if (err || !data?.Response?.JobsDetail?.JobId) {
                return console.log(JSON.stringify(err));
            }

            // 如果任务提交成功，记录当前的任务id
            const jobId = data?.Response?.JobsDetail?.JobId || "";
        })

})
```

说明：

- 语音合成计费规则参见 **智能语音费用 – 语音合成** 。
- 接口返回参数参考 **响应参数**，JobsDetail 节点下为任务接口响应信息。其中 JobId 为关键信息，步骤四中查询语音合成任务结果时会用到。

步骤四：获取合成的音频

通过 JobId 查询任务执行结果，详情请参见 **查询任务接口** 。
可定时查询任务的状态，当返回的 State 为 Success 时代表音频合成了，音频地址为：
`https://${Operation.Output.Bucket}.cos.${Operation.Output.Region}.myqcloud.com/${Operation.Output.Object}` 。

```
let TaskInterval = null;

/**
* 轮询查询任务的执行情况，参考接口： https://cloud.tencent.com/document/product/460/84765
* 也可以通过回调的方式获取任务执行结果：https://cloud.tencent.com/document/product/460/84958
**/
function queryTask(jobId, callback) {
    TaskInterval && clearInterval(TaskInterval);
    TaskInterval = setInterval(() => {
        cos.request({
            ...bucketConf,
            Method: 'GET',
            Url: `https://${bucketConf.Bucket}.ci.${bucketConf.Region}.myqcloud.com/jobs/${jobId}`,
            Key: `/jobs/${jobId}`
        }, (err, data) => {
            if(err) {
                console.log(JSON.stringify(err));
                return callback && callback(err);
            }
            if (data?.Response?.JobsDetail?.State === 'Success') {
                clearInterval(TaskInterval);
                callback && callback(data);
```

```
            }
        })
    }, 3000)
}
```

> **说明：**
> 推荐使用 API Explorer调试 。

合成的音频效果如下：

费用相关

- OCR 识别费用
- 语音合成费用
- 对象存储相关费用

Demo体验

具体代码可参考 cos-demo，您也可使用腾讯云对象存储控制台，在 智能工具箱 栏目中体验语音合成实际效果。

内容审核实践
内容违规（合规）场景

最近更新时间：2024-08-22 14:53:31

概述

如果您存储在对象存储（Cloud Object Storage，COS）中的数据为公有读权限，那么当您在公网中访问、传播这些数据时，需要符合相关法律法规要求，若传播的内容存在违规，腾讯云会依法进行处置，严重者可能会被永久封禁账号！
本篇实践将为您介绍几种数据管理的方法，希望能帮助您解决数据合规性问题，提前处理违规数据。

安全管理实践一：使用私有读方式访问（推荐）

通常情况下，存储桶和对象的访问权限都为私有读，但有部分用户为了方便使用，会将存储桶改为公有读权限。在公有读权限下，虽然使用方便，例如可通过对象链接直接访问对象，但这种行为存在很大风险：

- 对象可以被任何人访问，数据被肆意传播的风险大大增加。
- 可能会被一些黑产盗刷流量。

因此我们建议用户将存储桶和对象改为**私有读**权限，然后通过预授签名的方式，供第三方在有效期内进行下载访问。相关文档请参见 预签名授权下载。

安全管理实践二：使用自动化的内容审核服务

如果您的业务模式必须采用公有读的使用方式，那么建议您开启 对象存储内容审核服务。
您可以通过配置数据增量审核，对上传至 COS 的数据进行自动审核，并设置自动冻结，对审核结果为敏感的数据进行冻结处理，避免违规数据肆意传播。您可以参考以下步骤进行配置（以图片审核为例）：

1. 登录 对象存储控制台，并在**存储桶列表**页面选择需操作的存储桶，进入存储桶管理页面。
2. 在左侧导航栏中，选择**内容审核** > **自动审核配置**，单击**图片审核**。
3. 单击**添加图片审核配置**，进入图片审核配置页面，并按照如下配置项说明进行配置：

- **审核范围**：可选择审核的范围为整个存储桶或指定范围。
  指定路径：当选择指定范围，则填写您希望审核图片所在的路径。
示例1：如您需要审核指定目录 test 内的文件，则需要填写指定前缀为 test/。
示例2：如您需要审核指定前缀为123的文件，则需要填写指定前缀为123。

> **说明：**
> 您可以添加多条审核配置，但审核路径不能重复或存在包含关系。例如您已经配置了审核整个存储桶，则不能再添加针对存储桶内某个路径的审核。

审核后缀：建议您勾选**智能后缀**。

> **说明：**
> 智能判断后缀：选中后会根据文件的后缀和内容，智能判断是否为图片。

- **选择审核策略**：建议您选择**默认策略**，默认策略为算法专家经过多行业模型沉淀的策略配置，适用于大部分的内容安全需求。您也可以通过自定义策略定制个性化场景审核。支持审核涉黄、违法违规、广告审核场景，可勾选一种或多种检测场景。您可以前往 公共策略 查看如何配置审核策略。
- **已关联风险库**：与**审核策略**相关联的风险库名称，下方审核场景部分会展示出已关联风险库中包含的样本标签，例如风险库中有色情样本，则审核场景中可勾选色情内容。
- **审核场景**：建议您勾选**色情内容**等。
- **每日审核上限**：选择默认的无上限。
- **文件冻结设置**：开启敏感文件冻结服务。
- **冻结方式**：推荐使用**将文件权限变为私有读**方式，授权 COS 对相应类型文件进行自动机审冻结，从而禁止违规数据被任意访问。
- **冻结类型**：勾选**涉黄**、**违法违规**，并设置分值大于或等于90时进行冻结。
- **回调设置**：可选择开启回调，我们将把相应图片的审核结果反馈给您。您需选择回调的审核类型、回调内容、回调 URL、回调图片域名。若您选择自定义回调阈值，则需设定回调图片分值区间。设置回调地址后，数据万象会发送一份默认回调信息至您设置的回调地址，以检测回调地址是否能够正常接收回调信息。回调详情请查看 回调内容 。

4. 配置完成后，单击**保存**即可启用该功能，后续将对您新上传的图片进行审核。
5. 您可以在**内容审核** > **审核详情**中查看审核结果，进入审核详情页面。根据您的实际需求，选择相应的查看条件：

安全管理实践三：使用存量审核服务进行历史数据清洗

部分用户可能已开启自动审核服务，但历史上的旧数据仍然在外传播而导致被封禁的情况。针对这种情况我们建议您使用存量审核服务，对 COS 上的历史数据进行一次全面的清洗，将违规数据进行冻结和删除处置。

您可以按照以下步骤进行历史数据清洗：

1. 登录 对象存储控制台，并在**存储桶列表**页面选择需操作的存储桶，进入存储桶配置页面。
2. 在左侧导航栏中，选择**内容审核** > **历史数据审核**，进入存量数据审核页面。
3. 单击**创建审核任务**。
4. 在“扫描范围”界面，您可以通过不同的扫描方式，按需审核您的文件：
   扫描方式：包含存储桶文件列表、COS 清单报告、URL 列表文件三种：
   - 存储桶文件列表：您可以选择当前存储桶内的文件进行审核，扫描范围支持按文件上传时间扫描或按前缀扫描。
   - COS 清单报告：您可以选择扫描由 COS 清单功能 生成的清单列表，并将清单列表文件存放到当前存储桶内。
   - URL 列表文件：您可以选择扫描指定的 URL 列表文件，目前支持 txt 格式，每行一条 url。使用该方式暂不支持进行视频审核和文本审核。
5. 在“审核策略”界面，设置审核策略，配置相应的审核文件类型、审核场景类型，以图片审核为例：
   审核图片：
   - 审核后缀：建议您勾选**智能后缀**。
   - 大图审核：如果您有大于5MB的图片需要审核，则开启此开关。
   - 审核策略：建议您选择**默认策略**，默认策略为算法专家经过多行业模型沉淀的策略配置，适用于大部分的内容安全需求。
   - 审核场景：建议您勾选**色情内容**等内容。
6. 在“冻结策略”界面，开启敏感文件冻结服务，授权 COS 对相应类型文件进行自动机审冻结，从而禁止违规数据被任意访问。
   敏感图片冻结：勾选**涉黄**、**违法违规**，并设置分值大于或等于90时进行冻结。
7. 在“审核结果”界面，设置审核结果回调，单击**下一步**。
   开启回调设置后，我们会将审核结果发送至您指定的回调地址中，您需选择回调类型、回调内容，同时设置回调 URL。
   - 回调类型：根据您设置的审核策略，可选涉黄、违法违规、广告审核、谩骂。
   - 回调内容：可选仅回调违规文件、仅回调冻结文件、回调全部文件，支持对审核失败的文件再次审核。
   - 回调 URL：回调 URL 地址须默认返回200正确码方可使用。
   - 回调 URL 协议：可选择强制 HTTP 或 HTTPS。
8. 确认任务整体配置无误，单击**创建**即可完成任务创建。
9. 在存量数据审核页面查看审核任务结果，您可以根据任务状态进行不同的操作。

- 当任务状态为**执行中**，您可以查看**任务配置**或**终止任务**。
- 当任务状态为**执行成功**，您可以查看**审核详情**或查看**结果统计**。
- 查看审核详情：仅支持查看近1个月的审核详情，单击后会跳转到审核页面，您可以进行审核结果导出、手动审核等操作，具体操作指引请参见 审核详情。
- 查看结果统计：该页面展示了该审核任务的统计结果，如对审核结果有疑问，可前往控制台审核详情页面查看具体的审核内容。

Stable Diffusion AI 绘画审核

最近更新时间：2025-01-08 18:46:52

概述

腾讯云 Serverless 应用中心 提供了 stable-diffusion-webui 项目的 Serverless 化部署能力。该应用创建成功后，您可以使用 stable diffusion webui 的全部能力，例如文生图、图生图，以及 Lora、ControlNet 等高阶能力。本文将介绍如何通过图片审核服务对应用生成的图片做内容合规检测。

前提条件

- 登录 Serverless 应用控制台，开通 Serverless 产品服务。
- 开通对象存储产品服务，并创建一个存储桶，详见 新手指引。
- 进入 数据万象控制台，开通数据万象服务。

操作步骤

步骤一：通过 Stable Diffusion AI 应用生成图片

创建 Stable Diffusion AI 应用

1. 登录 Serverless 控制台，选择左侧导航栏中的 Serverless 应用，在 Serverless 应用页面，单击**新建应用**。

2. 在**新建应用**页面，根据页面相关信息提示进行配置：
   - **创建方式**：选择**应用市场**。
   - **模板**：选择 **Stable Diffusion AI 绘画自定义模型版**。如下图所示：

3. 单击**下一步**，根据页面相关信息提示进行配置：
   - **应用名**：例如 “aigc-auditing-server”。
   - **地域**：例如 “北京”。
   - **应用类型**：选择 **SD API（无Web UI）**的方式，通过 API 调用。

4. 单击**完成**，即可完成应用创建、函数创建以及 API 网关触发器创建。函数服务显示正常，即创建完成，如下图所示：

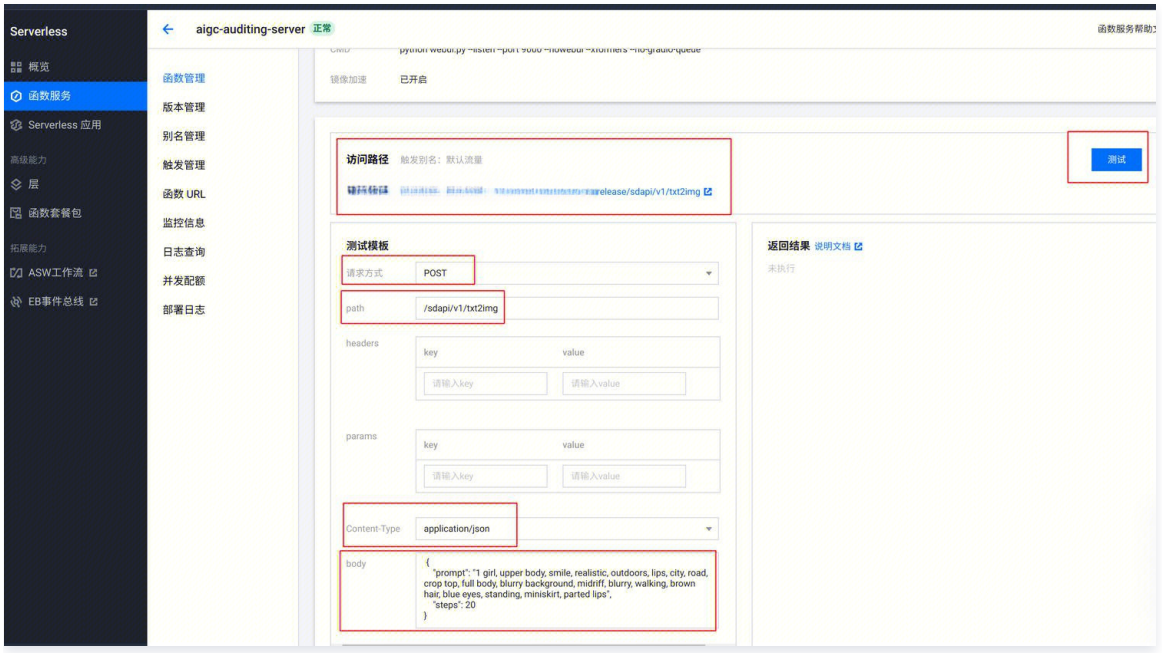
使用 Stable Diffusion AI 应用生成图片

您可以使用平台测试模板或API接口两种方式生成图片：

使用平台测试模板生成图片

通过控制台中的应用测试页面，您可以测试生成图片，按如下配置：

- **请求方式**：POST
- **path**: /sdapi/v1/txt2img
- **Content-Type**: application/json
- **body**: 传入绘图的参数，如 prompt



使用API接口生成图片（以 Python 为例）

- url：设置函数服务的访问路径。
- body：设置绘图参数。

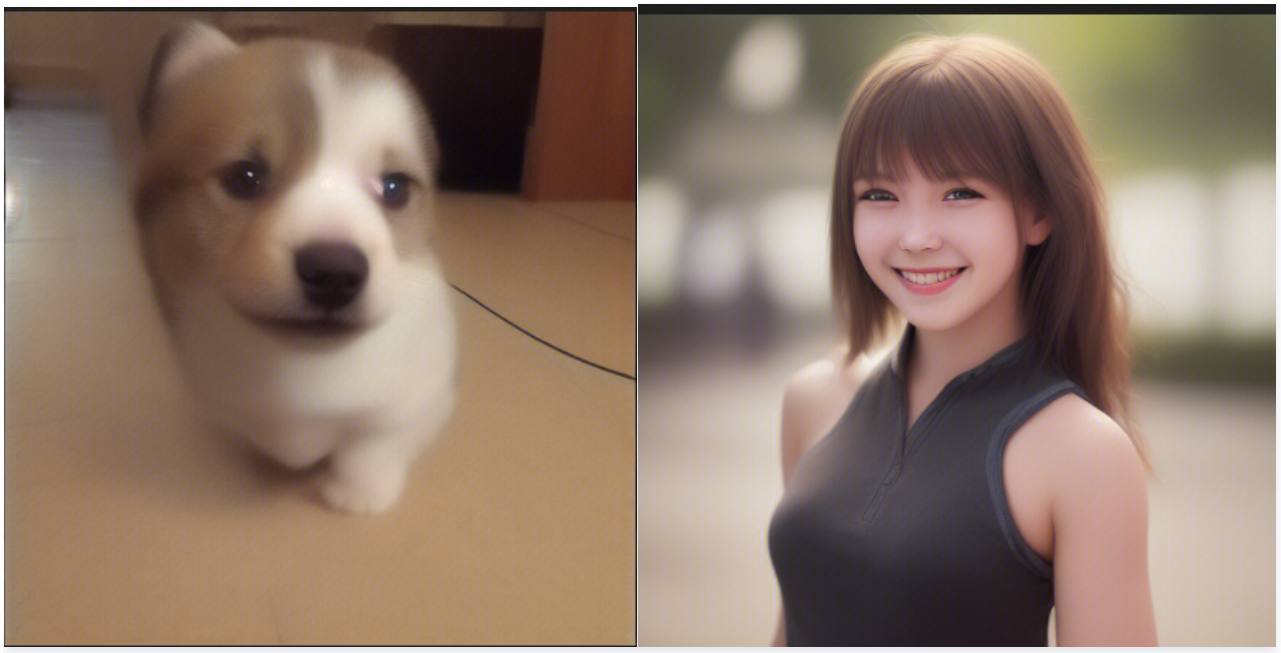
图像转换可以安装 pillow，推荐使用 pip 安装。

```
pip install pillow
```

代码示例：

```
import json
import requests
import io
import base64
from PIL import Image, PngImagePlugin

url = "https://service.*********.tencentcs.com/release/"

# 文生图接口
path='sdapi/v1/txt2img'

# 绘图参数
body = {
        "prompt": "1 girl, upper body, look at viewer",
        "steps": 20
       }

# API请求
response = requests.post(url=f'{url}{path}', json=body)
r = response.json()

# 保存生成的图片
if 'images' in r:
    for i in r['images']:
            image = Image.open(io.BytesIO(base64.b64decode(i.split(",",1)[0])))
            png_payload = {
                  "image": "" + i
            }
            response2 = requests.post(url=f'{url}sdapi/v1/png-info', json=png_payload)
            pnginfo = PngImagePlugin.PngInfo()
            pnginfo.add_text("parameters", response2.json().get("info"))
            image.save('output.png', pnginfo=pnginfo)
```

生成图片示例如下所示：



更多细节可参考 部署 Stable Diffusion AI 绘画应用（自定义模型版）。

步骤二：通过数据万象图片审核接口进行审核

图片审核概述

图片批量审核接口 支持同步、异步请求方式，您可以通过本接口对多个图片文件进行内容审核，该接口属于 POST 请求。

图片审核调用示例（以Python为例）

依赖 qcloud_cos 使用 pip 安装（推荐）。

```
pip install -U cos-python-sdk-v5
```

代码示例：

```
import sys
import os
import logging
import base64
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
from qcloud_cos.cos_comm import CiDetectType

# 正常情况日志级别使用 INFO，需要定位时可以修改为 DEBUG，此时 SDK 会打印和服务端的通信信息
logging.basicConfig(level=logging.INFO, stream=sys.stdout)

# 设置用户属性，包括 secret_id, secret_key, region等。Appid 已在 CosConfig 中移除，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成。
secret_id = os.environ['COS_SECRET_ID']           # 用户的 SecretId，建议使用子账号密钥，授权遵循最小权限指引，降低使用风险。子账号密钥获取可参见：https://cloud.tencent.com/document/product/598/37140

secret_key = os.environ['COS_SECRET_KEY']         # 用户的 SecretKey，建议使用子账号密钥，授权遵循最小权限指引，降低使用风险。子账号密钥获取可参见：https://cloud.tencent.com/document/product/598/37140

region = 'ap-beijing'         # 替换为用户的 region，已创建桶归属的 region 可以在控制台查看：https://console.cloud.tencent.com/cos5/bucket

token = None        # 如果使用永久密钥不需要填入 token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见：https://cloud.tencent.com/document/product/436/14048

scheme = 'https'        # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

bucket_name='test'          # 指定cos桶中bucket的名称

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
client = CosS3Client(config)     # 创建 cos client

img_path='./output.png'          # 生成的图片路径

with open(img_path, 'rb') as file:
    data = file.read()
    base64_data = base64.b64encode(data).decode("utf-8")     # 将数据转换为Base64编码
    response = client.ci_auditing_image_batch(               # 调用审核接口
        Bucket=bucket_name,
        DetectType=CiDetectType.PORN,
        Input=[
            {
            'Content':base64_data,      #图像数据的base64编码
            }
        ]
```

```
    )
    print(response)
```

其他细节，请参考 图片批量审核 。

图片审核结果

图片		
审核结果	正常	正常
详细结果	{'JobsDetail': [{'Category': None, 'CompressionResult': '0', 'DataId': None, 'ForbidState': '0', 'JobId': 'si21530d95416111ee82b55254008a7ac8', 'Label': 'Normal', 'PornInfo': {'Category': None, 'Code': '0', 'HitFlag': '0', 'Label': None, 'Msg': 'OK', 'Score': '0', 'SubLabel': None}, 'Result': '0', 'Score': '0', 'State': 'Success', 'SubLabel': None, 'Text': None}], 'RequestId': 'NjRlNTc2MjdfMWU2MTk3MDlfNmEwZl9kNjdjYQ==''}	{'JobsDetail': [{'Category': None, 'CompressionResult': '0', 'DataId': None, 'ForbidState': '0', 'JobId': 'si33f24754416111ee82b55254008a7ac8', 'Label': 'Normal', 'PornInfo': {'Category': None, 'Code': '0', 'HitFlag': '0', 'Label': None, 'Msg': 'OK', 'Score': '0', 'SubLabel': None}, 'Result': '0', 'Score': '0', 'State': 'Success', 'SubLabel': None, 'Text': None}], 'RequestId': 'NjRlNTc2NDdfMWU2MTk3MDlfNmEzMl9mMjBhZA==''}

内容分发网络（CDN）场景

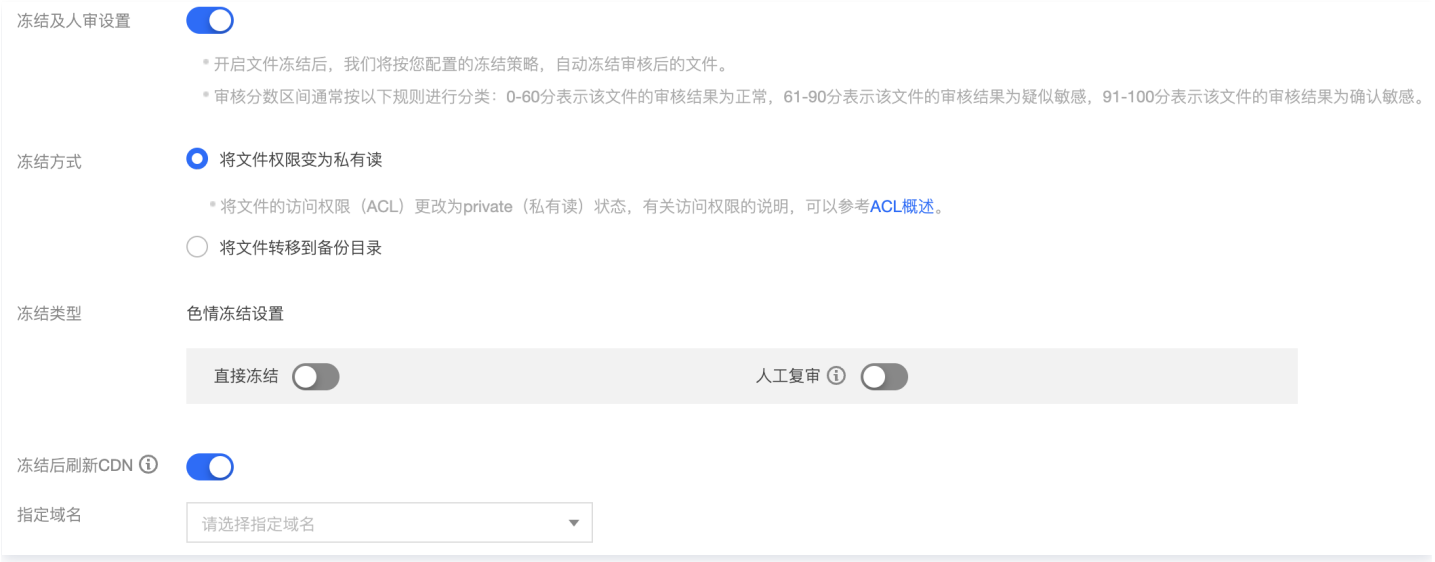
最近更新时间：2024-11-19 10:30:52

简介

内容审核功能提供了自动冻结能力，可以将违规文件自动进行冻结处理，对于使用了 CDN 场景的用户，内容审核的自动冻结能力也支持处理 CDN上的缓存数据。

操作步骤

1. 登录 数据万象控制台，在**存储桶列表**页面选择需操作的存储桶，进入存储桶管理页面。
2. 在左侧导航栏中，选择**内容审核 > 自动审核配置**，单击**图片审核**。
3. 单击**添加图片自动审核配置**，进入图片审核配置页面。
4. 开启**冻结及人审设置**，找到**冻结后刷新 CDN 开关**并开启。



5. 开启冻结后刷新 CDN 需要授权 CDN 服务，点击**前往 CAM 授权**进行授权。

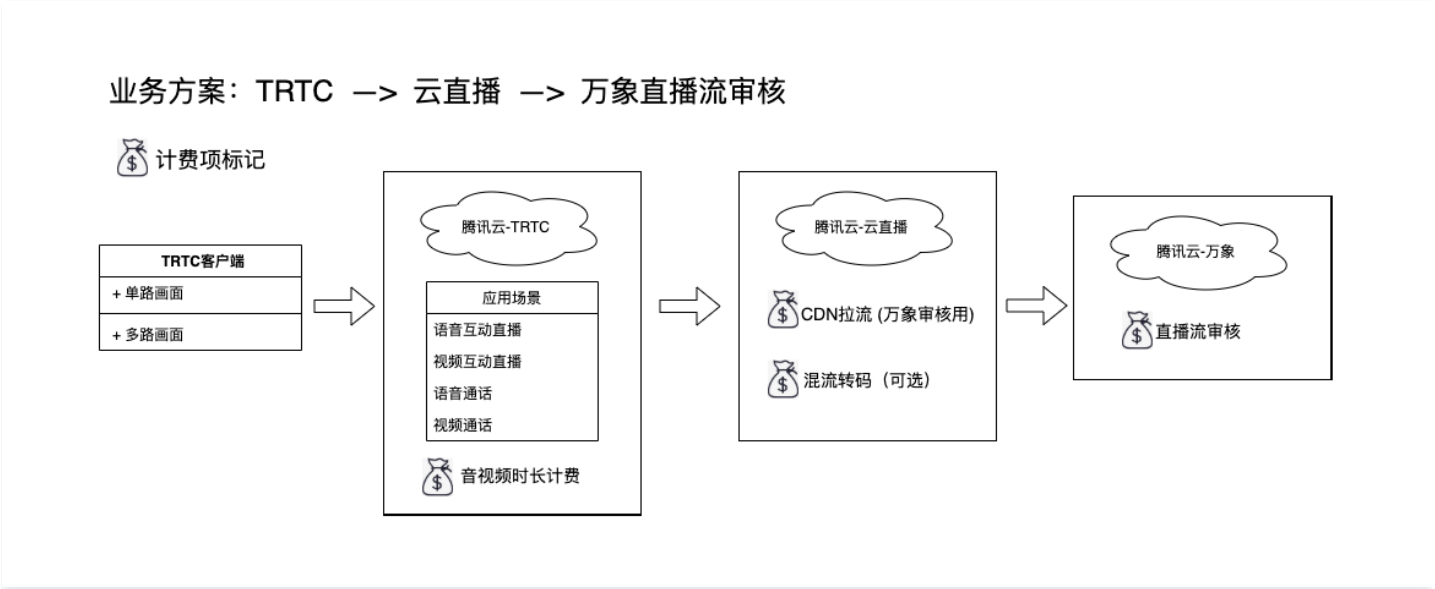
6. 授权完成后，返回当前页面刷新授权状态，选择您需要刷新的 CDN 域名，可多选。
7. 其余审核配置可参考：设置图片审核 。

实时音视频（RTC）场景

最近更新时间：2024-12-17 14:32:53

概述

随着低延时的实时音视频技术越来越成熟，也吸引了越来越多的用户使用实时音视频产品开发实现自己的音视频业务。在拓展音视频业务的同时，如何更好的管控音视频内容避免违规尤为重要。
为更好的支持用户在各种业务场景下的内容安全风险，对象存储与数据万象在原有静态音视频审核的基础上，进一步提供了流式音视频的审核，当用户直播内容属于违规内容时，可以及时发现并进行处理。
以腾讯云实时音视频产品（Tencent Real-Time Communication，TRTC）为例，整体审核流程如下图所示：



流程如下：

1. TRTC 客户端发起直播请求。
2. TRTC 服务端收到请求后进行直播。
3. 通过 发布音视频流到直播 CDN 可以拿到所需的直播流地址，有 rtmp、flv、hls 三种格式可用。
4. 使用万象直播流审核接口进行直播审核，SDK 封装此接口方便用户下载使用。
5. 客户拿到审核回调进行后续业务处理。

实践步骤

步骤1：创建存储桶

- 如果您已创建存储桶，可跳过该步骤。
- 如果您是首次使用 COS 控制台，可参考 创建存储桶 进行操作。

步骤2：创建 TRTC 应用

如果您已创建 TRTC 应用，且能够使用 TRTC 相关功能时，可直接跳转到 步骤4. 开启旁路推流 。

1. 登录 TRTC 控制台，单击**应用管理**，可以看到应用列表，单击**创建应用**，相关说明可参见 创建应用 。

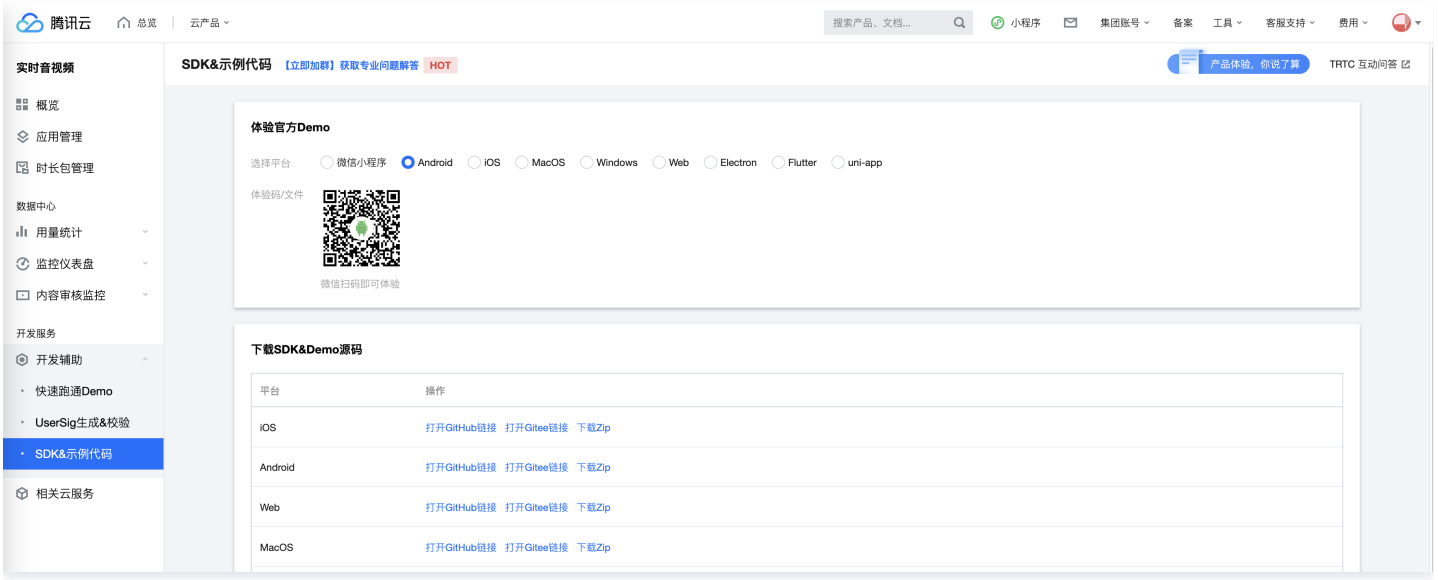
2. 创建应用后，单击应用列表右侧的**详情**，可以看到应用概览，其中 `SDKAppID` 、 `SDKSecretKey` 需要在后续流程中用到，可以先记录下来。

3. 完成以上步骤，我们知道了 `SDKAppID` 、 `SDKSecretKey` 这两个重要数据，切记不要泄漏这些数据。接下来可以开始对 TRTC DEMO 进行搭建了。

步骤3：TRTC DEMO 搭建流程

本 DEMO 目的是使客户快速了解直播流程，客户业务需求的具体实现可参考 TRTC 官网使用文档 。流程如下：

1. 下载所需平台的 SDK&Demo 源码 ，直接下载 ZIP 包即可，本次示例使用的是 Android 平台。



2. 配置 TRTC-API-Example 工程文件。找到并打开

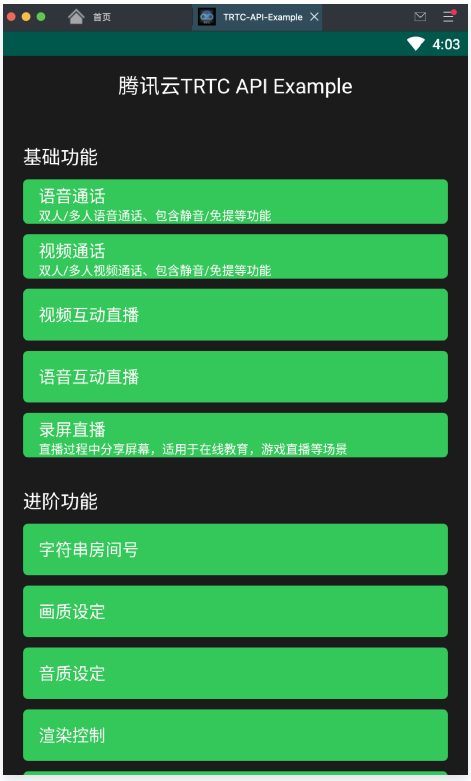
`LiteAVSDK_TRTC_Android_版本号/TRTC-API-Example/Debug/src/main/java/com/tencent/trtc/debug/GenerateTestUserSig.java`

文件。相关参数说明如下：

- BIZID：默认为 PLACEHOLDER，请设置为实际的 bizid，暂不使用 CDN 时请用 0 替换，以便编译通过。
- APPID：默认为 PLACEHOLDER，请设置为实际的 appid，暂不使用 CDN 时请用 0 替换，以便编译通过。
- SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID（上面流程已获取，即 `SDKAppID` ）。
- SECRETKEY：默认为 PLACEHOLDER，请设置为实际的 SecretKey（上面流程已获取，即 `SDKSecretKey` ）。

```
/**
 * CDN发布功能 混流bizId;
 * 使用 CND 时用实际值替换，暂不使用 CDN 时请用 0 替换，以便编译通过。
 *
 * `bizId` for CDN publishing and stream mixing.
 * Replace it with the actual value when using CND,
 * and replace it with 0 when not using CDN, so that the compilation can pass.
 */
public static final int BIZID = PLACEHOLDER;

/**
 * CDN发布功能 混流appId;
 * 使用 CND 时用实际值替换，暂不使用 CDN 时请用 0 替换，以便编译通过。
 *
 * `appId` for CDN publishing and stream mixing.
 * Replace it with the actual value when using CND,
 * and replace it with 0 when not using CDN, so that the compilation can pass.
 */
public static final int APPID = PLACEHOLDER;

/** 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。 ...*/
public static final int SDKAPPID = PLACEHOLDER;

/** 签名过期时间，建议不要设置的过短 ...*/
private static final int EXPIRETIME = 604800;

/** 计算签名用的加密密钥，获取步骤如下： ...*/
public static final String SECRETKEY = "PLACEHOLDER";
```

3. 编译运行。使用 Android Studio打开源码工程 TRTC-API-Example，单击**运行**即可。运行成功后页面如下图所示，TRTC DEMO 提供视频通话、录屏直播等多种功能示例。



步骤4：开启旁路推流

1. 登录 实时音视频控制台 。
2. 在左侧导航栏选择**应用管理**，单击目标应用右侧的**配置**。

3. 在**旁路转推配置**中，单击**开启旁路转推**右侧的，在弹出的**开启旁路转推功能**对话框中，单击**开启旁路转推功能**即可开通。

步骤5：云直播 CDN 拉流

可以通过 发布音视频流到直播 CDN 获取到直播流数据，最后生成一条直播流地址，有 rtmp、flv、hls 三种协议可用。

```
rtmp 协议的播放地址：rtmp://example.myhost.com/AppName_example/StreamName_example
flv 协议的播放地址：http://example.myhost.com/AppName_example/StreamName_example.flv
hls 协议的播放地址：http://example.myhost.com/AppName_example/StreamName_example.m3u8
```

我们推荐以 `http` 为前缀且以 `.flv` 为后缀的 **http – flv** 地址，该地址的播放具有时延低、秒开效果好且稳定可靠的特点。

步骤6：调用审核接口

获取到直播流后，使用万象的直播流审核接口进行直播审核。

- 直播流审核可使用对应语言的 COS SDK 。
- 直播审核参数说明，请参见 提交直播审核任务 。
- 直播审核回调内容，请参见 直播审核回调内容 。

提交审核时可以设置客户业务信息，审核回调时会原样返回，拿到审核回调后可以得到具体房间或具体用户的违规情况。相关参数说明，请参见 UserInfo 节点说明 。

步骤7：违规解散房间或踢用户

收到回调后判断是否违规，发现违规后发起解散房间或移出用户等处置操作，其他 API 接口可以参见 客户端 API 、服务端 API 。

- 移出用户
- 解散房间

语聊社交场景

最近更新时间：2024-12-20 12:14:22

背景

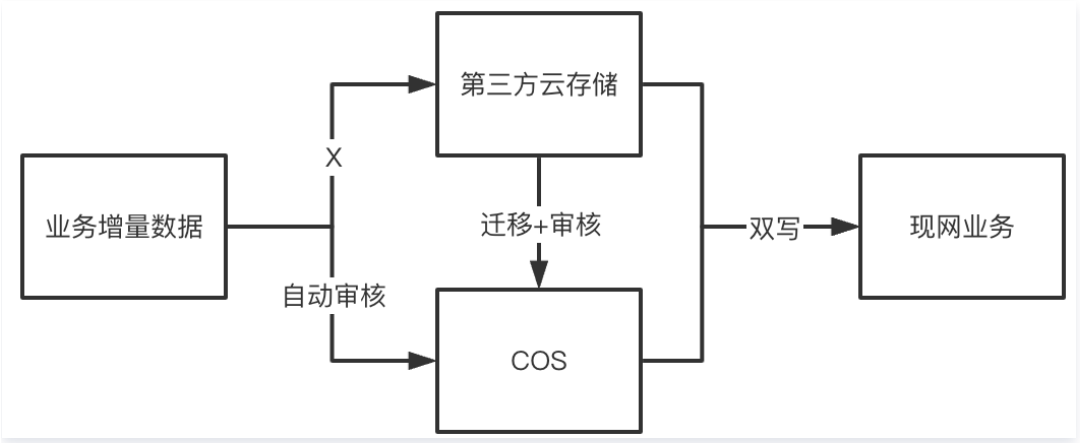
国内某移动语音社交平台，自成立至今已有数千万国内年轻人在使用。该平台以电竞游戏为切入点，极大的丰富了社交玩法，让年轻人们上分娱乐两不误。

客户痛点

随着语音工具的稳定成熟，社交聊天已经成为了游戏玩家的一项基本需求。但随着游戏玩家的逐渐增多，玩家的品性参差不齐，打着正常社交但实际做着许多影响正常聊天环境的事情，例如虚假广告、隐晦色情、辱骂谩骂等。这不但影响正常用户的使用体验，还可能被监管部门通报整改，平台的安全合规性面临着极大的考验。平台亟需一套精准、便捷的审核服务，在不影响业务的前提下，有效治理用户的社交环境。
客户初始数据不在腾讯云，本次需要将全量数据迁移至腾讯云，担心存在业务中断、审核不全等问题。
对象存储以保障业务连续性为前提，为客户提供了数据迁移 + 安全审核整套解决方案，帮助客户顺利完成了迁移。

迁移并审核解决方案

方案架构



客户从第三方厂商将数据迁移至腾讯云 COS，在迁移期间将增量数据直接改存到 COS，未迁移完毕的存量数据依然使用正常运行直至迁移完毕。
在迁移过程中，为 COS 存储桶开启自动审核和自动冻结能力，不管是增量数据进入 COS 还是迁移存量数据进入 COS，都会自动进行审核，并将违规数据进行自动冻结处理。
迁移完毕后，客户逐步停止了原平台的存储业务，并逐步灰度 COS 上的业务数据，直至全量业务通过 COS 稳定运行，全程保持自动审核能力开启。

实践步骤

1. 迁移第三方云存储数据

对于使用第三方云存储的用户，对象存储 COS 可以帮助用户方便快捷的进行迁移，使用方式请参见 第三方云存储数据迁移至 COS 。

2. 开启内容审核

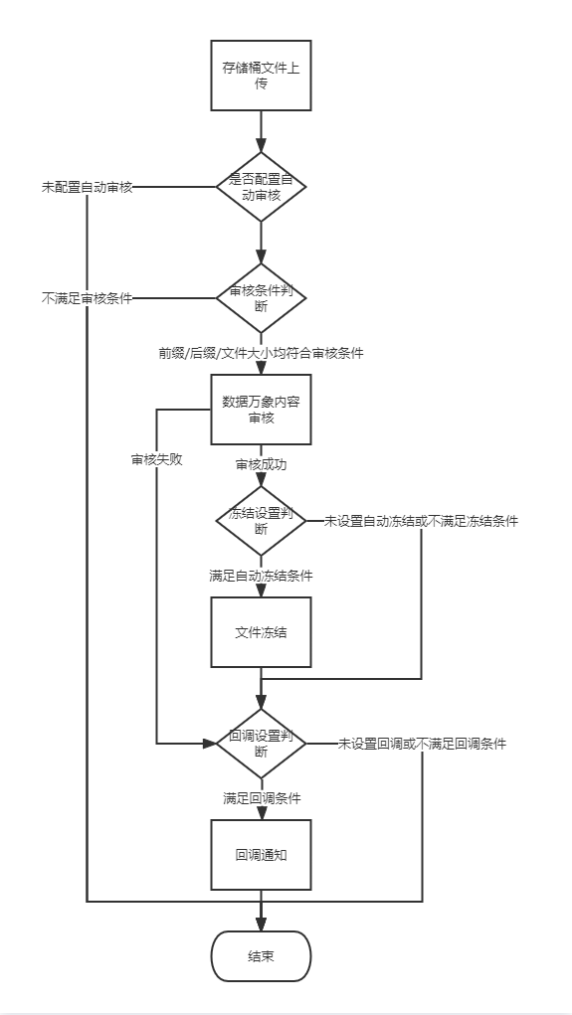
对象存储内容审核服务可审核的数据类型涵盖图片、音视频、文本、文档、网页等多媒体数据，可帮助客户有效识别色情低俗、违法违规、恶心反感等违禁内容，规避运营风险，了解更多信息请参见 内容审核概述 。

步骤1：定制化审核策略

客户的业务场景非常多，不同场景对违规内容的惩罚程度有所区别，内容审核服务提供了数十种审核二级标签供客户进行配置。客户根据自身的业务场景，梳理并定制了30多种审核策略，包含用户昵称、公屏文本、公会名称、公会评论、好友私聊等。
与通用审核策略相比，定制化的审核策略在不同场景下的审核准确率均有较大的提升，且审核策略的改动全部由后端进行配置，策略切换也不会影响业务正常运作。
配置审核策略请参见 设置审核策略 。

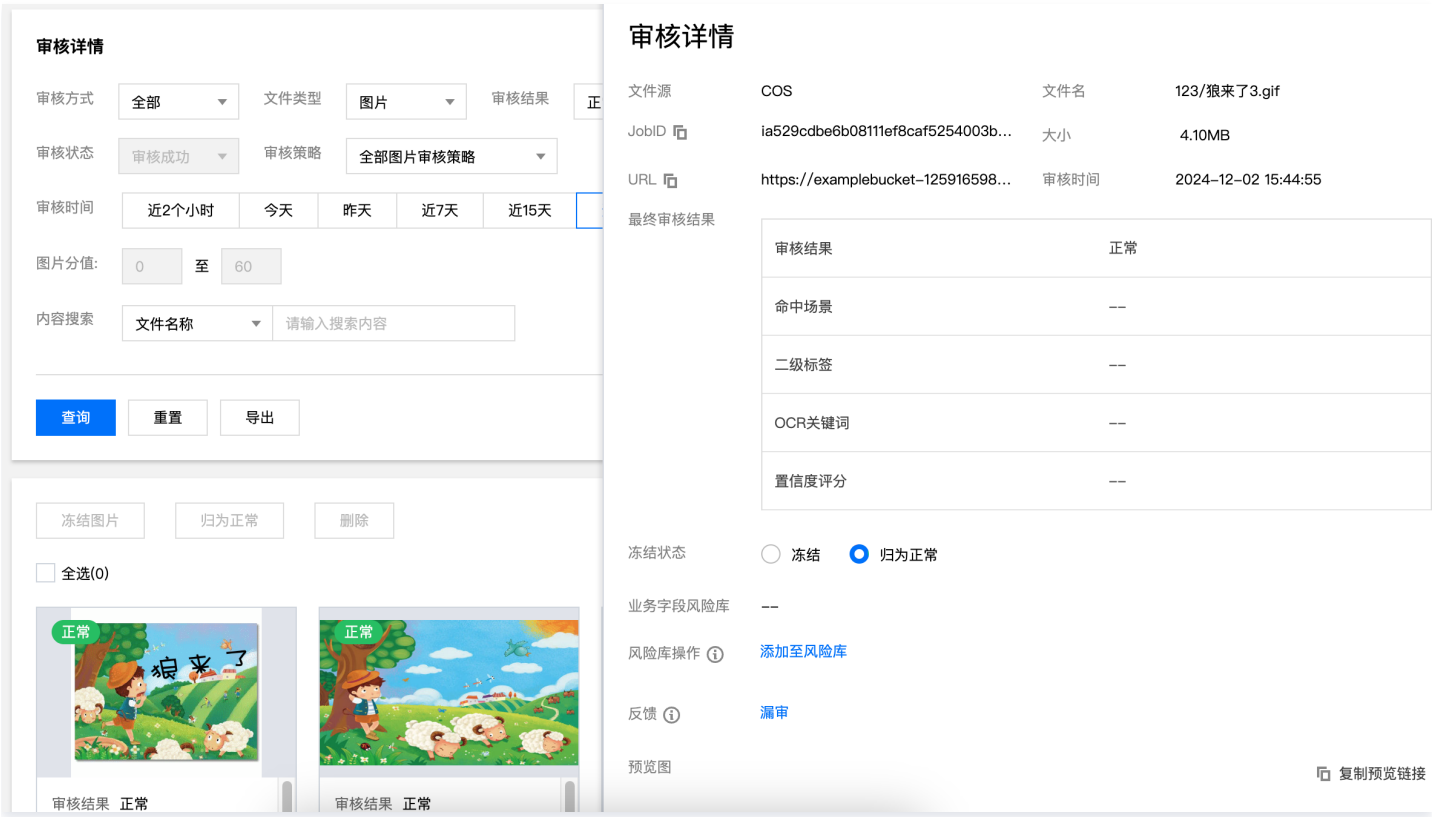
步骤2：配置自动审核

客户的业务中有大量的 UGC 图片、语音、短视频文件需要审核，此前并未整体审核过，近期被有关部门通报整改，需要在短时间内对已有的历史文件进行全面审核，同时又需要保障新增文件也进行全部审核。
对象存储提供的内容审核服务，支持控制台一键配置，无需开发，所有操作全部由产品后端自动完成，审核服务不影响现有业务流程，保证了客户的业务连续性。
客户通过在对象存储控制台开启自动审核服务，对新上传的文件进行自动审核，有关自动审核具体配置操作请参见 自动审核 。配置完成后，后端的执行过程如下图所示：



步骤3：人工复核

上述审核配置均完成后，客户通过控制台直接查看审核结果，根据审核时间、类型、名称等进行过滤，对审核结果有异议的文件进行人工冻结或人工放行。



方案优势

- 官方迁移工具，确保数据完整、稳定、高效的迁移至 COS。
- 应对合规要求，提前发现违规内容并自动处理。
- 审核策略完善，高准确率能帮助客户减少人工复核的人力投入。
- 内容审核服务通过内网拉取数据进行审核，产生的数据流量均为内网，无额外的流量费用，成本可降25%。
- 后端策略管理，可随时无中断业务调整策略，应对各种突发情况。

文档处理实践
小程序快速集成文档预览

最近更新时间：2025-04-03 14:19:02

简介

本文将介绍在微信小程序中如何通过 腾讯云数据万象（CI） 提供的文档预览功能，实现**文档转 HTML 在线预览**与**文档转图片预览**，您可根据自身需求接入不同类型的文档预览服务。

预览服务对比

本文采用同步方式接入文档预览服务，如您有更多需求，可以了解两种预览服务的异步接入方式：文档转 HTML 预览（异步） 和 文档转图片预览（异步）。

预览服务	效果	特点	适用场景
文档转 HTML 在线预览（同步方式）	转 HTML 预览，支持文档播放/翻页/全屏等功能。	接入简单，功能丰富，兼容多种文档格式。	高度还原文档样式，适用于文档格式复杂的场景。
文档转图片预览（同步方式）	转图片预览，支持图片缩放/裁剪/旋转/水印等处理。	实时在线预览，支持对生成的图片进行基础图片处理，使用简单。	比较适用于实时的 word/pdf 文档预览场景。

> **说明：**
> 关于费用介绍，请参见 文档处理费用。

文档转 HTML 在线预览

步骤1：获取文档转 HTML 预览链接

1. 创建一个存储桶
2. 开通文档预览
3. 上传文档文件
4. 获取文档对象地址，格式为：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<文档格式>`。
5. 拼接文档预览链接。需在文档对象地址后拼接 `ci-process=doc-preview&dstType=html`，即：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<文档格式>?ci-process=doc-preview&dstType=html`。您可根据自身需求决定是否继续拼接其他 请求参数。

> **说明：**
> - 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名。
> - 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问。

步骤2：配置业务域名

1. 登录 微信公众平台。
2. 选择**开发 > 开发管理 > 开发设置**，找到**业务域名**部分，单击**修改**，新增两个业务域名并保存。

   需新增的两个业务域名如下：
   - `https://<BucketName-APPID>.cos.<Region>.myqcloud.com`：您需要进行文档转 HTML 预览的对象存储域名。
   - `https://ci-1.prvsh.myqcloud.com`：文档转 HTML 预览底层服务域名。

3. 下载校验文件至本地，并将文件放置在**域名根目录**下。
   - 登录 对象存储控制台 ，进入相应存储桶根目录下，上传校验文件，设置该校验文件访问权限为**公有读**。
   - 联系我们 为您在 `https://ci-1.prvsh.myqcloud.com` 域名下配置校验文件，我们将在7个工作日内完成配置。

关于业务域名具体说明，请参见 业务域名说明 。

> **说明：**
> 关于是否需要配置业务域名，您可根据自身情况进行选择。如果只是个人在本地调试，可以在**微信开发者工具 > 详情 > 本地设置**中勾选 “不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书”，可不必配置业务域名。如果想要发布上线，那么必须有相应企业小程序并且配置业务域名。

步骤3：使用 webview 实现文档转 HTML 预览

1. 在 微信开发者工具 中，新建一个微信小程序。
2. 页面中使用 **web-view** 组件实现内嵌文档预览页面。此处的 `url` 为文档转 HTML 预览链接。

```
<web-view src="{{url}}"></web-view>
```

获取示例代码：微信小程序 – 文档转 HTML 预览示例代码

文档转图片预览

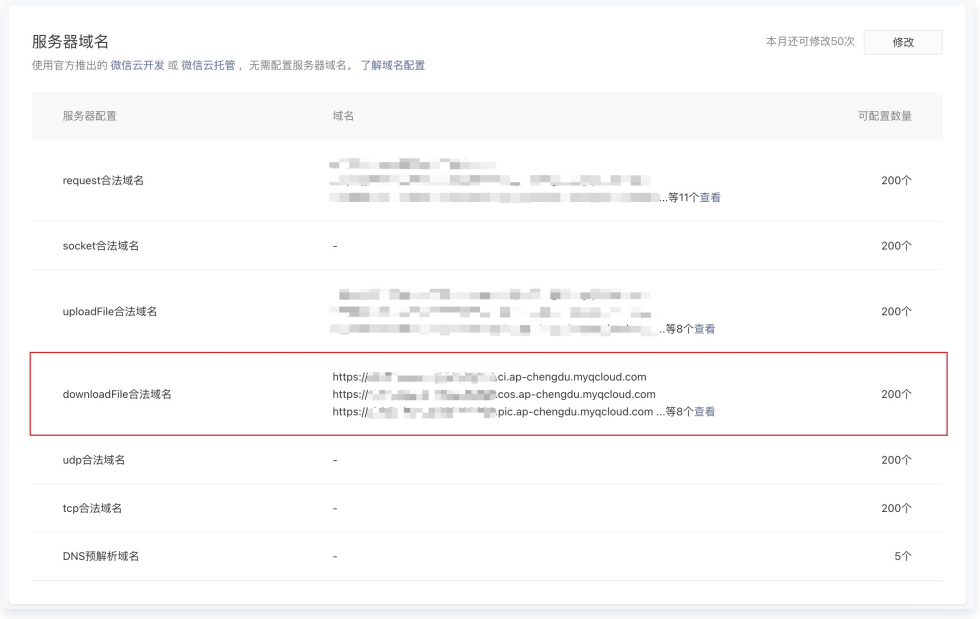
步骤1：获取文档转图片预览链接

1. 创建一个存储桶
2. 开通文档预览
3. 上传文档文件
4. 获取文档对象地址，格式为：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<文档格式>` 。
5. 拼接文档预览链接。需在文档对象地址后拼接 `ci-process=doc-preview` ，即：`https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<文档格式>?ci-process=doc-preview` 。您可根据自身需求决定是否继续拼接其他 请求参数 。文档转图片预览支持 基础图片处理 处理参数，您可拼接 **ImageParams** 参数在预览时对图片进行处理。

> **说明：**
> - 若存储桶为私有读写，则对象地址需要携带签名，详情请参见 请求签名 。
> - 若存在跨域问题，则需要进行存储桶跨域访问 CORS 设置，详情请参见 设置跨域访问 。

步骤2：配置 downloadFile 合法域名

1. 登录 微信公众平台 。
2. 选择**开发 > 开发管理 > 开发设置**，找到**服务器域名**部分，修改 **downloadFile 合法域名**。



3. 在 **downloadFile 合法域名**中新增域名 `https://<BucketName-APPID>.cos.<Region>.myqcloud.com` ，即您需要进行文档转图片预览的对象存储域名。保存配置。详情请参见 服务器域名配置 。

步骤3：实现文档转图片预览

1. 在 微信开发者工具 中，新建一个微信小程序。
2. 使用 **wx.downloadFile** 下载文档图片资源到本地。可从请求结果中获取文档总页数与临时图片地址。此处的 `url` 为文档转图片预览链接，例如 `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<文档格式>?ci-process=doc-preview&page=1&dstType=png` 。

```
wx.downloadFile({
    url,
    method: "GET",
    success(res) {
        const totalPage = Number(res?.header["X-Total-Page"]) || 0;
        const imageUrl = res?.tempFilePath || '';
    },
    fail(err) {
        console.log(err);
    }
});
```

3. 页面加载文档图片，实现文档转图片预览。imageUrl 即下载的临时图片地址。

```
<image src="{{imageUrl}}"></image>
```

4. 您可自定义文档转图片预览页面结构与效果，例如一次性请求或者以懒加载的方式请求文档的预览图片，使用 **swiper** 组件实现文档转图片预览的滑动翻页效果。

获取示例代码：微信小程序 – 文档转图片预览示例代码

Demo 体验

您可使用微信扫描下方的小程序码，在文档预览栏目中使用示例文件或自行上传文件，可分别体验小程序中文档转 HTML 预览与文档转图片预览的实际效果。

工作流实践
使用自定义函数管理 COS 文件

最近更新时间：2024-01-16 17:58:31

概述

对象存储（Cloud Object Storage，COS）工作流提供了一系列针对音视频、图片等媒体文件的处理能力，用户可以按照自身需求，灵活快速地搭建媒体处理流水线。随着越来越多用户接入 COS 工作流，一些定制化需求也被提上议程。为了保障灵活性，COS 工作流推出了自定义云函数功能，支持用户在工作流中配置云函数节点，在云函数中实现定制化逻辑。
为了进一步降低用户的使用门槛，COS 工作流整理了常用的云函数功能模板，并将其创建流程集成至节点的配置步骤中，方便用户对处理前的源文件、处理后的产物文件进行后续加工。现已支持修改对象属性、移动对象，删除对象等基本操作。

应用场景

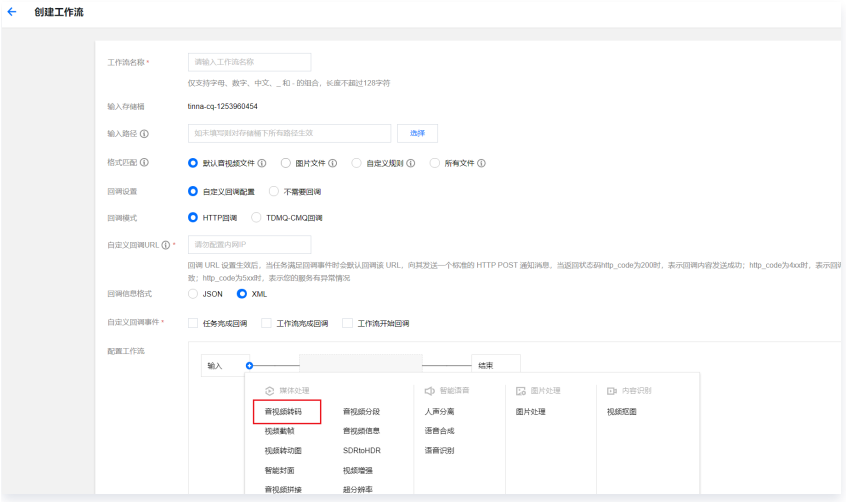
- 在媒体处理后，对源文件进行移动、沉降等处理，降低存储成本。
- 在媒体处理后，对产物文件进行标签设置，修改头部等处理，方便业务使用。

方案优势

- 开箱即用：无需开发函数逻辑，无需关注复杂的部署流程，简易配置即可使用。
- 配置灵活：可以按需配置常用功能节点，分别针对源文件和产物文件进行不同的业务操作。
- 拓展方便：支持用户修改云函数逻辑，以满足更多定制化需求。

操作步骤

1. 登录 对象存储控制台 。
2. 在左侧导航栏中，单击**存储桶列表**，进入存储桶列表页面。
3. 单击待操作的存储桶，进入存储桶详情页面。
4. 在左侧导航栏中，选择**数据工作流 > 工作流管理**，单击**创建工作流**。
5. 在创建工作流页面，配置业务需要的媒体处理节点，例如“音视频转码”节点，详情可参阅 工作流 。



6. 在创建工作流页面，添加**自定义函数**节点，选择您需要的常用功能函数。

7. 如果您尚未创建此类函数，则单击新增函数。

8. 创建常用功能函数的步　如下：

8.1 填写函数基础配置。输入函数名称前缀，勾选授权SCF服务，单击下一步。

8.2 填写属性配置。按业务需求设置存储类型，自定义头部等，单击**下一步**。

8.3 勾选处理对象，可以只针对工作流源文件执行该操作。

8.4 单击**确认**。

8.5 COS 工作流封装了函数创建，版本发布，别名切流等过程，请等待其创建完成。

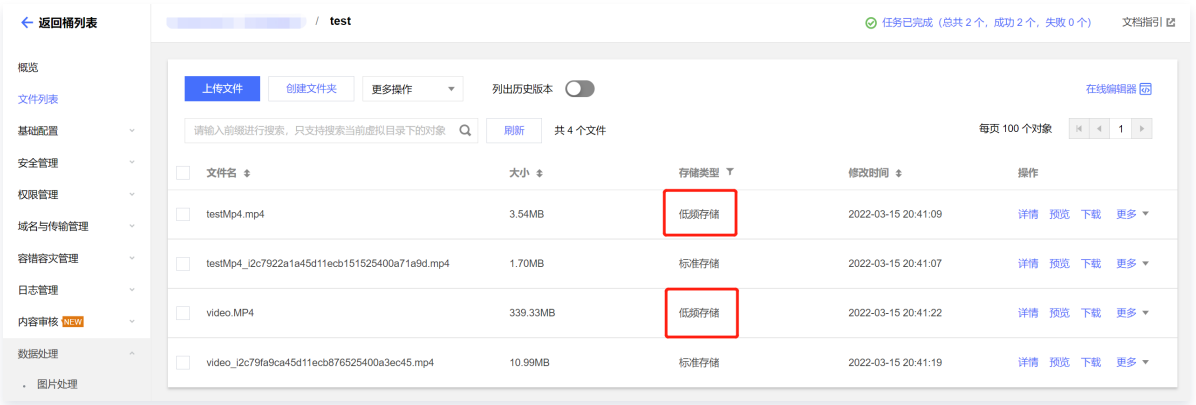
8.6 创建完成后，选中刚刚创建的函数实例，单击**确定**。

8.7 单击**保存**当前工作流即可。

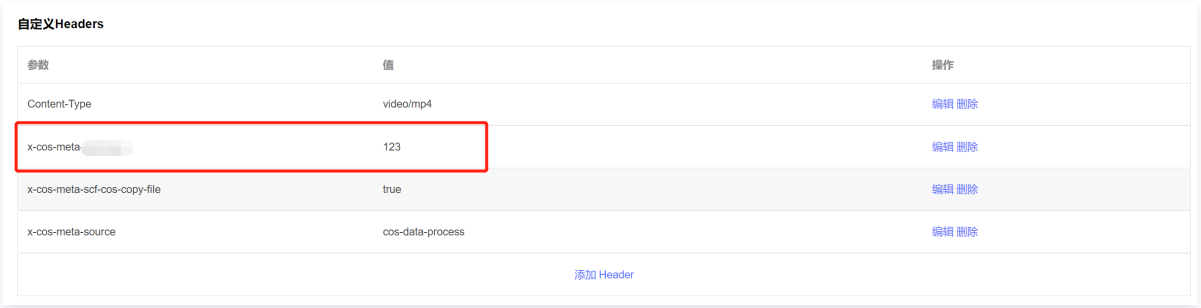
操作验证

1. 登录 对象存储控制台 。
2. 在左侧导航栏中，单击**存储桶列表**，进入存储桶列表页面。
3. 单击待操作的存储桶，进入存储桶详情页面。
4. 在左侧导航栏中，选择**数据工作流 > 工作流**，进入工作流管理页面。
5. 找到刚创建的工作流，单击启用，并前往指定存储桶路径上传媒体文件，等待工作流执行。

6. 待工作流运行结束后，即：
   可以看到媒体处理成功，产物已生成。



源文件已成功设置了存储类型和自定义头部。



文件管理实践
如何在上传文件请求回包中返回文件信息

最近更新时间：2024-07-22 16:22:11

背景介绍

本文将介绍如何在上传文件到 COS 时同步获取文件信息，如图片的宽高、格式等。
目前，可以通过 COS 上传接口，如 PUT Object、CompleteMultipartUploads 等将文件存储至 COS 存储桶中，我们针对以下三种场景提供上传时同步获取文件信息的方式：

场景	文件类型	实现方式
上传文件时，同步获取文件元信息	所有文件	ReturnBody
上传文件时，同步获取图片信息	图片文件	• 方式一：ReturnBody • 方式二：Pic-Operations
上传文件时，同步获取媒体文件信息	媒体文件	ReturnBody

- **ReturnBody** 是 COS 对外提供的一种获取文件信息的方式。在上传请求（PUT Object、POST Object、CompleteMultipartUploads）中携带 **x-cos-return-body** 头部，传入**自定义**的 ReturnBody 参数，便可在请求响应结果中获取到文件信息，可参考 ReturnBody 。
- **Pic-Operations** 是上传时的一个请求包头，在上传请求（PUT Object、POST Object、CompleteMultipartUploads）中携带该包头并设置需要返回原图信息的参数，就可在图片上传至 COS 时同步获取原图信息，可参考 图片处理机制介绍 。

> **说明：**
> Pic-Operations 是由数据万象服务提供的获取图片信息的能力，使用时会产生 图片基础处理费用 。

主要流程如下图所示：

场景一：同步获取文件元信息

如需要在上传文件后，同步获取文件元信息，可以通过 ReturnBody 实现。在上传请求头部中携带由文件元信息组成的 ReturnBody 参数，便可在请求响应结果中获取到文件元信息。
ReturnBody 提供以下文件元信息参数：

变量名	变量说明
bucket	对象上传的目标存储桶
object	对象上传到存储桶内，使用的对象名称

size	对象大小，单位为 Byte
region	对象上传的存储桶所在地域
mimeType	对象元数据 Content-Type

使用时，需要先自定义构造 **ReturnBody 参数**，这是您希望在返回结果中获取到的文件元信息。

> ⚠ **注意：**
> ReturnBody 参数的 key 可以自定义名称，value 必须跟上述 ReturnBody 提供的变量名保持一致。

请求示例

```
// ReturnBody 参数的 key 可以自定义名称，value 必须跟 ReturnBody 提供的变量保持一致
{
 "bucket": "${bucket}",
 "key": "${object}",
 "filesize": "${size}",
 "mime_type": "${mimeType}"
}
```

再将 ReturnBody 参数转换为字符串，并进行 URL 安全 的 Base64 编码，可以得到：

```
eyJidWNrZXQiOiIke2J1Y2tldH0iLCJrZXkiOiIke29iamVjdH0iLCJmaWxlc2l6ZSI6IiR7c2l6ZX0iLCJtaW1lX3R5cGUiOiIke21
pbWVUeXBlfSJ9
```

之后便可在上传文件的请求中，通过设置请求头部 **x-cos-return-body** 传入上面 Base64 编码后的结果，即可在请求响应中获取到自定义的 bucket、key、filesize、mime_type 文件信息，步骤详情可查看 ReturnBody 使用步骤 。

响应示例

```
HTTP/1.1 200 OK
x-cos-request-id: NWU5MDNkZjVfYzVjNzJhMDlfMjVhNzNfMWMy****

{
 "bucket":"examplebucket-1250000000000",
 "filesize":"30262104",
 "key":"test.pptx",
 "mime_type":"application/vnd.openxmlformats-officedocument.presentationml.presentation"
}
```

场景二：同步获取图片信息

如需要在上传图片文件后，同步获取图片信息，有两种实现方式：通过 ReturnBody 同步获取图片信息 和 通过 Pic-Operations 同步获取图片信息。同步获取图片信息后，可以用于后续处理，如给图片分类、打标签等操作。

> ⚠ **注意：**
> - 两种实现方式都依赖数据万象（Cloud Infinite，CI）服务的能力。使用前需先开通数据万象并绑定存储桶，可参考 存储桶操作 。
> - 获取图片信息会由 CI 服务收取基础图片处理费用，详情可参考 图片处理费用 。

方式一：通过 ReturnBody 同步获取图片信息

通过 ReturnBody 同步获取图片信息的方式，需要在上传请求头部中携带由图片信息组成的 ReturnBody 参数，便可在请求响应结果中获取到图片信息。

> ⓘ **说明：**

> 当前仅支持在中国大陆公有云地域使用。

ReturnBody 提供的图片信息包括：**图片基本信息（imageInfo）和图片 exif 信息**。
其中，图片基本信息（imageInfo）包括：

变量格式	变量说明
imageInfo.format	图片类型。例如 png、gif 等
imageInfo.width	图片的宽度。单位为像素（px）
imageInfo.height	图片的高度。单位为像素（px）
imageInfo.size	图片的大小。单位为 Bytes
imageInfo.md5	图片的 md5 值
imageInfo.frame_count	图片的帧数。静态图为1，动图为对应的帧数
imageInfo.bit_depth	图片的位深
imageInfo.vertical_dpi	图片的垂直分辨率
imageInfo.horizontal_dpi	图片的水平分辨率

图片 exif 信息主要是记录拍摄的硬件或软件信息，包含但不限于以下几项：

变量格式	变量说明
exif.ColorSpace.val	色域、色彩空间
exif.DateTime.val	创建时间
exif.Orientation.val	方向
exif.ResolutionUnit.val	分辨率单位
exif.XResolution.val	X 方向分辨率
exif.YResolution.val	Y 方向分辨率
...	...

使用时，仍需根据 ReturnBody 提供的图片信息先自定义构造 **ReturnBody 参数**。为了避免 ReturnBody 携带的内容太多，影响接口性能，不支持直接填写 `${imageInfo}` 或 `${exif}`，必须写明具体的子变量，例如 `${变量名.子变量}`、`${变量名.二级子变量}`。

> **注意：**
> ReturnBody 参数的 key 可以自定义名称，value 必须跟上述 ReturnBody 提供的变量名保持一致。

请求示例

```
// ReturnBody 参数的 key 可以自定义名称，value 必须跟 ReturnBody 提供的变量保持一致
{
 "color_space": "${exif.ColorSpace.val}",
 "format": "${imageInfo.format}",
 "width": "${imageInfo.width}",
 "height": "${imageInfo.height}",
 "md5": "${imageInfo.md5}",
 "bit_depth": "${imageInfo.bit_depth}",
 "vertical_dpi": "${imageInfo.vertical_dpi}",
 "horizontal_dpi": "${imageInfo.horizontal_dpi}"
}
```

再将 ReturnBody 参数转换为字符串，并进行 URL 安全 的 Base64 编码，可以得到：

```
eyJjb2xvcl9zcGFjZSI6IiR7ZXhpZi5Db2xvclNwYWNlLnZhbH0iLCJmb3JtYXQiOiIke2ltYWdlSW5mby5mb3JtYXR9Iiwid2lkdGg
iOiIke2ltYWdlSW5mby53aWR0aH0iLCJoZWlnaHQiOiIke2ltYWdlSW5mby5oZWlnaHR9IiwibWQ1IjoiJHtpbWFnZUluZm8ubWQ1fS
IsImJpdF9kZXB0aCI6IiR7aW1hZ2VJbmZvLmJpdF9kZXB0aH0iLCJ2ZXJ0aWNhbF9kcGkiOiIke2ltYWdlSW5mby52ZXJ0aWNhbF9kc
Gl9IiwiaG9yaXpvbnRhbF9kcGkiOiIke2ltYWdlSW5mby5ob3Jpem9udGFsX2RwaX0ifQ
```

之后便可在上传文件的请求中，通过设置请求头部 x-cos-return-body 传入上面 Base64 编码后的结果，即可在请求响应中获取到自定义的 color_space、format、width、height 等图片信息，步骤详情可查看 ReturnBody 使用步骤 。

响应示例

```
HTTP/1.1 200 OK
x-cos-request-id: NWU5MDNkZjVfYzVjNzJhMDlfMjVhNzNfMWMy****

{
 "bit_depth":"8",
 "color_space":"sRGB",
 "format":"png",
 "height":"800",
 "horizontal_dpi":"0",
 "md5":"4f5c260f63af3c56cea542a1c62a0a1b",
 "vertical_dpi":"0",
 "width":"1200"
}
```

方式二：通过 Pic-Operations 同步获取图片信息

通过 Pic-Operations 同步获取图片信息的方式，只需要在请求包头部中加入 Pic-Operations 项并设置需要返回原图信息参数，就可在图片上传至 COS 后同步获取原图信息，可参考 图片处理机制介绍 。
通过 putObject 上传图片，请求头中设置 Pic-Operations，设置 is_pic_info 为 1，代表返回原图信息。在请求结果中可以拿到原图图片信息。

请求示例

```
PUT /filename.jpg HTTP/1.1
Host: examplebucket-1250000000.cos.ap-chengdu.myqcloud.com
Date: Tue, 04 Apr 2023 09:06:15 GMT
Authorization:XXXXXXXXXXXX
Pic-Operations: {"is_pic_info":1}
Content-Length: 64

[Object]
```

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 645
Date: Tue, 04 Apr 2023 09:06:16 GMT
Status: 200 OK
x-cos-request-id: NWFjMzQ0MDZfOTBmYTUwXzZkZV8z****

<UploadResult>
      <OriginalInfo>
          <Key>filename.jpg</Key>
          <Location>examplebucket-1250000000.cos.ap-chengdu.myqcloud.com/filename.jpg</Location>
          <ETag>&amp;quot;580cd6930444576523c25f86ce2af9b1fc2d5484&amp;quot;</ETag>
```

```
            <ImageInfo>
                <Format>JPEG</Format>
                <Width>640</Width>
                <Height>427</Height>
                <Quality>100</Quality>
                <Ave>0xa18454</Ave>
                <Orientation>1</Orientation>
                <FrameCount>1</FrameCount>
            </ImageInfo>
        </OriginalInfo>
</UploadResult>
```

说明：

- Pic-Operations 支持 COS V5 的分块上传同步获取图片信息，在使用 COS V5 的 Complete Multipart Upload 接口时只需在请求包头部中加入 Pic-Operations 项，在分片上传完成后可以在请求结果中获取到图片信息。
- Pic-Operations 只能在数据万象支持的地域使用，详情请参见 地域与域名 。
- 仅支持 32M 以内的图片。

场景三：同步获取媒体文件信息

注意：

通过 ReturnBody 同步获取媒体文件信息依赖数据万象（Cloud Infinite，CI）服务的媒体处理功能。使用前需先开通数据万象并绑定存储桶，可参考 存储桶操作，并开启媒体处理功能开关。获取媒体文件信息会由 CI 服务收取视频元信息获取费用，详情可参考 媒体处理费用 。

如需要在上传媒体文件后，同步获取媒体文件信息，可以通过 ReturnBody 实现。在上传请求头部中携带由媒体文件信息组成的 ReturnBody 参数，便可在请求响应结果中获取到媒体文件信息。

说明：

当前仅支持在中国大陆公有云地域使用。

ReturnBody 提供的媒体文件信息主要内容可参考下表，更多可查看 媒体文件信息 。

变量格式	变量说明
videoInfo.video.bit_rate	视频比特率，单位为 kbps
videoInfo.video.codec_name	视频编解码格式名字
videoInfo.video.profile	视频编码档位
videoInfo.video.pix_fmt	像素格式
videoInfo.audio.bit_rate	音频比特率，单位 kbps
videoInfo.audio.codec_name	音频编解码格式名字
videoInfo.format.duration	时长，单位为秒
...	...

使用时，仍需根据 ReturnBody 提供的媒体文件信息先自定义构造 **ReturnBody 参数**。为了避免 ReturnBody 携带的内容太多，影响接口性能，不支持直接填写 `${videoInfo}`，必须写明具体的子变量，例如 `${变量名.子变量}` 、 `${变量名.二级子变量}` 。

注意：

ReturnBody 参数的 key 可以自定义名称，value 必须跟上述 ReturnBody 提供的变量名保持一致。

请求示例

```
// ReturnBody 参数的 key 可以自定义名称，value 必须跟 ReturnBody 提供的变量保持一致
{
 "video_bit_rate": "${videoInfo.video.bit_rate}",
 "video_codec_name": "${videoInfo.video.codec_name}",
 "video_profile": "${videoInfo.video.profile}",
 "video_pix_fmt": "${videoInfo.video.pix_fmt}",
 "audio_bit_rate": "${videoInfo.audio.bit_rate}",
 "audio_codec_name": "${videoInfo.audio.codec_name}",
 "duration": "${videoInfo.format.duration}",
}
```

再将 ReturnBody 参数转换为字符串，并进行 URL 安全 的 Base64 编码，可以得到：

```
eyJ2aWRlb19iaXRfcmF0ZSI6IiR7dmlkZW9JbmZvLnZpZGVvLmJpdF9yYXRlfSIsInZpZGVvX2NvZGVjX25hbWUiOiIke3ZpZGVvSW5
mby52aWRlby5jb2RlY19uYW1lfSIsInZpZGVvX3Byb2ZpbGUiOiIke3ZpZGVvSW5mby52aWRlby5wcm9maWxlfSIsInZpZGVvX3BpeF
9mbXQiOiIke3ZpZGVvSW5mby52aWRlby5waXhfZm10fSIsImF1ZGlvX2JpdF9yYXRlIjoiJHt2aWRlb0luZm8uYXVkaW8uYml0X3Jhd
GV9IiwiYXVkaW9fY29kZWNfbmFtZSI6IiR7dmlkZW9JbmZvLmF1ZGlvLmNvZGVjX25hbWV9IiwiZHVyYXRpb24iOiIke3ZpZGVvSW5m
by5mb3JtYXQuZHVyYXRpb259In0
```

之后便可在上传文件的请求中，通过设置请求头部 x-cos-return-body 传入上面 Base64 编码后的结果，即可在请求响应中获取到自定义的 video_bit_rate、video_codec_name、video_profile 等媒体文件信息，步骤详情可查看 ReturnBody 使用步骤 。

响应示例

```
HTTP/1.1 200 OK
x-cos-request-id: NWU5MDNkZjVfYzVjNzJhMDlfMjVhNzNfMWMy****

{
 "audio_bit_rate":"189.375000",
 "audio_codec_name":"aac",
 "duration":"123.875000",
 "video_bit_rate":"2936.675000",
 "video_codec_name":"h264",
 "video_pix_fmt":"yuv420p",
 "video_profile":"Main"
}
```

版权保护解决方案

最近更新时间：2024-11-06 18:42:22

背景

随着网络基础设施的完善，当今各大平台的图片、视频内容正迎来爆发式增长。创作者、平台在享受媒体内容带来的流量红利的同时，也面临着内容被盗用、知识产权受到侵犯的风险。
针对以上痛点，数据万象提供了文档水印、图片和视频的明水印、盲水印，以及视频 DNA 等高级功能，为各种业务场景提供接入方便、高性价比的一站式版权保护解决方案。

图片版权保护方案介绍

图片明水印

图片明水印在各类社交平台、UGC 内容创作平台已经得到广泛应用，下文将主要介绍数据万象图片明水印功能的特性与优势：

- 支持文字水印与图片水印。
- 支持平铺水印。
- 接入方便，控制台可视化页面操作，实时预览水印效果。
- 支持配置样式，一次调用进行多种图片处理操作。
- 水印明显，警示程度高。

您可通过以下示例，并结合产品文档，在对象存储 COS 的图片下载链接后拼接相关参数，即可实现下载时处理。示例如下：

```
https://examples-1251000004.cos.ap-shanghai.myqcloud.com/sample.jpeg?watermark/2/text/6IW-6K6v5LqRLeaVsOaNruS4h-ixoQ==/fill/IzNEM0QzRA/fontsize/20/dissolve/50/gravity/northeast/dx/20/dy/20/batch/1/degree/45
```

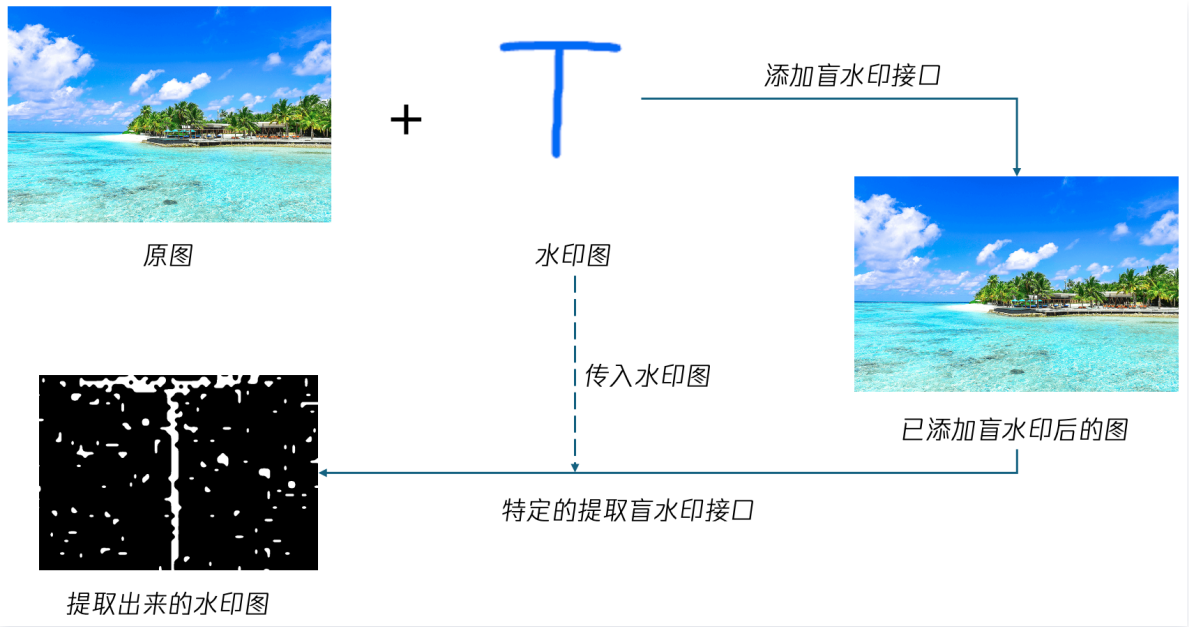
处理后效果如下图所示：

明水印适用于商品图等品牌明显、强警示图片归属场景，该方案具有价格低廉、配置方便、处理时延短等优点。但当明水印被涂抹、遮盖后，图片失去保护，并且可见的水印也会一定程度上影响到用户查看图片时的视觉体验。因此，如果您对水印的抗攻击能力和图片质量有较高的要求，可以选择下文中的盲水印方案。

图片盲水印

盲水印功能是一种更安全的水印模式，您可将水印图以**不可见的形式**添加到原图信息中，同时不会对原图质量产生太大影响。在图片被盗取后，您可对疑似被盗取的资源进行盲水印提取，以此**验证图片归属**。盲水印的抗攻击能力极强，即使图片经过旋转、裁剪、涂抹、压缩等处理，解码算法在多数情况下仍然能够

提取出水印信息。盲水印添加及提取过程如下图所示：



有关盲水印的更多操作指引，请查看 图片盲水印 。

视频版权保护方案介绍

数据万象支持将图片、字符串隐藏在视频中，不易被探知和再次修改。通过隐藏在内容载体中的水印，可确认内容的创作人、版权所有人、传播者，判断视频内容是否被篡改，水印不易被探知和再次修改，同时也不会破坏视频载体的完整性与可观赏性。
数字水印具备透明性、鲁棒性、安全性及标识性特点，创作者可充分利用这些特点及相关功能。目前数据万象数字水印服务已助力腾讯视频、视频号等进行版权保护，并已获取 ChinaDRM 权威认证。
您还可自定义数字水印内容，添加数字水印前后效果对比图如下：

> **说明**
> 数据万象不仅支持视频数字水印，也可提供视频 DNA、视频加密等版权保护方案。如您需要相关服务，请通过 联系我们 获取测试名额。

文档版权保护方案介绍

数据万象为文档版权保护提供了以下两种方案，您可根据业务场景选择使用：

结合图片处理将文档转码为带水印图片

数据万象文档转码功能支持将文档转码为图片格式，在 COS 文件下载链接后方拼接参数就能实现转码，添加水印参数即可实现自定义水印。该功能可实现文档的轻量化分发，同时保护文档内容不被盗用。示例如下：

```
https://ci-h5-demo-1258125638.cos.ap-chengdu.myqcloud.com/defaults/defaultSlides.pptx?ci-process=doc-preview&page=9&ImageParams=watermark/2/text/Q09T5paH5qGj6aKE6KeI/fontsize/20/batch/1/dissolve/30/degree/45
```

转换后效果如下图所示：

使用文档转 HTML 功能直接在网页上添加水印

文档转 HTML 功能可让您**直接在网页中**查看文档内容，同时保留页面的可交互性。例如 PPT 的动画、翻页效果，以及 EXCEL 文件的 sheet 切换等功能。该功能同样支持在链接中拼接参数的方式进行接入，同时支持以参数的方式配置**水印、防复制功能**。示例如下：

```
https://ci-h5-demo-1258125638.cos.ap-chengdu.myqcloud.com/defaults/defaultSlides.pptx?ci-process=doc-preview&dsttype=html&htmlwaterword=Q09T5paH5qGj6aKE6KeI&copyable=0
```

转换后效果如下图所示：